



**SequeLink<sup>®</sup>**

Developer's Reference

June 2002

© 2002 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 ADO, DataDirect Connect Integrator, DataDirect Connect JDBC, DataDirect Connect ODBC, DataDirect Connect OLE DB, DataDirect Connect Premium, DataDirect Reflector, and DataDirect SequeLink Integrator are trademarks of DataDirect Technologies. All other trademarks are the property of their respective owners.

Portions created by Netscape are Copyright (C) 1998-1999 Netscape Communications Corporation. All Rights Reserved.

Portions created by Eric Young are Copyright (C) 1995-1998 Eric Young. All Rights Reserved.

This product includes Xerces, developed by the Apache Software Foundation (<http://www.apache.org>). Copyright (C) 1999-2000 The Apache Software Foundation. All rights reserved.

This product includes Xalan, developed by the Apache Software Foundation (<http://www.apache.org>). Copyright (C) 1999-2000 The Apache Software Foundation. All rights reserved.

This product includes JDOM, developed by the JDOM Project (<http://www.jdom.org>). Copyright (C) 2000 Brett McLaughlin & Jason Hunter. 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 U.S. Government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of The Rights in Technical Data and Computer Software clause at DFAR 252.227-7013 et. seq. or subparagraphs (c)(1) and (2) of the Commercial Computer Software Restricted Rights at FAR 52.227-19, as applicable. Contractor is 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

<b>List of Tables</b> .....	<b>11</b>
<b>Preface</b> .....	<b>15</b>
What Is DataDirect SequeLink? .....	15
Using This Book .....	15
Other SequeLink Documentation .....	18
Conventions Used in This Book .....	20
Typographical Conventions .....	20
Environment-Specific Information .....	21
Ordering Printed Books .....	22
Contacting Technical Support .....	24
 <b>Part 1: Developing ODBC Applications</b>	
<b>1 Using the SequeLink ODBC Client</b> .....	<b>29</b>
About the SequeLink 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 odbc.ini Files . . . . .	46
Example: odbc.ini for Solaris . . . . .	46
Setting Environment Variables . . . . .	47
Using a Centralized odbc.ini File . . . . .	47
Connecting Using a Connection String . . . . .	48
ODBC Connection Attributes . . . . .	49
<b>2 Developing ODBC Applications . . . . .</b>	<b>59</b>
Required ODBC Libraries and Header Files . . . . .	60
UNIX Compiler Requirements . . . . .	61
ODBC API Functions . . . . .	62
SQL Escape Sequences . . . . .	65
Data Types and Isolation Levels . . . . .	65
Threading . . . . .	65
Threading Architecture . . . . .	65
Cancelling Functions in Multithreaded Applications . . . . .	67
Using Scrollable Cursors . . . . .	68
Static and Keyset-Driven Cursors . . . . .	68
Using Static Scrollable Cursors . . . . .	69
Using Keyset-Driven Scrollable Cursors . . . . .	70
Using Stored Procedures with Oracle . . . . .	70
Specifying Application IDs . . . . .	74
Specifying Application IDs Explicitly . . . . .	74
Generating Application IDs Automatically . . . . .	75
Persisting a Result Set as an XML Data File . . . . .	76
Error Handling . . . . .	77
SequeLink ODBC Driver Errors . . . . .	78
SequeLink Client Errors . . . . .	78
SequeLink Server Errors . . . . .	79
Database Errors . . . . .	79

Developing Performance-Optimized ODBC Applications . . . .	80
Catalog Functions . . . . .	80
Retrieving Data . . . . .	86
ODBC Function Selection. . . . .	90
Designing ODBC Applications. . . . .	93
Updating Data . . . . .	95

## Part 2: Developing ADO Applications

<b>3 Using the SequeLink ADO Client . . . . .</b>	<b>101</b>
About the SequeLink ADO Client. . . . .	101
Using the DataDirect Configuration Manager . . . . .	102
Working with the DataDirect Configuration Manager. . . . .	104
Displaying Data Source Properties . . . . .	105
Configuring ADO Client Data Sources . . . . .	107
Creating an ADO Client Data Source . . . . .	108
Modifying an ADO Client Data Source. . . . .	111
Renaming an ADO Client Data Source. . . . .	111
Deleting an ADO Client Data Source . . . . .	112
Copying an ADO Client Data Source . . . . .	112
Changing Data Source Directories . . . . .	113
Defining Default Setup Options. . . . .	114
Connecting to a SequeLink ADO Client . . . . .	117
Testing ADO Connections . . . . .	117
ADO Connection Dialogs. . . . .	118
Connecting with a Provider String . . . . .	125
ADO Connection Attributes . . . . .	126
<b>4 Developing ADO Applications . . . . .</b>	<b>131</b>
OLE DB Objects and Interfaces . . . . .	132
Supported Schema Rowsets . . . . .	134

Supported OLE DB Property Groups . . . . .	135
Data Source Property Group . . . . .	136
Data Source Information Property Group. . . . .	136
Initialization Property Group. . . . .	141
Rowset Property Group . . . . .	143
Session Property Group . . . . .	148
OLE DB Interfaces Supported in ADO . . . . .	148
Mapping ADO Methods and Properties . . . . .	150
ADO Command Object. . . . .	150
Connection Object . . . . .	153
Recordset Object . . . . .	161
Data Shaping . . . . .	166
Persisting Information . . . . .	167
Using Rowsets. . . . .	167
Unicode Support. . . . .	168
Mapping Data Types . . . . .	168
Specifying Application IDs . . . . .	169
Specifying Application IDs Explicitly . . . . .	169
Generating Application IDs Automatically . . . . .	169
Error Handling . . . . .	170
SequeLink ADO Provider Errors. . . . .	170
SequeLink Client Errors . . . . .	170
SequeLink Server Errors . . . . .	171
Database Errors . . . . .	171

## Part 3: Developing JDBC Applications

<b>5</b>	<b>Using JDBCTest. . . . .</b>	<b>175</b>
	JDBCTest Tutorial . . . . .	175
	Configuring JDBCTest . . . . .	176
	Starting JDBCTest . . . . .	177
	Connecting Using JDBCTest . . . . .	179
	Executing a Simple Select Statement . . . . .	184
	Executing a Prepared Statement . . . . .	186
	Retrieving Database Metadata . . . . .	190
	Scrolling Through a Result Set . . . . .	192
	Batch Execution on a Prepared Statement . . . . .	196
	Returning ParameterMetaData . . . . .	200
	Establishing Savepoints . . . . .	201
	Updatable Result Sets . . . . .	208
	LOB Support . . . . .	220
<b>6</b>	<b>Using the SequeLink Java Client . . . . .</b>	<b>227</b>
	About the SequeLink Java Client . . . . .	227
	SequeLink JDBC Driver . . . . .	228
	SequeLink Proxy Server . . . . .	228
	JDBCSpy . . . . .	229
	JDBCTest . . . . .	230
	jXTransformer . . . . .	230
	DataDirect Connection Pool Manager . . . . .	230
	J2EE Connector Architecture (JCA) Resource Adapter . . . . .	231
	SequeLink Java Client Directory Structure . . . . .	232
	Loading the SequeLink JDBC Driver . . . . .	235
	Specifying SequeLink JDBC Driver Connection URLs . . . . .	236
	Configuring JDBC Data Sources . . . . .	238
	Creating and Managing JDBC Data Sources . . . . .	239
	Using JNDI for Naming Databases . . . . .	239
	Using Connection Pooling . . . . .	240

Using the Java Transaction API . . . . .	243
J2EE Connector Architecture (JCA) Resource Adapter . . . . .	244
Using the Resource Adapter with an Application	
Server . . . . .	245
Using the Resource Adapter from an Application . . . . .	245
Specifying Connection Properties . . . . .	248
Using Connection URLs or the JDBC Driver	
Manager . . . . .	248
Using JDBC Data Sources . . . . .	248
JDBC Connection Properties . . . . .	249
Testing SequeLink JDBC Connections . . . . .	255
Using the SequeLink Java Client on a Java 2 Platform . . . . .	256
<b>7 Tracking JDBC Calls . . . . .</b>	<b>259</b>
About Spy . . . . .	259
Loading the Spy JDBC Driver . . . . .	260
Spy URL Syntax and Spy Attributes . . . . .	261
Using Spy with JDBC Data Sources . . . . .	262
Spy URL Examples . . . . .	263
Spy Log Example . . . . .	264
<b>8 Developing JDBC Applications . . . . .</b>	<b>267</b>
JDBC 3.0 Support . . . . .	268
JCA Resource Adapter Class . . . . .	269
SQL Escape Sequences . . . . .	269
Data Types and Isolation Levels . . . . .	269
Threading . . . . .	270
Threading Architecture . . . . .	270
Cancelling Functions in Multithreaded Applications . . . . .	270



Using Scrollable Cursors . . . . .	272
Result Set Types . . . . .	272
Concurrency Types . . . . .	273
Using Scrollable Cursors . . . . .	274
Specifying Application IDs . . . . .	275
Error Handling . . . . .	276
SequeLink JDBC Driver Errors . . . . .	276
SequeLink Server Errors . . . . .	276
Database Errors . . . . .	277
Fine-Tuning JDBC Application Performance . . . . .	278
Reducing Download Time . . . . .	278
Fetching BigDecimal Objects . . . . .	279
Using Database Metadata Methods . . . . .	280
Retrieving Data . . . . .	283
Selecting JDBC Objects and Methods . . . . .	285
Designing JDBC Applications . . . . .	288
Updating Data . . . . .	290

## Part 4: Reference

<b>A SQL Escape Sequences for ODBC and JDBC . . . . .</b>	<b>297</b>
Date, Time, and Timestamp Escape Sequences . . . . .	298
Scalar Functions . . . . .	298
String Functions . . . . .	305
Numeric Functions . . . . .	307
Date and Time Functions . . . . .	309
System Functions . . . . .	312
Like Predicate Escape Characters . . . . .	312
Outer Join Escape Sequences . . . . .	313
Procedure Call Escape Sequences . . . . .	315

- B Data Types and Isolation Levels . . . . . 317**
  - Supported Data Types . . . . . 317
    - DB2 V5 on OS/390. . . . . 318
    - DB2 V6, V7 . . . . . 319
    - Informix 9 . . . . . 320
    - Microsoft SQL Server 7 . . . . . 321
    - Microsoft SQL Server2000 . . . . . 323
    - Oracle8 . . . . . 324
    - Oracle9i . . . . . 325
    - Sybase . . . . . 327
    - SequeLink Legacy Server . . . . . 328
  - Isolation Levels . . . . . 328
- C JDBC Support . . . . . 331**
  - JDBC Compatibility. . . . . 331
  - Supported Functionality . . . . . 332
- D Connection Pool Manager . . . . . 383**
  - Creating a Data Source . . . . . 383
    - Creating a DataDirect SequeLink Data Source Object . . . . . 384
    - Creating a Data Source Using the DataDirect Connection Pool Manager . . . . . 386
  - Connecting to a Data Source. . . . . 389
  - Terminating the Pool Manager . . . . . 392
- Index . . . . . 393**

# List of Tables

Table 1-1.	ODBC Attributes . . . . .	49
Table 2-1.	Sources for Required ODBC Development Tools. . . . .	60
Table 2-2.	Compiler Requirements for UNIX . . . . .	61
Table 2-3.	ODBC Function Conformance for 2.x ODBC Applications . . . . .	62
Table 2-4.	Function Conformance for 3.x ODBC Applications. . . . .	64
Table 2-5.	Multithreading Functionality of the SequeLink ODBC Driver . . . . .	66
Table 2-6.	Using SQLCancel in Multithreaded Applications . . . . .	67
Table 2-7.	Support for Scrollable Cursors (ODBC) . . . . .	68
Table 3-1.	DataDirect Technologies Configuration Manager: Parts and Functions for SequeLink ADO Data Sources . . . . .	104
Table 3-2.	ADO Connection Attributes . . . . .	126
Table 4-1.	Objects and Interfaces Supported by the SequeLink ADO Provider . . . . .	132
Table 4-2.	OLE DB Schema Rowsets Supported by the SequeLink ADO Provider . . . . .	134
Table 4-3.	OLE DB Property Groups Supported by the SequeLink ADO Provider . . . . .	135
Table 4-4.	OLE DB Data Source Property Supported by the SequeLink ADO Provider . . . . .	136
Table 4-5.	OLE DB Data Source Information Properties Supported by the SequeLink ADO Provider . . . . .	137
Table 4-6.	Initialization Properties Supported by the SequeLink ADO Provider . . . . .	142
Table 4-7.	Rowset Properties Supported by the SequeLink ADO Provider. . . . .	144
Table 4-8.	Session Properties Supported by the SequeLink ADO Provider. . . . .	148

Table 4-9.	Supported OLE DB Interfaces Used by ADO. . . . .	148
Table 4-10.	Dynamic Properties Used for the ADO Command Object. . . . .	150
Table 4-11.	Mapping Methods Supported by the ADO Connection Object . . . .	153
Table 4-12.	Dynamic Properties Supported for the ADO Connection Object . . .	154
Table 4-13.	Mapping Methods Supported by the Recordset Object . . . . .	161
Table 4-14.	Dynamic Properties Used for the Recordset Object. . . . .	163
Table 6-1.	SequeLink Java Client Directory and Files . . . . .	232
Table 6-2.	Support for the Java Transaction API (JTA) by the SequeLink Java Client . . . . .	243
Table 6-3.	JDBC Properties . . . . .	249
Table 8-1.	Supported JDBC Features . . . . .	268
Table 8-2.	Using Cancel in Multithreaded JDBC Applications . . . . .	271
Table 8-3.	Support for Scrollable Cursors (JDBC). . . . .	273
Table A-1.	Scalar Functions Supported by the SequeLink ODBC Driver and SequeLink JDBC Driver . . . . .	299
Table A-2.	Scalar String Functions . . . . .	305
Table A-3.	Scalar Numeric Functions . . . . .	308
Table A-4.	Scalar Time and Date Functions . . . . .	310
Table A-5.	Scalar System Functions. . . . .	312
Table A-6.	Outer Join Escape Sequences Supported by the SequeLink ODBC Driver and SequeLink JDBC Driver . . . . .	314
Table B-1.	Data Types (DB2 V5) . . . . .	318
Table B-2.	Data Types (DB2 V6, V7) . . . . .	319
Table B-3.	Data Types (Informix 9) . . . . .	320
Table B-4.	Data Types (Microsoft SQL Server 7). . . . .	321
Table B-5.	Data Types (Microsoft SQL Server2000) . . . . .	323
Table B-6.	Data Types (Oracle8) . . . . .	324
Table B-7.	Data Types (Oracle9i) . . . . .	325

Table B-8.	Data Types (Sybase 11 and 12) . . . . .	327
Table B-9.	Data Types (Legacy Server) . . . . .	328
Table B-10.	Isolation Levels . . . . .	329
Table C-1.	JDBC Compatibility . . . . .	331



# Preface

This book is your guide to developing client applications for DataDirect® SequeLink® 5.3 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, and ActiveX Data Objects (ADO).

---

## 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 SequeLink ODBC Client” on page 29](#) provides information about using ODBC applications with the SequeLink ODBC Client.

- [Chapter 2 “Developing ODBC Applications” on page 59](#) provides information about developing ODBC applications for SequeLink environments.

## **Part 2: Developing ADO Applications**

- [Chapter 3 “Using the SequeLink ADO Client” on page 101](#) provides information about using ADO applications with the SequeLink ADO Client.
- [Chapter 4 “Developing ADO Applications” on page 131](#) provides information about developing ADO applications for SequeLink environments.

## **Part 3: Developing JDBC Applications**

- [Chapter 5 “Using JDBCTest” on page 175](#) introduces JDBCTest, 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 6 “Using the SequeLink Java Client” on page 227](#) provides information about using JDBC applications with the SequeLink Java Client.
- [Chapter 7 “Tracking JDBC Calls” on page 259](#) 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 267](#) provides information about developing JDBC applications for SequeLink environments.



## Part 4: Appendixes

- [Appendix A “SQL Escape Sequences for ODBC and JDBC” on page 297](#) 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 317](#) lists the data types and isolation levels supported for each data store supported by SequeLink.
- [Appendix C “JDBC Support” on page 331](#) provides information about JDBC compatibility and developing JDBC applications for SequeLink environments.
- [Appendix D “Connection Pool Manager” on page 383](#) provides sample code as an example of using the DataDirect Connection Pool Manager to allow your applications to handle connection pooling.

This SequeLink release includes jXTransformer, a development software component that works with DataDirect JDBC drivers to provide enhanced XML support. For information about jXTransformer, refer to the *jXTransformer User's Guide*.

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

---

## Other SequeLink Documentation

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

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

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/download/docs/dochome.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 SequeLink Java Client, HTML-based online help for developing JDBC 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. 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.

To access help from a GUI environment, navigate to the help directory and open the file:

```
help/wwhelp/js/html/frames.htm
```

To access help from a command-line environment, at a command prompt, enter:

```
browser_exe install_dir/help/wwhelp/js/html/frames.htm
```

where *browser\_exe* is the name of your browser executable and *install\_dir* is the path to the SequeLink installation directory.

---

# 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
<i>italics</i>	Introduces new terms that you may not be familiar with, and is used occasionally for emphasis.
<b>bold</b>	Emphasizes important information. Also indicates button, menu, and icon names on which you can act. For example, click <b>Next</b> .
UPPERCASE	Indicates the name of a file. For operating environments that use case-sensitive 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.
<i>monospaced italics</i>	Indicates names that are placeholders for values you specify; for example, <i>filename</i> .
forward slash /	Separates menus and their associated commands. For example, Select File / Copy means to select Copy from the File menu.
vertical rule	Indicates an OR separator to delineate items.

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

## Environment-Specific Information

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

Symbol	Environment
	<i>Windows.</i> Information specific to the Microsoft Windows 98, Windows Me, Windows NT, Windows 2000, and Windows XP environments is identified by the Windows symbol.
	<i>Windows NT.</i> Information specific to the Microsoft Windows NT environment is identified by the Windows symbol and the letters "NT."
	<i>Windows 2000.</i> Information specific to the Microsoft Windows 2000 environment is identified by the Windows symbol and the number "2000".
	<i>Windows XP.</i> Information specific to the Microsoft Windows XP environment is identified by the Windows symbol and the letters "XP".

**Symbol      Environment**

*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.

*OS/390*

*OS/390*. Information specific to OS/390 and z/OS environments is identified by the characters OS/390.

---

## Ordering Printed Books

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

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

# Order Form

Fax your request to DataDirect Technologies at (919) 461-4526.  
The cost of shipping will be added to your order.

SequeLink Key: \_\_\_\_\_

Ordered By: \_\_\_\_\_ Phone: ( \_\_\_\_ ) \_\_\_\_ - \_\_\_\_\_

Company Name: \_\_\_\_\_

Mailing Address: \_\_\_\_\_

City: \_\_\_\_\_ State: \_\_\_\_\_ Zip: \_\_\_\_\_

Credit Card: ☐ Master Card® ☐ VISA® ☐ Discover® ☐ American Express®

Credit Card Number: 

--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

Expiration: 

--	--

--	--

 Signature: \_\_\_\_\_  
Mo Yr

SequeLink Book Title	Price	Quantity	Total
Documentation Set	\$100.00		
<i>Getting Started with SequeLink</i>	\$35.00		
<i>SequeLink Installation Guide</i>	\$35.00		
<i>SequeLink Administrator's Guide</i>	\$35.00		
<i>SequeLink Developer's Reference</i>	\$35.00		
<i>jXTransformer User's Guide</i>	\$35.00		
<i>SequeLink Troubleshooting Guide and Reference</i>	\$35.00		

**Shipping:** Orders are shipped via Standard Airborne delivery from our Rockville Distribution Center. Items should arrive within 5 business days of receipt of order.

---

## 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](http://www.datadirect-technologies.com/support/support_index.asp).

Your user ID and password are issued to you by email upon registration.

For support after 30 days, 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. You also can download product fixes and upgrades for your DataDirect products.

### World Wide Web

[http://www.datadirect-technologies.com/support/support\\_index.asp](http://www.datadirect-technologies.com/support/support_index.asp)

### E-Mail

USA, Canada, and Mexico      [datadirect.answerline@datadirect-technologies.com](mailto:datadirect.answerline@datadirect-technologies.com)

Japan      [jpn.answerline@datadirect.co.jp](mailto:jpn.answerline@datadirect.co.jp)

All other countries      [int.datadirect.answerline@datadirect-technologies.com](mailto:int.datadirect.answerline@datadirect-technologies.com)



## Local Telephone Support

Local phone numbers can be found at:

[http://www.datadirect-technologies.com/support/support\\_contact\\_aline.asp](http://www.datadirect-technologies.com/support/support_contact_aline.asp)

Answerline 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 SequeLink ODBC Client” on page 29](#) provides information about using ODBC applications with the SequeLink ODBC Client.
- [Chapter 2 “Developing ODBC Applications” on page 59](#) provides information about developing ODBC applications for SequeLink environments.



# 1 Using the SequeLink ODBC Client

This chapter provides information about using ODBC applications with the SequeLink ODBC Client.

---

## About the SequeLink ODBC Client

The SequeLink ODBC Client supports ODBC applications through a component called the *SequeLink ODBC driver*. On Windows and UNIX platforms, the SequeLink 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 SequeLink ODBC driver can connect all commercial ODBC-compliant applications with server databases.

For instructions on installing the SequeLink 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 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 **SequeLink ODBC Client 5.3**, 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. For instructions on creating ODBC client data sources for UNIX, see [“Configuring ODBC Client Data Sources on UNIX” on page 46](#).

# Configuring ODBC Client Data Sources on Windows

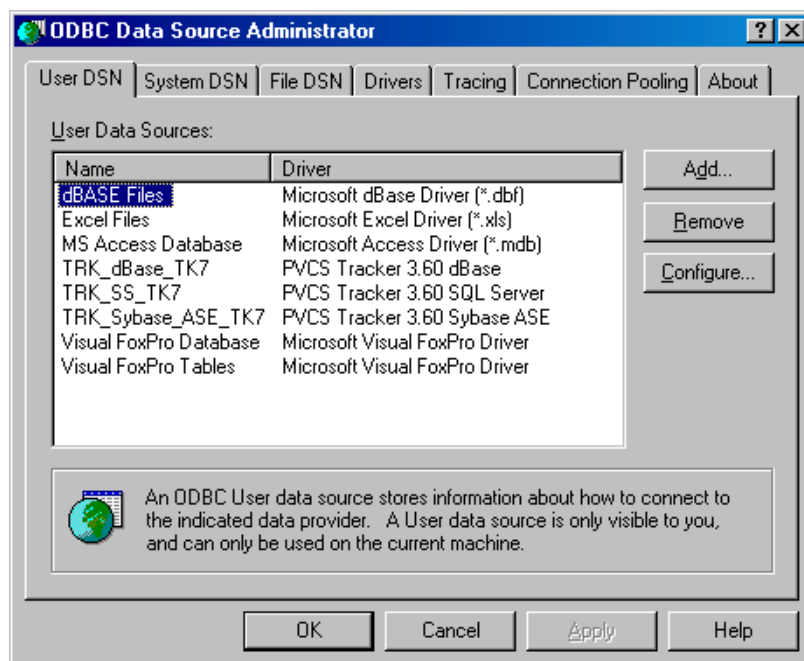


To configure client data sources for the SequeLink ODBC Client on Windows platforms, you use the ODBC Administrator.

## 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 **SequeLink ODBC Client 5.3**, and then select the **ODBC Administrator** application.

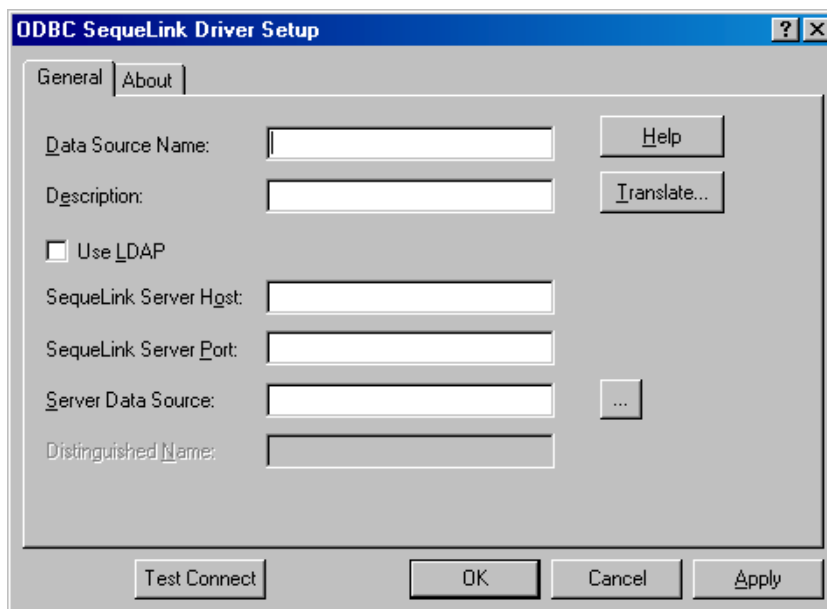
Click the **User DSN** tab or the **System DSN** tab to list user or system data sources, respectively.



- 2 To configure a new data source, click the **Add** button. A list of installed drivers appears. Select **DataDirect 32-BIT SequeLink 5.3**; then, click **Finish**.

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

The ODBC SequeLink Driver 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 SequeLink 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 SequeLink 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 SequeLink 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*.

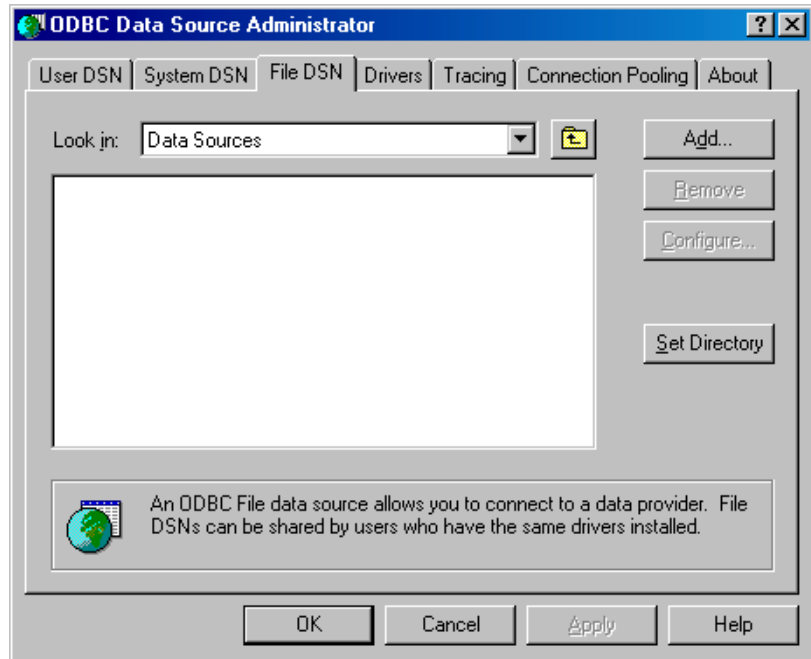
## Configuring ODBC File Client Data Sources

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:**

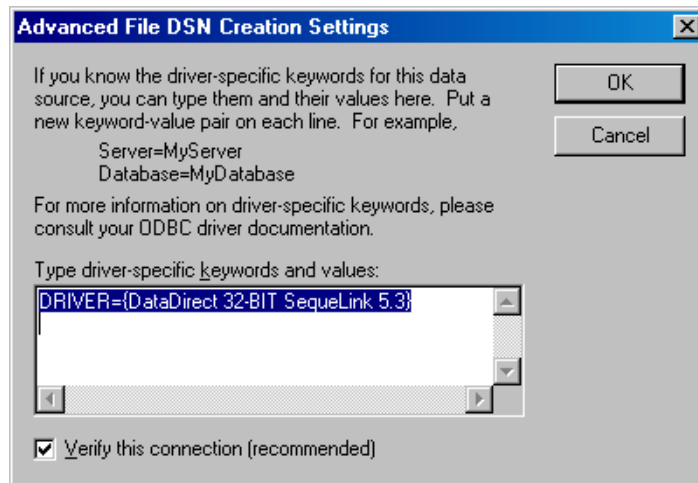
- 1 Start the ODBC Administrator by clicking **Start**, then **Programs**. From the Programs menu, select **SequeLink ODBC Client 5.3**, and then select the **ODBC Administrator** application.

- 2 Click the **File DSN** tab. The File DSN tab lists any file data sources in the specified directory.



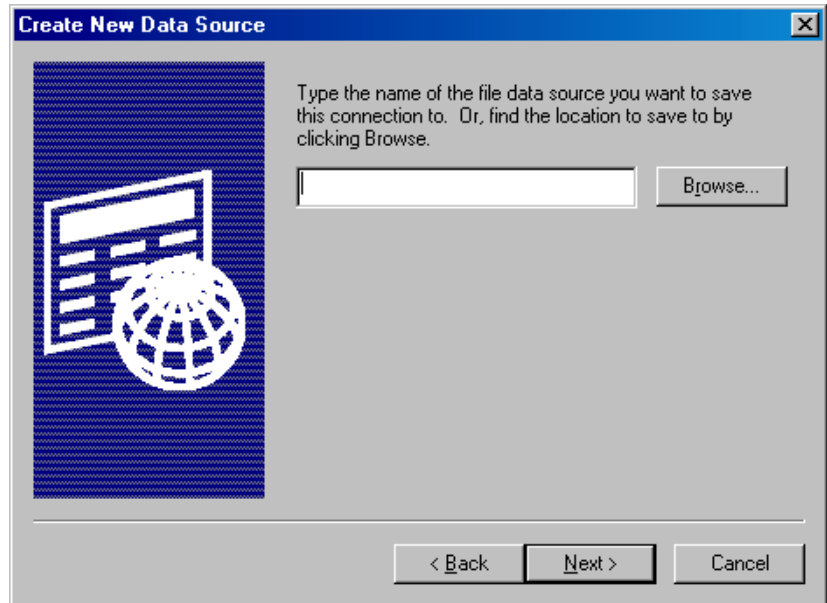
- 3 To configure a new data source, click the **Add** button. A list of installed drivers appears. Select **DataDirect 32-BIT SequeLink 5.3**; then, perform one of the following actions:
  - To configure the file data source to connect directly to a SequeLink Server without retrieving connection information 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.



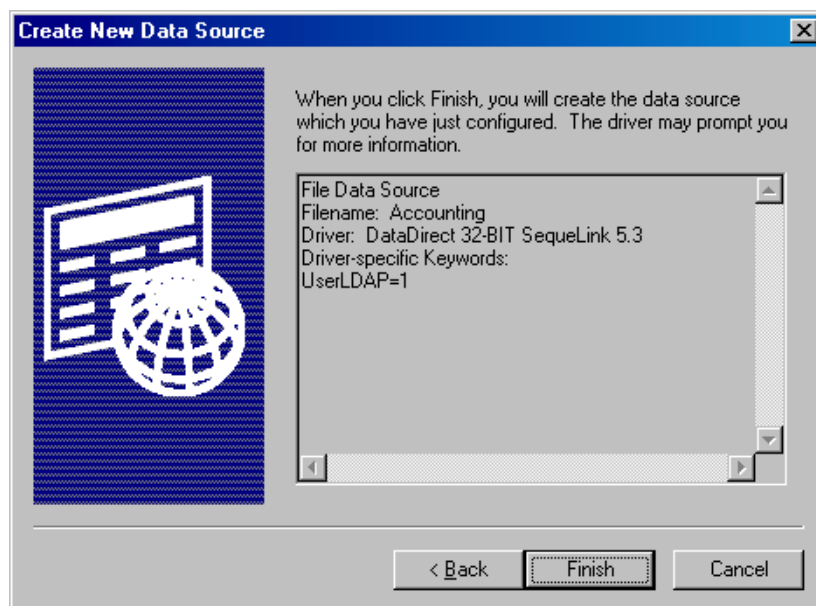
Type `UserLDAP=1` in the Type driver-specific keywords and values scrollable box; then, click **OK**. You are returned to the list of drivers. Click **Next** and continue with [Step 5](#).

- 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**.

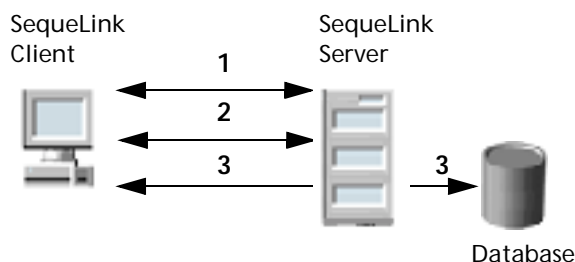
- 6 The Create New Data Source displays the settings you've configured for this data source.



- 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:



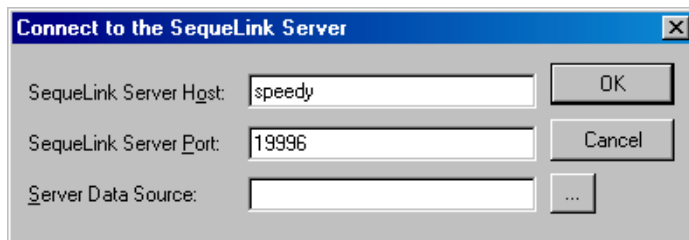
- 1 A network connection is established.
- 2 An authentication mechanism is used to establish the identity of the SequeLink Client to the SequeLink Server.
- 3 Based on information provided by the SequeLink Client application (for example, a database user name and password), a database connection is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

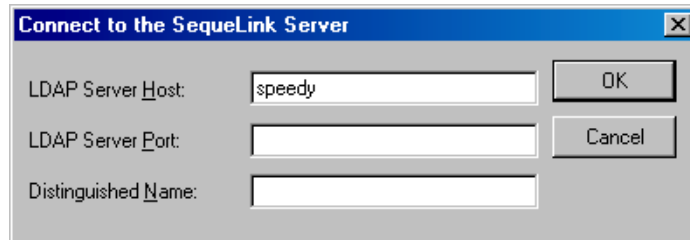
**SequeLink Server Port:** Type the TCP/IP port on which the SequeLink service is listening. A default installation of SequeLink Server uses the port 19996.

**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.

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

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

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

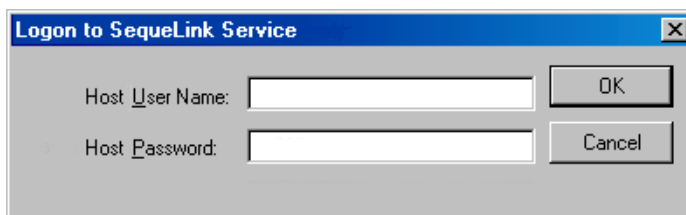
**LDAP Server Port:** Type the TCP/IP port on which the LDAP server is listening.

**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.



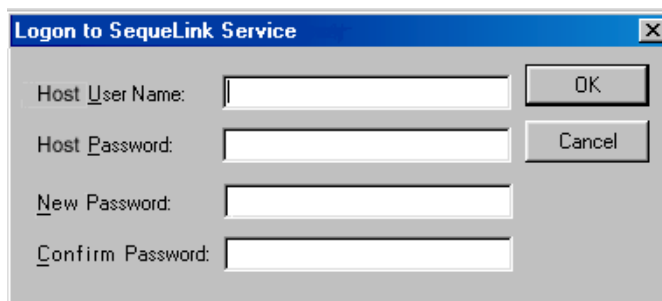
Provide the following information; then, click **OK**.

**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.

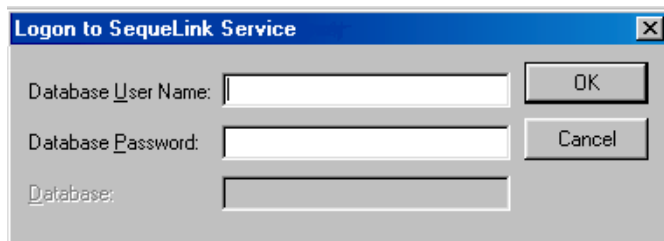
**Confirm Password:** Type again the new password to confirm it.

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.

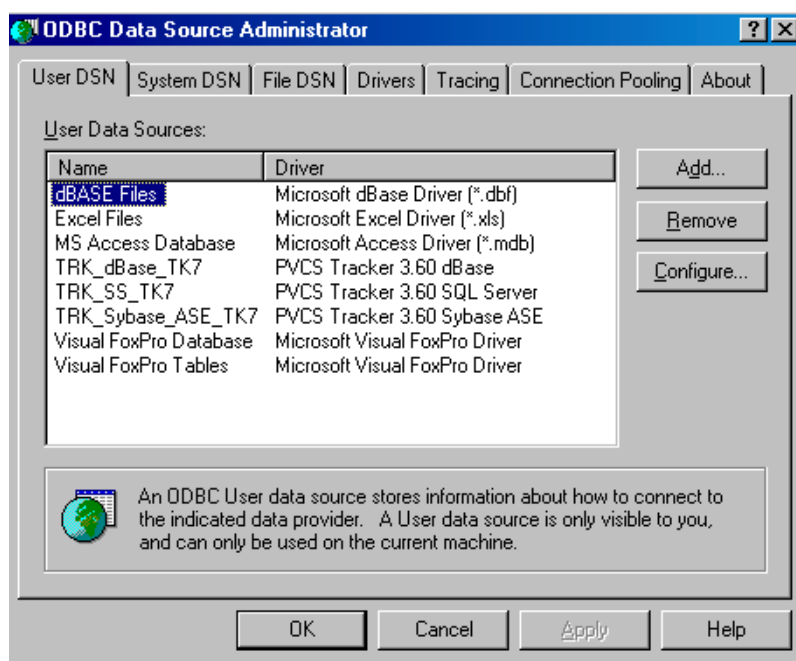
**Database Password:** Type the database password.

**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 **SequeLink ODBC Client 5.3**, 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 `odbc.ini` file and how to set some required environment variables to use the SequeLink ODBC Client on UNIX.

### Configuring `odbc.ini` Files

To configure an ODBC data source for UNIX, you must edit the `odbc.ini` file using the attributes in [Table 1-1 “ODBC Attributes” on page 49](#).

### Example: `odbc.ini` for Solaris

The following code shows an example of an `odbc.ini` file for a SequeLink ODBC Client installed on a Solaris machine.

```
[ODBC Data Sources]
DataSourceName=SALESDB

[SALESDB]
Driver=path_of_installdir/lib/ivslk16.so
Description=DataDirect 32-BIT SequeLink 5.3
Host=
Port=
UseLDAP=0
DistinguishedName=
[ODBC]
Trace=0
TraceFile=odbctrace.out
TraceDll=path_of_installdir/lib/odbctrace.so
InstallDir=path_of_installdir
```

where *path\_of\_installdir* is the path to the SequeLink ODBC Client installation directory.

## Setting Environment Variables

You must set several environment variables for the SequeLink 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
```

- If you are using the C shell, type:

```
source .sqlnk.csh
```

Executing this shell script sets the following environment variables:

ODBCINI	Specifies where the centralized <code>odbc.ini</code> file is located.
SQLNK_ODBC_HOME	Specifies the full path of the directory containing the SequeLink 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 `odbc.ini` 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:

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

- In the C shell:

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

The `odbc.ini` also requires 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, the following line must be in the [ODBC] section of the `odbc.ini` file:

```
InstallDir=/usr/slodbc53
```

---

## Connecting Using a Connection String

If your application requires a connection string to connect to a data source, you must specify the data source name that tells the driver which 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.

For a list of ODBC connection attributes and their valid values, see [“ODBC Connection Attributes” on page 49](#).

---

## ODBC Connection Attributes

[Table 1-1 “ODBC Attributes” on page 49](#) lists ODBC connection attributes in alphabetical order. The list includes long and short names and provides a description of each attribute. The defaults listed in [Table 1-1](#) 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. In [Table 1-1](#), short names are shown enclosed within parentheses ( ).

---

**Table 1-1. ODBC Attributes**

---

Attribute	Description
ApplicationID (APPID)	<p>Specifies the application ID that identifies the client application to the SequeLink service. This attribute is only required when the SequeLink service you are connecting to has been configured to limit access to specific applications.</p> <p>For more information about using application IDs to limit access to SequeLink services, see <a href="#">“Specifying Application IDs” on page 74</a>.</p>

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

Attribute	Description
ApplicationName (APPNAME)	Identifies the application that is establishing the connections.  The default is SequeLink ODBC Application.
AutomaticApplicationID (AUTOAPPID)	Specifies an application ID that is automatically generated by the SequeLink 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.  For more information about using application IDs to limit access to SequeLink services, see <a href="#">“Specifying Application IDs” on page 74</a> .
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 SequeLink ODBC Client does not lock the expected row.  When set to 0, the appropriate row is locked.  When set to 1 (the initial default), the appropriate row is not locked.  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.

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

Attribute	Description
DistinguishedName (DN)	Specifies the distinguished name identifying the LDAP entry from which connection information is retrieved. This attribute is required when UseLDAP=1.
EnableDescribeParam (EDP)	<p>EnableDescribeParam={0   1}. Specifies a workaround connection attribute for connections to Oracle data stores only.</p> <p>When set to 0 (the initial default), support is turned off for SQLDescribeParam.</p> <p>When set to 1, support is turned on for SQLDescribeParam and will describe all parameters as SQL_CHAR with a precision of 999.</p>
FetchNextOnly (FNO)	<p>FetchNextOnly={TRUE   FALSE}. Turns on a workaround for Visual Basic/Remote Data Objects (RDO) that circumvents a problem with FORWARD_ONLY cursors when the driver reports other values than FETCH_NEXT for SQLGetInfo(SQL_FETCH_DIRECTION).</p> <p>For example, if the driver only reports FETCH_NEXT, RDO performs SQLExecDirect, SQLBindCol, and SQLExtendedFetch(NEXT). If the driver supports more than FETCH_NEXT, RDO performs SQLExecDirect, SQLExtendedFetch(NEXT), and SQLGetData. This is only valid when the rowsize is 1, but RDO uses a larger rowsize in this situation.</p> <p>When set to TRUE, the driver will incorrectly report that only SQL_FETCH_NEXT is supported, which satisfies RDO.</p> <p>When set to FALSE (the initial default), the driver will correctly report other values than SQL_FETCH_NEXT.</p>

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

Attribute	Description
FixCharTrim (FCT)	<p>FixCharTrim={0   1}. Turns on a workaround for applications that have a problem using SQL_CHAR data padded with spaces. The SequeLink ODBC driver returns SQL_CHAR data padded with spaces as mandated by the ODBC specification.</p> <p>When set to 0 (the initial default), the workaround is turned off.</p> <p>When set to 1, SQL_CHAR data that is not padded with spaces is returned.</p>
GetOutputParams (GOP)	<p>Turns on a workaround that allows you to control when output parameters of stored procedures are returned to calling applications. This attribute uses a bitmask with the following options:</p> <p>When set to 1, output parameters are returned after an execute.</p> <p>When set to 2, output parameters are returned after a fetch is complete.</p> <p>When set to 4, output parameters are returned after moreresults returns no more rows.</p> <p>When set to 7 (the initial default), output parameters are returned after all of the above.</p> <p>The value for this connection attribute should be set to the cumulative value of all chosen options added together.</p> <p>NOTE: Set GetOutputParams=3 when executing stored procedures with output parameters in RDO (Visual Basic 5 and 6).</p>
HLogonID (HUID)	<p>Specifies the host user name, which may be required depending on the server configuration.</p>
HPassword (HPWD)	<p>Specifies the host password, which may be required depending on the server configuration.</p>

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

Attribute	Description
Host (HST)	<p>Specifies the TCP/IP address of the SequeLink Server, specified in dotted format or as a host name.</p> <p>LDAP: If LDAP is enabled, this attribute identifies the TCP/IP address of the LDAP server. This attribute can also be a list of LDAP servers separated by a blank space (for example, "ld1.foo.com ld2.foo.com ld3.foo.com"). If the first LDAP server in the list does not respond, the SequeLink ODBC Client will try to connect to the next LDAP server in the list.</p>
LogonID (UID)	<p>Specifies the host or data store user name, which may be required depending on the server configuration.</p>
NewPassword (NPWD)	<p>Specifies the new host password to be used. If specified and applicable to the connection, the SequeLink password change mechanism is invoked. When the password has been changed successfully, the following warning is generated:</p> <pre data-bbox="701 973 1200 1069">[DataDirect][ODBC SequeLink driver][SequeLink Server] The user password was changed successfully</pre> <p>If unspecified and the SequeLink Server detects that the host password has expired, you will be prompted for a new host password.</p> <p>For more information about the SequeLink password change mechanism, refer to the <i>SequeLink Administrator's Guide</i>.</p>
Password (PWD)	<p>Specifies the host or data store password, which may be required depending on the server configuration.</p>

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

Attribute	Description
Port (PRT)	<p>Specifies the TCP/IP port on which the SequeLink Server is listening.</p> <p>LDAP: If LDAP is enabled, this attribute identifies the TCP/IP port on which the LDAP server is listening. If you do not specify a port, the default port for LDAP (389) will be used.</p>
ServerDataSource (SDSN)	<p>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.</p>
SessionConnectTimeout (SCTO)	<p>Imposes a time limit on:</p> <ul style="list-style-type: none"><li>■ The establishment of the TCP/IP connection with the server.</li><li>■ 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).</li></ul>
SLKStaticCursorLongColBuffLen (SSCLCBL)	<p>Turns on a workaround that allows you to specify the amount of data (in KB) that is buffered for SQL_LONGVARCHAR and SQL_LONGVARBINARY columns with a static cursor.</p> <p>The default is 4.</p>

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

Attribute	Description
UseLDAP (LDAP)	<p>UseLDAP={0   1}. Determines whether the parameters to establish a connection to the SequeLink Server should be retrieved from LDAP.</p> <p>When set to 0 (the initial default), the SequeLink Client will connect directly to the specified SequeLink Server.</p> <p>When set to 1, the SequeLink Client will retrieve the TCP/IP host, TCP/IP port, and SequeLink server data source (optional) from an LDAP entry identified by a Distinguished Name (DN). Once the connection information is retrieved, the SequeLink Client will connect directly to the specified SequeLink Server. The DistinguishedName (DN) attribute is required.</p>
WorkArounds (WA)	<p>Turns on workarounds that allow you to take full advantage of the SequeLink ODBC driver with ODBC applications requiring nonstandard or extended behavior.</p> <p>IMPORTANT: Each of these options has potential side effects related to its use. An option should only be used to address the specific problem for which it was designed.</p> <p>When set to 1, the ODBC driver returns 1, allowing Microsoft Access to open tables as read-write. If an ODBC driver reports to Microsoft Access 2.0 that its SQL_CURSOR_COMMIT_BEHAVIOR or SQL_CURSOR_ROLLBACK_BEHAVIOR is 0, Microsoft Access opens tables as read-only.</p> <p>When set to 2, the driver reports that qualifiers are not supported. This option is provided because some applications cannot handle database qualifiers.</p>

Table 1-1. ODBC Attributes (cont.)

Attribute	Description
WorkArounds (cont.)	<p>When set to 4, the driver detects when Visual Basic requires multiple connections to a DBMS and has the multiple ODBC connections share a single physical connection to the DBMS. For DBMSs that support only a single connection, the second attempt fails.</p> <p>When set to 8, the driver returns 1. However, if an ODBC driver cannot detect the number of rows that are affected by an Insert, Update, or Delete statement, it may return -1 in SQLRowCount. Some products cannot handle this.</p> <p>When set to 16, the driver returns no INDEX_QUALIFIER, allowing Microsoft Access to open the table. If an ODBC driver in SQLStatistics reports to Microsoft Access 1.1 that an INDEX_QUALIFIER contains a period, Microsoft Access returns a tablename is not a valid name error.</p> <p>When set to 32, users of flat-file drivers are allowed to abort a long-running query by pressing the ESC key.</p> <p>When set to 64, the result is a column name of <i>Cposition</i> where <i>position</i> is the ordinal position in the result set. For example:</p> <pre>SELECT col1, col2+col3 FROM table1</pre> <p>produces the column names col1 and C2. SQLColAttributes/SQL_COLUMN_NAME returns an empty string for result columns that are expressions. Use this option for applications that cannot handle empty strings in column names.</p> <p>When set to 256, SQLGetInfo/SQL_ACTIVE_CONNECTIONS is forced to return as 1.</p> <p>When set to 512, the SQLSpecialColumns function returns a unique index as returned from SQLStatistics to prevent ROWID results.</p>



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

Attribute	Description
<b>WorkArounds</b> <i>(cont.)</i>	<p data-bbox="701 317 1275 409">When set to 2048, SQLDriverConnect returns "Database=" instead of "DB=" in the returned connection string.</p> <p data-bbox="701 425 1300 579">When set to 65536, trailing zeros are stripped from decimal results, which prevents Microsoft Access from generating an error when decimal columns containing trailing zeros are included in the unique index.</p> <p data-bbox="701 595 1318 812">When set to 131072, all occurrences of the double quote character (") are turned into the accent grave character (`). Some applications always quote identifiers with double quotes. Double quoting causes problems for data sources that do not return SQLGetInfo/ SQL_IDENTIFIER_QUOTE_CHAR = ".</p> <p data-bbox="701 828 1243 920">When set to 524288, the precision and scale settings for SQL_DECIMAL parameters are overridden to precision 40 and scale 20.</p> <p data-bbox="701 935 1318 1124">When set to 8388608, SQLGetInfo/SQL_DATABASE_NAME is returned as an empty string when SQLGetInfo/ SQL_MAX_QUALIFIER_NAME_LEN is 0. This option should be used with Inprise/Borland tools, such as Delphi.</p> <p data-bbox="701 1140 1300 1232">When set to 536870912, SQLBindParameter is allowed to be called after SQLExecute to change the precision of previously bound parameters.</p> <p data-bbox="701 1248 1318 1402">When set to 1073741824, Microsoft Access assumes that ORDER BY columns do not have to be in the SELECT list. This option provides a workaround for data stores that always use ORDER BY columns.</p>

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

Attribute	Description
WorkArounds2 (WA2)	<p>Turns on workarounds that allow you to take full advantage of the SequeLink ODBC driver with ODBC applications requiring nonstandard or extended behavior.</p> <p>IMPORTANT: Each of these options has potential side effects related to its use. An option should only be used to address the specific problem for which it was designed.</p> <p>When set to 2, the driver ignores the ColumnSize/DecimalDigits specified by the application and uses the database defaults instead. Some applications incorrectly specify ColumnSize/DecimalDigits when binding timestamp parameters.</p> <p>When set to 4, Microsoft Access uses the most recent native type mapping, as returned by SQLGetTypeInfo, for a specific SQL type. This option reverses the order in which types are returned, so that Microsoft Access will use the most appropriate native type. This option is recommended if you are using Microsoft Access against an Oracle8 data store.</p> <p>When set to 32, Microsoft Access requires that the characters "DSN=" are returned by SQLDriverConnect in the connection string output parameter.</p>

## 2 Developing ODBC Applications

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

- [“Required ODBC Libraries and Header Files” on page 60](#)
- [“UNIX Compiler Requirements” on page 61](#)
- [“ODBC API Functions” on page 62](#)
- [“SQL Escape Sequences” on page 65](#)
- [“Data Types and Isolation Levels” on page 65](#)
- [“Threading” on page 65](#)
- [“Using Scrollable Cursors” on page 68](#)
- [“Using Stored Procedures with Oracle” on page 70](#)
- [“Specifying Application IDs” on page 74](#)
- [“Persisting a Result Set as an XML Data File” on page 76](#)
- [“Error Handling” on page 77](#)
- [“Developing Performance-Optimized ODBC Applications” on page 80](#)

---

# 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 98, Windows Me, Windows NT, Windows 2000, Windows XP	Microsoft Open Database Connectivity Software Development Kit (SDK).
	UNIX	The required header files and libraries are shipped with the SequeLink ODBC Client.

---

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

---

# UNIX Compiler Requirements

The SequeLink 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	C Set++ for AIX 3.1.4.0 (using the reentrant version, xLC_r) or higher
Solaris	SUNWspro (SPARCompiler) 6.0 or higher
Linux	GNU project C++ Compiler version 2.96
HP-UX 11 aCC	HP aC++ Compiler version A.03.05 or higher

---

# ODBC API Functions

The SequeLink 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

---

**Table 2-3. ODBC Function Conformance for 2.x ODBC Applications** *(cont.)*

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

---



---

## SQL Escape Sequences

See [Appendix A “SQL Escape Sequences for ODBC and JDBC” on page 297](#) for information about the SQL escape sequences supported by the SequeLink ODBC driver.

---

## Data Types and Isolation Levels

The data types and isolation levels supported by the SequeLink ODBC driver depend on the data store to which you are connecting. See [Appendix B “Data Types and Isolation Levels” on page 317](#) 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 SequeLink ODBC driver is platform dependent as shown in [Table 2-5](#).

**Table 2-5. Multithreading Functionality of the SequeLink 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 2-6](#).

- *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 2-6. Using `SQLCancel` in Multithreaded Applications**

---

Data Store	SQLCancel
DB2 V5, V6, V7R1 on OS/390	Ignored
DB2 V6 and V7R2 on Windows	OK
DB2 V6 and V7R2 on UNIX	Ignored
Informix 9, 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 2-7](#) shows the type of scrollable cursors supported for each database.

**Table 2-7. Support for Scrollable Cursors (ODBC)**

Database	Static	Keyset-Driven
DB2 on OS/390	X	
DB2 on Windows	X	
Informix	X	X
Legacy Server	X	
Microsoft SQL Server (see note)	X	X
ODBC Socket	X	

NOTE: To use keyset-driven cursors with Microsoft SQL Server and Sybase, the table must contain an identity column.

**Table 2-7. Support for Scrollable Cursors** (*cont.*) (ODBC)

Database	Static	Keyset-Driven
Oracle 8 and 9	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 SequeLink ODBC driver supports static cursors for all types of result set generating statements, including result sets generated by stored procedures.
- The SequeLink ODBC driver supports LOB data for static cursors; however, by default, only the first 4096 bytes of the LOB column is buffered. For more information about specifying the amount of data that is buffered, see the `SLKStaticCursorLongColBuffLen` connection attribute in [Table 1-1 “ODBC Attributes” on page 49](#).
- To persist a result set as an XML data file with embedded schema, you must use static cursors. For more information about persisting XML data files, see [“Persisting a Result Set as an XML Data File” on page 76](#).

## Using Keyset-Driven Scrollable Cursors

- The SequeLink ODBC driver does not support using keyset-driven cursors on stored procedures or explicit batches.
- The SequeLink 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 SequeLink 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 SequeLink 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 SequeLink 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 SequeLink ODBC driver allows you to persist a result set as an XML data file with embedded schema. To implement XML persistence, a client application must do the following:

- 1 Turn on STATIC cursors. For example:

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

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

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

- 2 Execute a SQL statement. For example:

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

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

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

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

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

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

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

Function Sequence Error

---

## Error Handling

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

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

## SequeLink ODBC Driver Errors

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

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

For example:

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

The native error code is always zero (0).

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 available from Microsoft. The *ODBC 3.0 Software Development Kit and Programmer's Reference* is available from Microsoft Press. For information on later versions of ODBC, refer to the documentation included in the ODBC SDK.

## SequeLink Client Errors

An error generated by the SequeLink 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 80](#)
- [“Retrieving Data” on page 86](#)
- [“ODBC Function Selection” on page 90](#)
- [“Designing ODBC Applications” on page 93](#)
- [“Updating Data” on page 95](#)

## Catalog Functions

The following ODBC functions are catalog functions:

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

SQLGetTypeInfo is listed as a potentially performance-expensive ODBC function, because many drivers must query the server to obtain accurate information about which data types are supported (for example, to find dynamic data types such as user-defined types).



## ***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. For example, call `SQLGetTypeInfo` once in the application and cache the elements of the result set on which your application depends. It is unlikely that any application uses every element of the result set generated by a catalog function, so the cached information should not be difficult to maintain.

## ***Avoiding Search Patterns***

Passing null arguments 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 OS/390 only)

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

### OS/390 ***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 OS/390 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, (SQLLEN)
    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 SequeLink ADO Client” on page 101](#)  
provides information about using ADO applications with the SequeLink ADO Client.
- [Chapter 4 “Developing ADO Applications” on page 131](#)  
provides information about developing ADO applications for SequeLink environments.



## 3 Using the SequeLink ADO Client

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

---

### About the SequeLink ADO Client

The SequeLink ADO Client supports ADO/OLE DB applications through a component called the *SequeLink ADO provider*. The SequeLink 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 SequeLink 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 SequeLink ADO provider. This book provides programming information for developers who want to control data source connections from within their applications.

After you install the SequeLink 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 SequeLink 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 SequeLink 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 107](#) for instructions on creating and configuring data sources.

After you have defined SequeLink ADO data sources, you can access them from your data consumer and connect to them. For more information, see [“Testing ADO Connections” on page 117](#).

---

## Using the DataDirect Configuration Manager

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

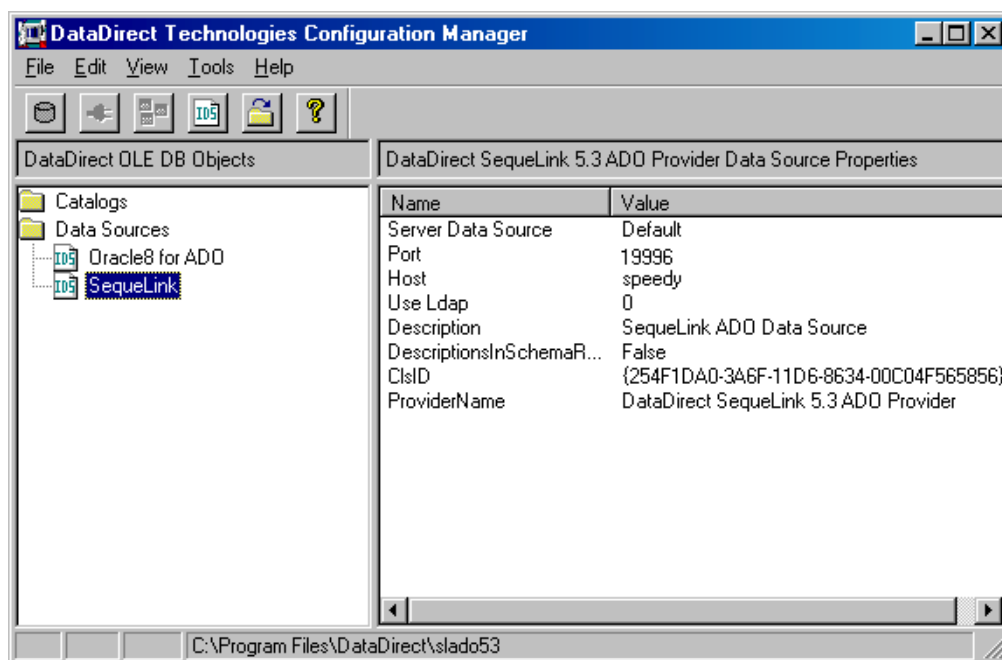
To start the Configuration Manager, select **Start / Programs**, and select **SequeLink ADO Client 5.3**. Then, select the **DataDirect Configuration Manager** application.

The Configuration Manager window is divided into two panes. As [Figure 3-1 on page 103](#) 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 SequeLink 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\slado53` directory.

# Working with the DataDirect Configuration Manager

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

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

**Table 3-1. DataDirect Technologies Configuration Manager: Parts and Functions for SequeLink 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 Technologies Configuration Manager.
	Edit	■ Delete a data source.
		■ Rename a data source.
		■ Modify a data source.

**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 SequeLink ADO Data Sources** (cont.)

Use this element...	To do this...
View	<ul style="list-style-type: none"> <li>■ View or hide the toolbar and status bar.</li> </ul>
Tools	<ul style="list-style-type: none"> <li>■ Refresh the Configuration Manager.</li> <li>■ Change the directory in which to look for data sources.</li> <li>■ Define a Template data source directory.</li> <li>■ Define a Master data source directory.</li> </ul>
Help	View online help.
Vertical splitter bar	Click on the bar and drag it to the right or left to change the size of the left and right panes.
Status bar	<ul style="list-style-type: none"> <li>■ Show the current keyboard state, including when Num Lock, Scroll Lock, and Caps Lock are turned on.</li> <li>■ Show the current directory.</li> </ul>

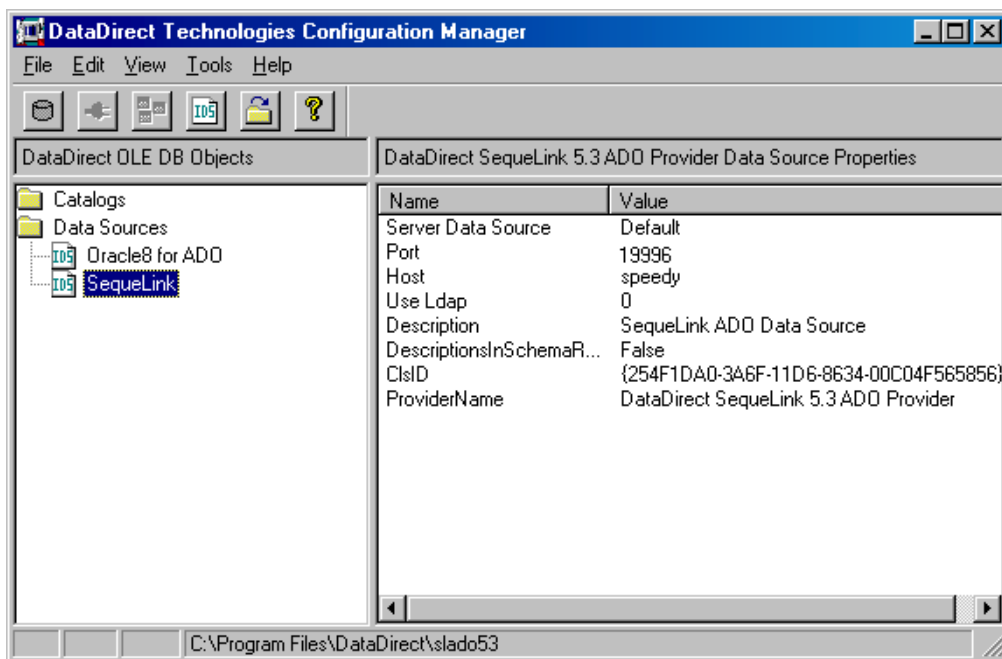
**Shortcut Tip:** Right-clicking an item in the left pane displays a pop-up menu that allows you to perform the same actions that are available from the toolbar and menu bar.

## Displaying Data Source Properties

- 1 Start the Configuration Manager. To start the Configuration Manager, select **Start / Programs**, and select **SequeLink ADO Client 5.3**. 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.

---

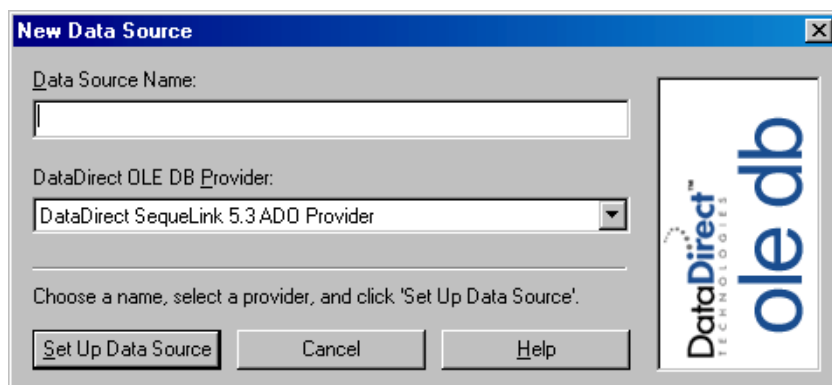
## Configuring ADO Client Data Sources

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

- [“Creating an ADO Client Data Source” on page 108](#)
- [“Modifying an ADO Client Data Source” on page 111](#)
- [“Renaming an ADO Client Data Source” on page 111](#)
- [“Deleting an ADO Client Data Source” on page 112](#)
- [“Copying an ADO Client Data Source” on page 112](#)
- [“Changing Data Source Directories” on page 113](#)
- [“Defining Default Setup Options” on page 114](#)

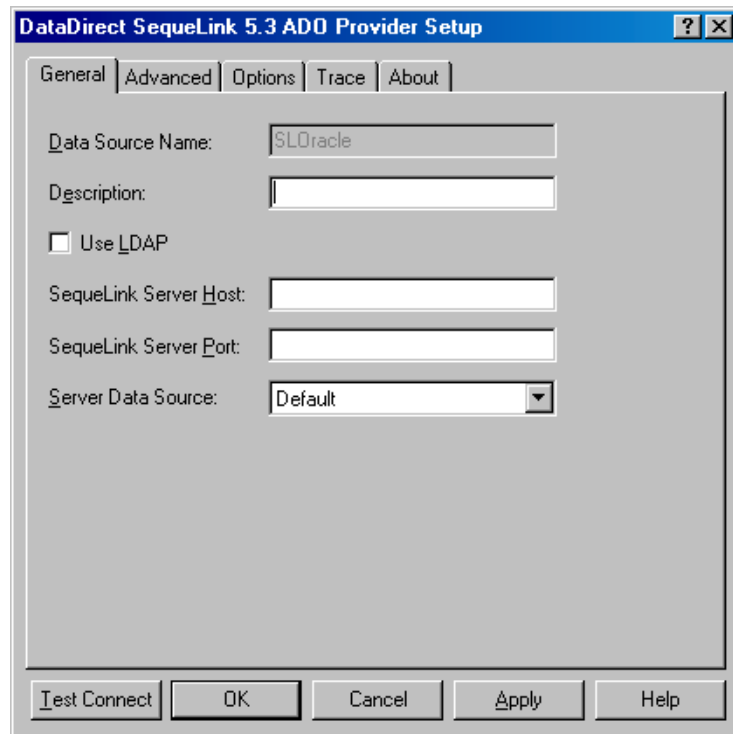
## Creating an ADO Client Data Source

- 1 Start the DataDirect Configuration Manager. To start the Configuration Manager, select **Start / Programs**, and select **SequeLink ADO Client 5.3**. 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 5.3 ADO Provider**.

- 5 Click the **Set Up Data Source** button. The DataDirect SequeLink ADO Provider Setup window appears.



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

- 6 Provide the following information.

**Data Source Name:** This is a read-only field that uniquely identifies this ADO data source configuration. Examples include "Accounting" or "SequeLink to Oracle Data".

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

**SequeLink Server Host:** Type the TCP/IP host name of the SequeLink service to which you want the SequeLink 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 SequeLink 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 SequeLink 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. For instructions on changing the current directory, see [“Changing Data Source Directories” on page 113](#).

## 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 ADO Provider Setup window. See [“Creating an ADO Client Data Source” on page 108](#) 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 a SequeLink ADO Provider data source:**

- 1 Start the Configuration Manager. To start the Configuration Manager, select **Start / Programs**, and select **SequeLink ADO Client 5.3**. 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 **SequeLink ADO Client 5.3**. 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 SequeLink 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 SequeLink 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 SequeLink 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

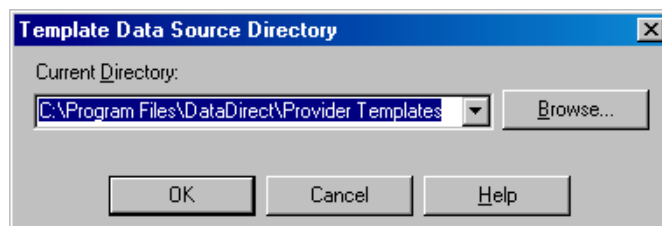
You can define template data source files to simplify the creation of data source files. A template data source file allows you to define the default setup options for SequeLink data providers. The Configuration Manager supplies these values in the Setup dialog box when a user creates a new data source.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

To create a template data source file:

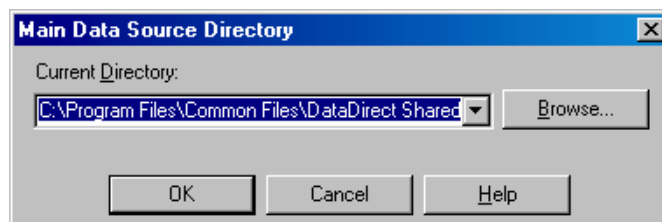
- 1 Create a directory in which to store the template data source file.

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.

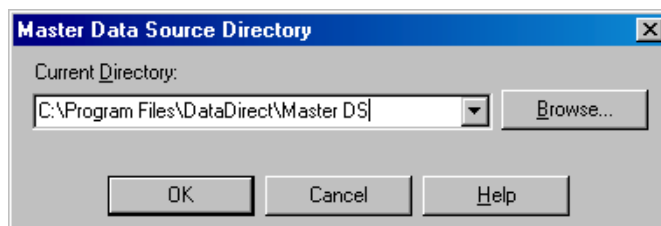
During connection, the Main data source directory is checked for a data source, and connection values are retrieved. If a Master data source directory exists, it is then checked for the same data source. The connection settings for user data sources will be overridden by the master data source file.

**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 a SequeLink ADO Client

You can connect to a data source using a Connection window, or using a provider string. For information about connecting using an ADO provider string, see [“Connecting with a Provider String” on page 125](#).

## Testing ADO Connections

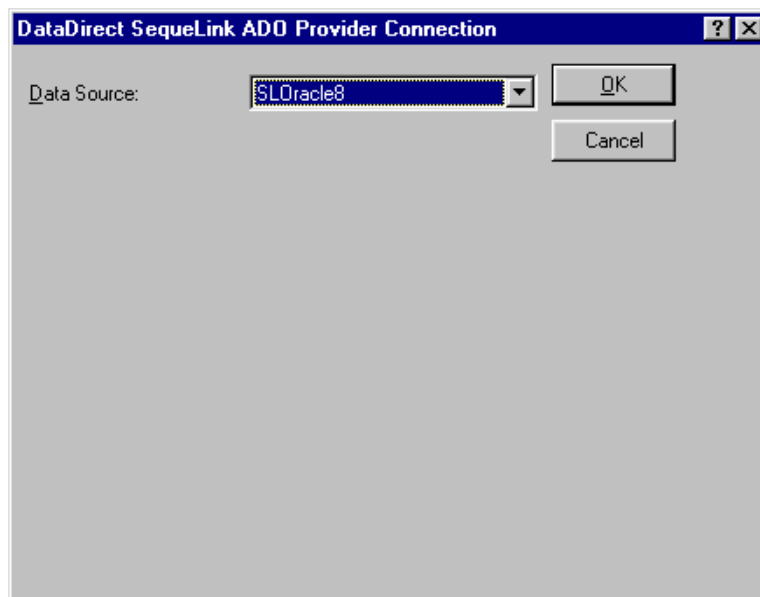
The SequeLink ADO Provider opens a Connection window when you perform either of the following actions:

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

For more information about ADO connection dialogs that may appear, see [“ADO Connection Dialogs” on page 118](#).

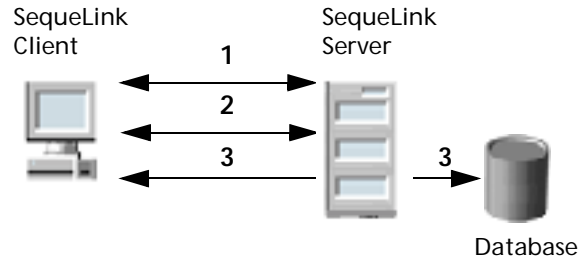
## ADO Connection Dialogs

When your data consumer requests the SequeLink ADO Provider to prompt for missing connection parameters and an ADO data source has not been specified, the DataDirect SequeLink 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:



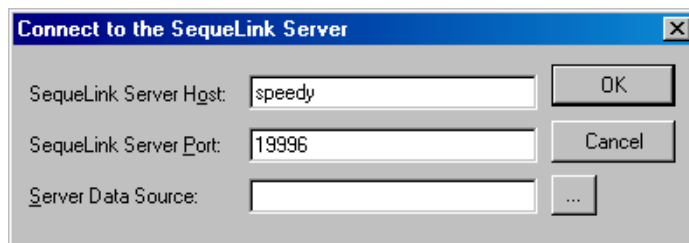
- 1 A network connection is established.
- 2 An authentication mechanism is used to establish the identity of the SequeLink Client to the SequeLink Server.
- 3 Based on information provided by the SequeLink Client application (for example, a database user name and password), a database connection is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

A screenshot of a Windows-style dialog box titled "Connect to the SequeLink Server". The dialog has a blue title bar with a close button (X) in the top right corner. It contains three text input fields and three buttons. The first field is labeled "SequeLink Server Host:" and contains the text "speedy". The second field is labeled "SequeLink Server Port:" and contains the text "19996". The third field is labeled "Server Data Source:" and is empty. To the right of the first field is an "OK" button. To the right of the second field is a "Cancel" button. To the right of the third field is a button with three dots "...".

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

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

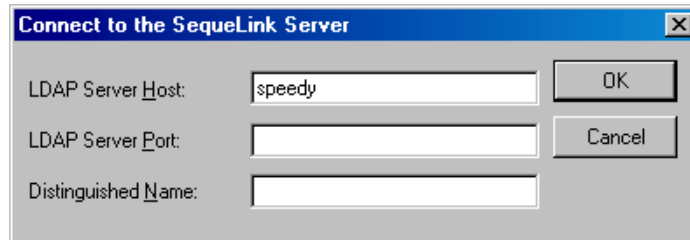
**SequeLink Server Port:** Type the TCP/IP port on which the SequeLink service is listening. A default installation of SequeLink Server uses the port 19996.

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



## ***Retrieving Connection Information from an LDAP Directory***

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

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

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

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

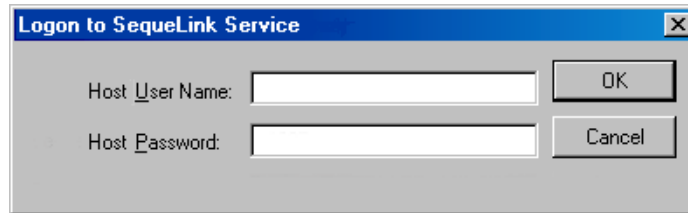
**LDAP Server Port:** Type the TCP/IP port on which the LDAP server is listening.

**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.



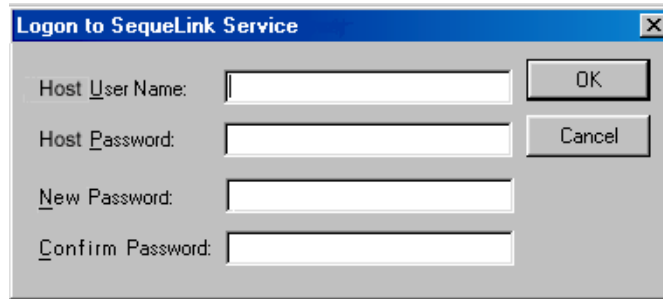
Provide the following information; then, click **OK**.

**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.

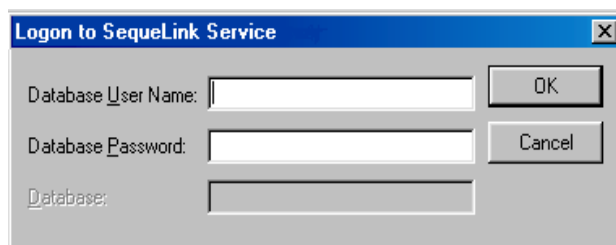
**Confirm Password:** Type again the new password to confirm it.

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 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;"
```

For a list of ADO connection attributes, see [“ADO Connection Attributes” on page 126](#).

# ADO Connection Attributes

Table 3-2 provides a list of ADO connection attributes supported by the SequeLink 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	<p>Specifies the application ID that identifies the client application to the SequeLink service. This attribute is only required when the SequeLink service you are connecting to has been configured to limit access to specific applications.</p> <p>For more information about using application IDs to limit access to SequeLink services, see <a href="#">“Specifying Application IDs” on page 169</a>.</p>
ApplicationName	<p>Identifies the application that is establishing the connections (for example, ApplicationName=Account01) and can be used to identify where problems that are associated with a particular application occur.</p> <p>The default is SequeLink ADO Application.</p>
Automatic Application ID	<p>Specifies an application ID that is automatically generated by the SequeLink ADO Client to identify the client application to the SequeLink service. This attribute is only required when the SequeLink service you are connecting to has been configured to limit access to specific applications.</p> <p>For more information about using application IDs to limit access to SequeLink services, see <a href="#">“Specifying Application IDs” on page 169</a>.</p>
Database	<p>Specifies the name of the database to which you want to connect.</p>

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

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



---

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

---

Attribute	Description
Use LDAP	<p>UseLDAP={0   1}. Determines whether the parameters to establish a connection to the SequeLink Server should be retrieved from LDAP.</p> <p>When set to 0 (the initial default), the SequeLink Client will connect directly to the specified SequeLink Server.</p> <p>When set to 1, the SequeLink Client will retrieve the TCP/IP host, TCP/IP port, and SequeLink data source (optional) from an LDAP entry identified by a Distinguished Name (DN). Once the connection information is retrieved, the SequeLink Client will connect directly to the specified SequeLink Server. The DistinguishedName (DN) attribute is required.</p>
User ID	<p>Specifies the host or data store user name, which may be required depending on the server configuration.</p>

---



## 4 Developing ADO Applications

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

- [“OLE DB Objects and Interfaces” on page 132](#)
- [“Supported Schema Rowsets” on page 134](#)
- [“Supported OLE DB Property Groups” on page 135](#)
- [“OLE DB Interfaces Supported in ADO” on page 148](#)
- [“Mapping ADO Methods and Properties” on page 150](#)
- [“Data Shaping” on page 166](#)
- [“Persisting Information” on page 167](#)
- [“Using Rowsets” on page 167](#)
- [“Unicode Support” on page 168](#)
- [“Mapping Data Types” on page 168](#)
- [“Specifying Application IDs” on page 169](#)
- [“Error Handling” on page 170](#)

---

# OLE DB Objects and Interfaces

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

---

**Table 4-1. Objects and Interfaces Supported by the SequeLink 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	
Enumerator	IDBInitialize	
	IDBProperties	
	IParseDisplayName	
	ISourcesRowset	
	ISupportErrorInfo	

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

OLE DB Object	Interface	
MultipleResults	IMultipleResults	
	ISupportErrorInfo	
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 SequeLink ADO provider.

---

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

---

CATALOGS <sup>(1), (2)</sup>	PROCEDURE_PARAMETERS
COLUMN_PRIVILEGES <sup>(1), (2)</sup>	PROCEDURES
COLUMNS	PROVIDER_TYPES
FOREIGN_KEYS	SCHEMATA <sup>(1), (2)</sup>
INDEXES	STATISTICS
PRIMARY_KEYS	TABLE_PRIVILEGES <sup>(1), (2)</sup>
PROCEDURE_COLUMNS <sup>(1), (2)</sup>	TABLES

---

NOTES:

<sup>1</sup> Not supported for DB2

<sup>2</sup> 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 SequeLink 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 SequeLink 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 SequeLink 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 SequeLink 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 SequeLink 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_ASYNCCTXNABORT	VALUE=VARIANT_FALSE. The data provider cannot abort transactions asynchronously.
DBPROP_ASYNCCTXNCOMMIT	VALUE=VARIANT_FALSE. The data provider cannot commit transactions asynchronously.
DBPROP_BYREFACCESSORS	VALUE=VARIANT_FALSE. The data provider does not support the DBACCESSOR_PASSBYREF flag.
DBPROP_CATALOGLOCATION	The value depends on the SequeLink service you are using.
DBPROP_CATALOGTERM	Specifies the name the data source uses for a catalog. VALUE="Database"
DBPROP_CATALOGUSAGE	A combination of zero or one or more of the following: VALUE=DBPROPVAL_CU_DML_STATEMENTS. Catalog names are supported in all Data Manipulation Language (DML) statements. VALUE=DBPROPVAL_CU_TABLE_DEFINITION. Catalog names are supported in all table definition statements. VALUE=DBPROPVAL_CU_INDEX_DEFINITION. Catalog names are supported in all index definition statements. VALUE= DBPROPVAL_CU_PRIVILEGE_DEFINITION. Catalog names are supported in all privilege definition statements.
DBPROP_COLUMNDEFINITION	VALUE=DBPROPVAL_CD_NOTNULL. Columns can be created non-nullable.

**Table 4-5. OLE DB Data Source Information Properties Supported by the SequeLink 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. For more information about configuring ADO data sources, see <a href="#">Chapter 3 “Using the SequeLink ADO Client”</a> on page 101.
DBPROP_DATASOURCEREADONLY	VALUE=VARIANT_FALSE. The data source can be updated.
DBPROP_DBMSNAME	Specifies the name of the data store accessed by the SequeLink 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 SequeLink 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 SequeLink 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 multipleparameter 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 SequeLink 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 ADO Provider.
DBPROP_PROVIDERNAME	VALUE=slslknn.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 SequeLink 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 SequeLink 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 SequeLink 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. For more information about connection attributes, see <a href="#">“ADO Connection Attributes” on page 126</a> .
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. For more information about connection attributes, see <a href="#">“ADO Connection Attributes” on page 126</a> .
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 SequeLink ADO Provider** *(cont.)*

Property Name	Default Value and Description
DBPROP_INIT_PROVIDERSTRING	<p>Specifies a provider-specific string that contains extra initialization information. For information about using the provider string, see <a href="#">“Connecting with a Provider String” on page 125</a>.</p> <p>Consumers should use this property only for provider-specific connection information as described in <a href="#">Table 3-2 on page 126</a>.</p>

## 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 SequeLink 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 SequeLink ADO Provider” on page 144](#) provides the properties that are included in the SequeLink 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 SequeLink 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_CANSROLLBACKWARDS	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 SequeLink 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 SequeLink 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_OTHERUPDELETEDELETE	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_OWNUPDELETEDELETE	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 SequeLink 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	<p>A combination of zero or one or more of the following:</p> <p>Value=DBPROPVAL_UP_CHANGE. SetData is supported.</p> <p>Value=DBPROPVAL_UP_DELETE. DeleteRows is supported.</p> <p>Value=DBPROPVAL_UP_INSERT. InsertRow is supported.</p>

---

## Session Property Group

[Table 4-8](#) lists the properties the SequeLink 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 SequeLink 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 SequeLink ADO provider supports additional OLE DB interfaces that are not used by ADO.

For a description of the OLE DB objects and interfaces supported by the SequeLink ADO provider, see [“OLE DB Objects and Interfaces” on page 132](#).

---

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

---

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

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

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

---

# Mapping ADO Methods and Properties

This section maps the methods, properties, and collections of ADO objects to the OLE DB methods supported by the SequeLink 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 SequeLink 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	<p>Specifies whether access rights are restricted on a column-by-column basis.</p> <p>VALUE=True. Access rights are restricted on a column-by-column basis.</p> <p>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.</p>
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	<p>VALUE=False. The consumer must call IRowsetIdentity::IsSameRow to determine whether two row handles point to the same row.</p>
Lock Mode	VALUE=1. The provider is not required to lock rows at any time to ensure successful updates.
Maximum Open Rows	<p>Specifies the maximum number of rows that can be active at the same time.</p> <p>VALUE=4096</p>
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 SequeLink 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 <i>or</i> 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	<p>Specifies how catalog names can be used in text commands. A combination of zero or one or more of the following:</p> <p>VALUE=1. Catalog names are supported in all Data Manipulation Language statements.</p> <p>VALUE=2. Catalog names are supported in all table definition statements.</p> <p>VALUE=4. Catalog names are supported in all index definition statements.</p> <p>VALUE=8. Catalog names are supported in all privilege definition statements.</p>
Column Definition	VALUE=1. Columns can be created non-nullable.
COM Object Support	VALUE=1. The data providers support access to BLOBs as structured storage objects. A data consumer determines which interfaces are supported through the Structured Storage property.
Connection Status	VALUE=1. The data source is in an initialized state and able to communicate with the data store.
Current Catalog	Specifies the name of the current catalog. The data consumer can use the CATALOGS rowset to enumerate catalogs. If not set, the data provider uses the default catalog.
Data Source	Specifies the name of the data source or enumerator to which to connect.
Data Source Name	Specifies the name of the data source (server) used during the connection process.
Data Source Object Threading Model	VALUE=1. The data provider uses the free-threading model.
DBMS Name	Specifies the name of the product accessed by the SequeLink 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	<p>A provider-specific string that contains extra initialization information. Consumers should use this property only for provider-specific connection information.</p> <p>For information on using the provider string with the SequeLink ADO provider, see <a href="#">“Connecting with a Provider String” on page 125</a>.</p>
GROUP BY Support	<p>This value depends on the SequeLink service you are using.</p> <p>VALUE=1. The GROUP BY clause is not supported.</p> <p>VALUE=2. The GROUP BY clause must contain all nonaggregated columns in the select list. It cannot contain any other columns.</p> <p>VALUE=4. The GROUP BY clause must contain all nonaggregated columns in the select list. It can contain columns that are not in the select list.</p> <p>VALUE=8. The columns in the GROUP BY clause and the select list are not related. The meaning of nongrouped, nonaggregated columns in the select list is data source-dependent.</p>
Heterogeneous Table Support	<p>VALUE=0. The provider cannot join tables from different catalogs or providers.</p>
Identifier Case Sensitivity	<p>This value depends on the SequeLink service you are using.</p> <p>VALUE=1. Identifiers in SQL are case-sensitive and are stored in upper case in the system catalog.</p> <p>VALUE=2. Identifiers in SQL are case-insensitive and are stored in lower case in the system catalog.</p> <p>VALUE=4. Identifiers in SQL are case-sensitive and are stored in mixed case in the system catalog.</p> <p>VALUE=8. Identifiers in SQL are case-insensitive and are stored in mixed case in the system catalog.</p>
Initial Catalog	<p>Specifies the name of the initial, or default, catalog to use when connecting to the data source.</p>

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

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

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

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

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

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



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

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

## Recordset Object

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

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

<b>ADO Property</b>	<b>Default Value and Description</b>
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 SequeLink 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 as `Data Provider=DataDirect SequeLink 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 SequeLink ADO provider using the SequeLink ADO data source named *HR*:

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

---

## Persisting Information

A data source object can be persisted (saved). The SequeLink 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 SequeLink 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.

For information on the schema rowsets supported, see the ["Supported Schema Rowsets" on page 134](#).

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

---

## Unicode Support

The SequeLink ADO provider supports Unicode.

When the string is data, the SequeLink 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 SequeLink ADO provider performs the appropriate conversion.

When the string is not data, the SequeLink 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 317](#) 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 SequeLink 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 SequeLink 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 ADO Client:

- SequeLink ADO provider errors
- SequeLink Client errors
- SequeLink Server errors
- Database errors

### SequeLink ADO Provider Errors

An error generated by the SequeLink ADO 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 ADO Client has the following format:

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

For example:

```
[DataDirect] [SequeLink ADO provider] [SequeLink Client]
Memory allocation error occurred.
```

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

## SequeLink Server Errors

An error generated by SequeLink Server has the following format:

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

For example:

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

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

## Database Errors

An error generated by the database has the following format:

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

For example:

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

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

# Part 3: Developing JDBC Applications

This part contains the following chapters:

- [Chapter 5 “Using JDBCTest” on page 175](#) introduces JDBCTest, 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 6 “Using the SequeLink Java Client” on page 227](#) provides information about using JDBC applications with the SequeLink Java Client.
- [Chapter 7 “Tracking JDBC Calls” on page 259](#) introduces Spy, a tool that allows you to track JDBC calls, and describes how to use it.
- [Chapter 8 “Developing JDBC Applications” on page 267](#) provides information about developing JDBC applications for SequeLink environments.



## 5 Using JDBCTest

This chapter provides information about JDBCTest, 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.

JDBCTest 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

---

### JDBCTest Tutorial

This JDBCTest tutorial explains how to use the most important features of JDBCTest (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 JDBCTest

The default JDBCTest configuration file is:

*install\_dir*/jdbc/test/Config.txt

where *install\_dir* is your SequeLink 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 JDBCTest

How you start JDBCTest depends on your platform:

- **As a Java application on Windows**—Run the `jdbctest.bat` file located in the `jdbctest` directory.

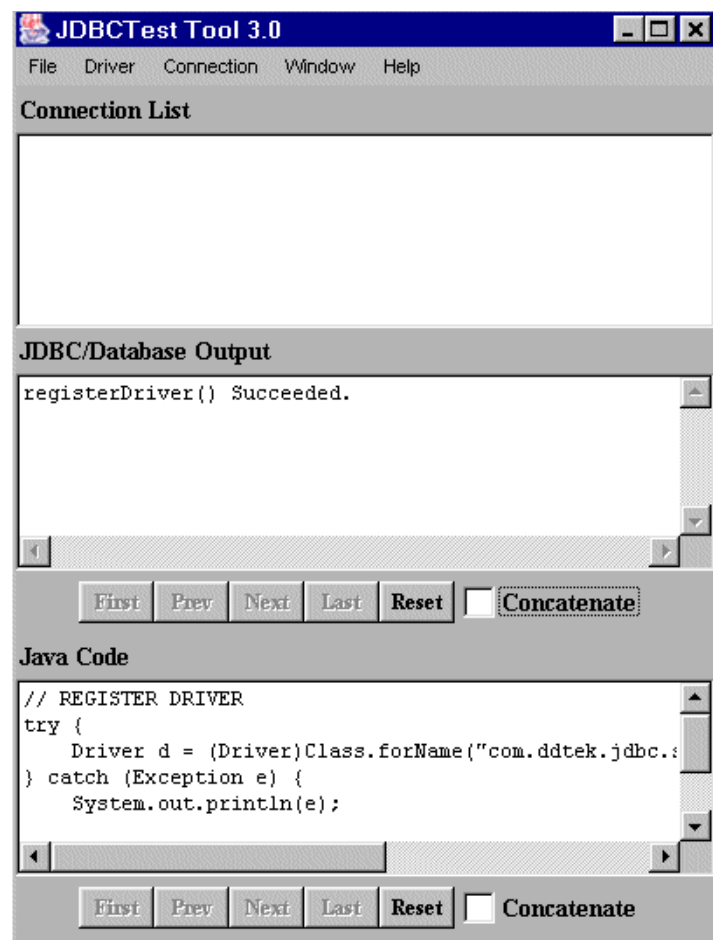
Under Windows 9x and Me, double-clicking `jdbctest.bat` opens a DOS window and displays the error "Out of environment space." To prevent this, use the following procedure:

- a After installing SequeLink JDBC Driver, locate the `jdbctest.bat` file in the `jdbctest` directory beneath the installation directory.
- b Right-click `jdbctest.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 `jdbctest` shortcut is created in the same directory with `jdbctest.bat`.
- d Double-click either `jdbctest.bat` or the `jdbctest` shortcut. JDBCTest will open normally without producing the error.

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

- **As a Java application on UNIX**—Run the `jdbctest.sh` shell script located in the `jdbctest` directory beneath the installation directory.

After you start JDBCTest, the following window appears:



The main JDBCTest 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.

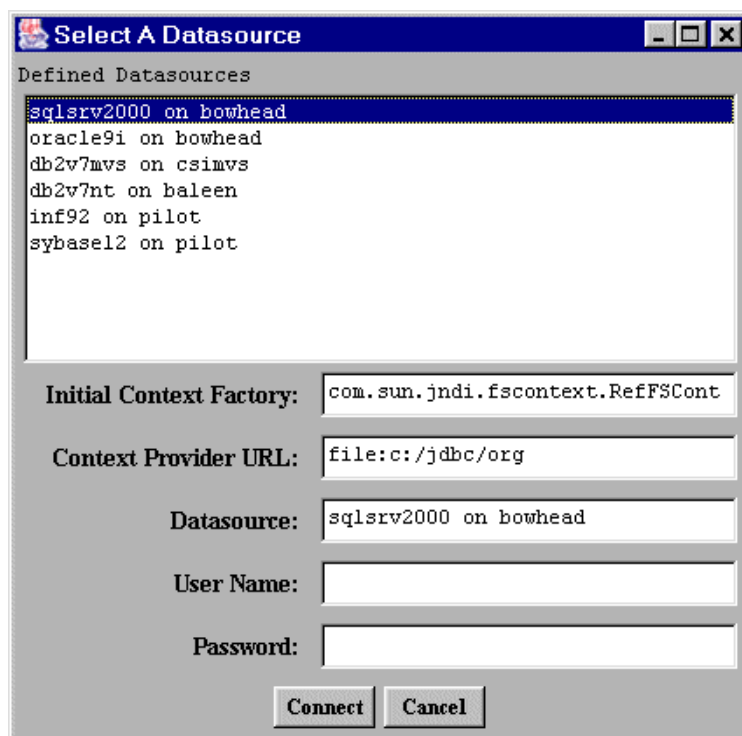
TIP: The JDBCTest windows contain two Concatenate check boxes. Select a Concatenate check box to see a cumulative record of previous actions; otherwise, only the last action is shown. Selecting Concatenate can degrade performance, particularly when displaying large resultSets.

## Connecting Using JDBCTest

There are two methods to connect using JDBCTest: through a data source or through driver/database selection.

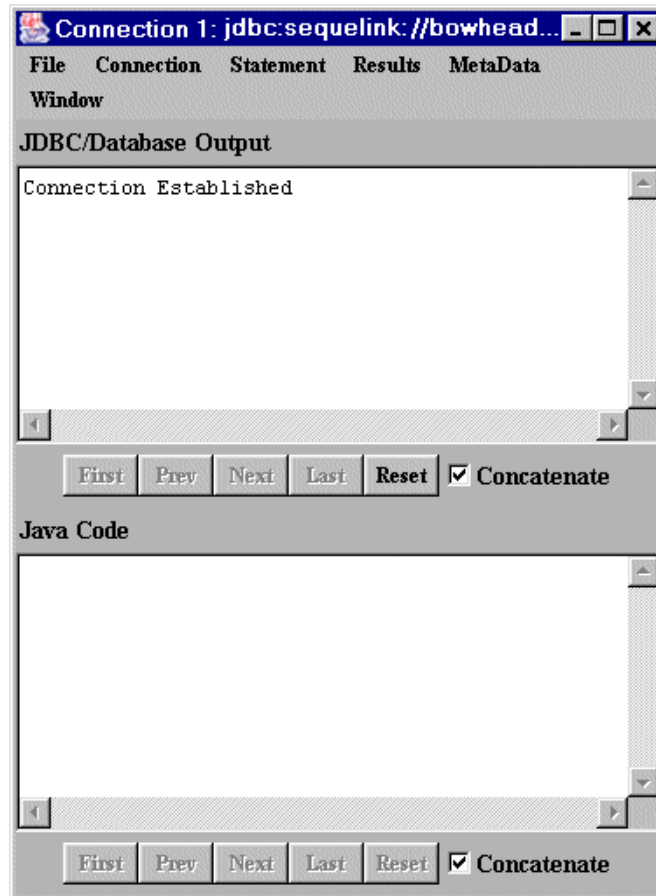
### *Connecting through a Data Source*

- 1 From the JDBCTest main window menu, select **Connection / Connect to DB via Data Source**. JDBCTest 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**. For information about JDBC connection properties, see the [Chapter 8, "Developing JDBC Applications,"](#) on page 267.

- 3 If the connection was successful, the Connection window appears and shows the Connection Established message in the JDBC/Database Output scroll box.

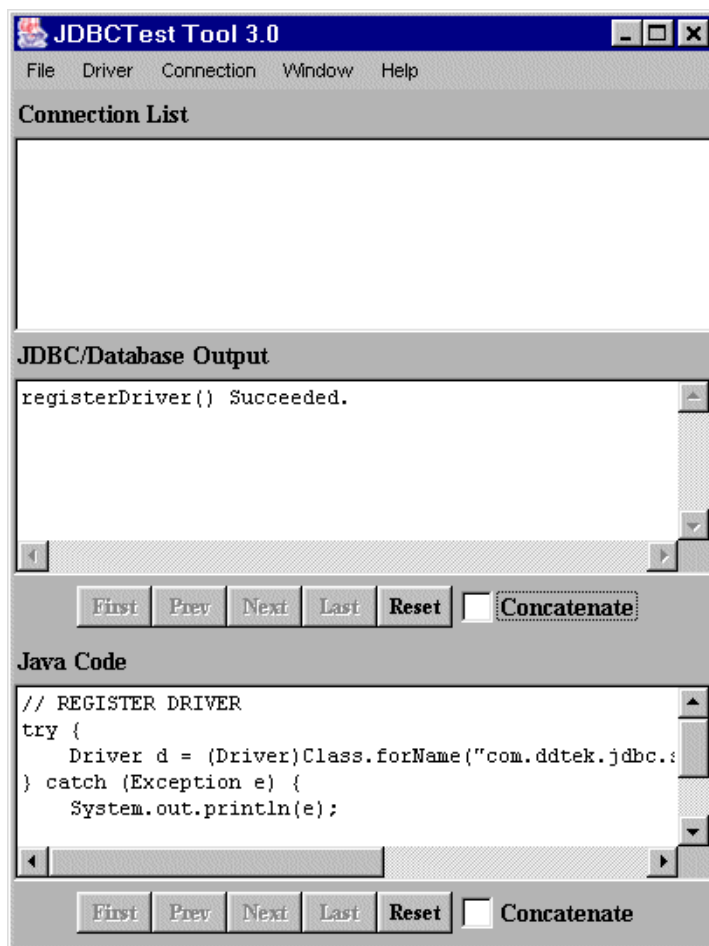


## Connecting through Driver/Database Selection

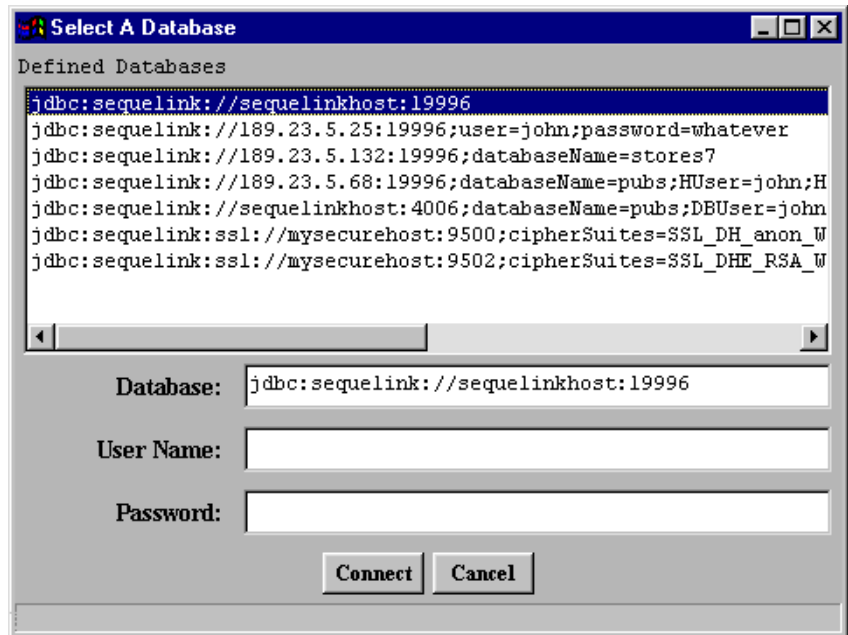
- 1 From the JDBCTest main window menu, select **Driver / Register Driver**. JDBCTest prompts you for a JDBC driver name.
- 2 In the Please Supply a Driver URL field, make sure that a driver is specified, as in the following example; then, click **OK**.

```
com.ddtek.jdbc.sequellink.SequeLinkDriver
```

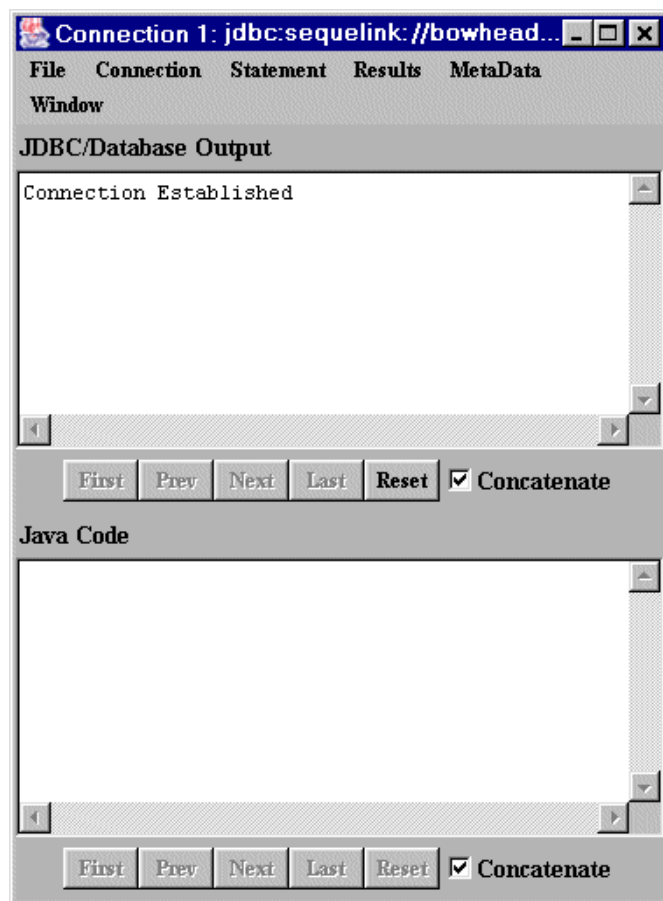
If the SequeLink JDBC Driver was registered successfully, the main JDBCTest window appears with a confirmation in the JDBC/Database Output scroll box.



- 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 SequeLink 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**. For information about JDBC connection properties, see [Chapter 8, "Developing JDBC Applications," on page 267](#).
- 6 If the connection was successful, the Connection window appears and shows the Connection Established message in the JDBC/Database Output scroll box.



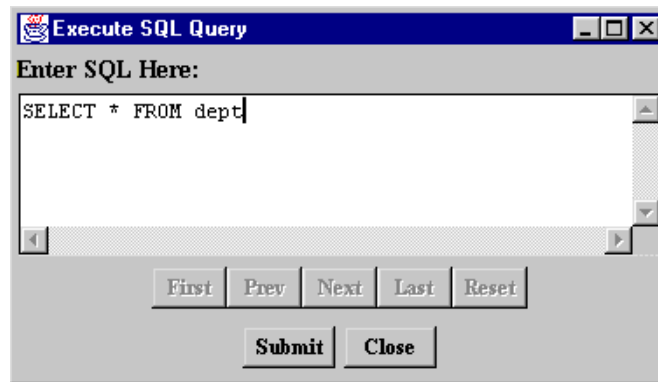
## Executing a Simple Select Statement

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**. JDBCTest displays a dialog box that prompts for a SQL statement.

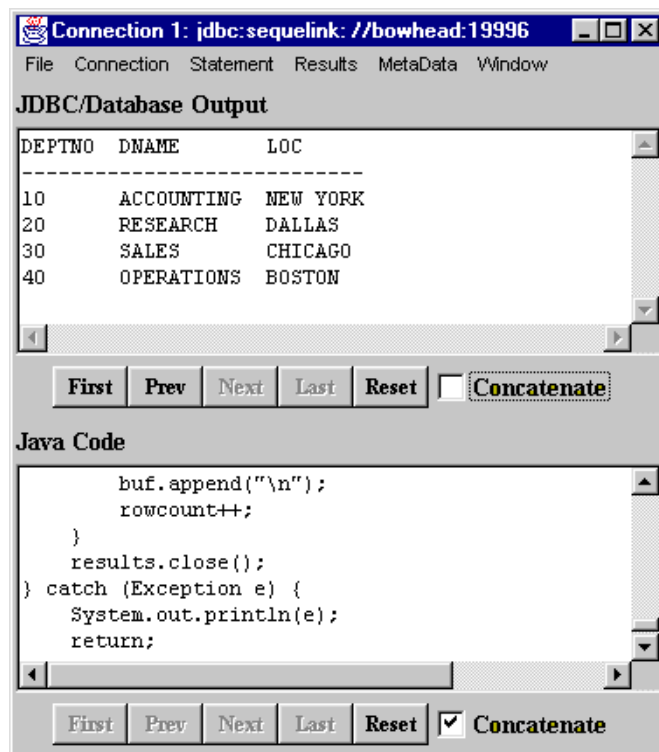


- Specify the Select statement that you want to execute.



Click **Submit**; then, click **Close**.

- Select **Results / Show All Results**. The data from your result set is displayed.

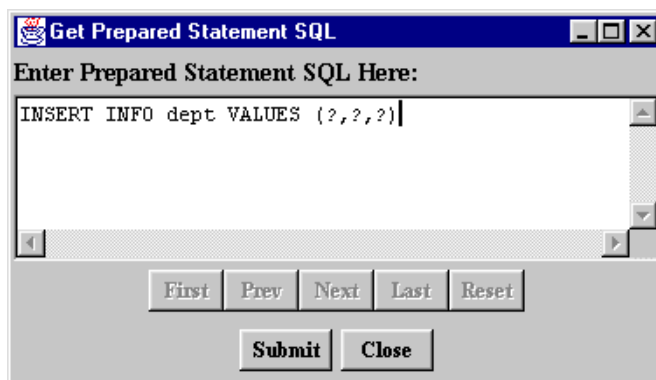


- 5 Scroll through the code in the Java Code scroll box to see which JDBC calls have been implemented by JDBCTest.

## 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**. JDBCTest prompts you for a SQL statement.
- 2 Specify the Insert statement that you want to execute.



Click **Submit**; then, click **Close**.

- 3 Select **Statement / Set Prepared Parameters**. To set the value and type for each parameter:
  - a Type the parameter number.
  - b Select the parameter type.
  - c Type the parameter value.
  - d Click **Set** to pass this information to the JDBC driver.

#	Type	Value
1	short	50
2	String	DEVELOPMENT
3	String	SAN FRANCISCO

Parameter #:

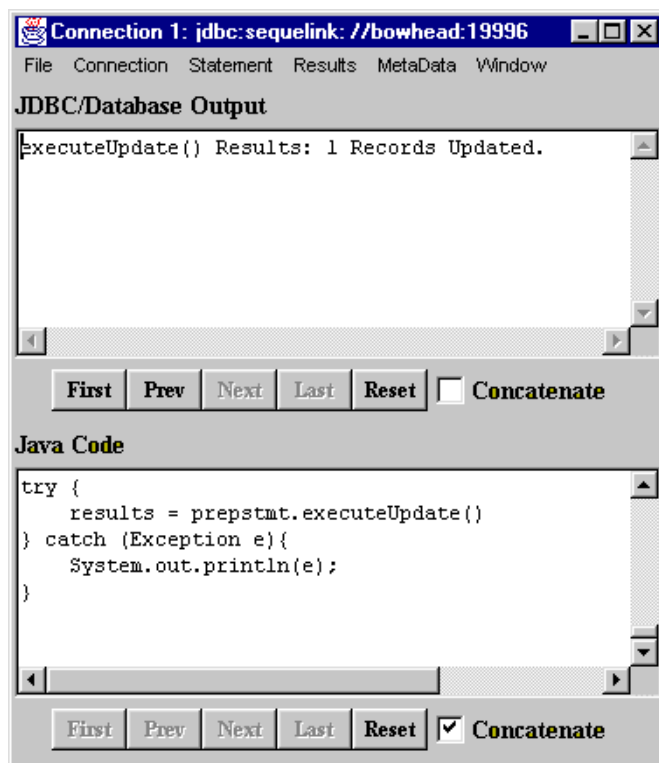
Parameter Type:

Use Calendar: ☐ Zone:

Parameter Value:

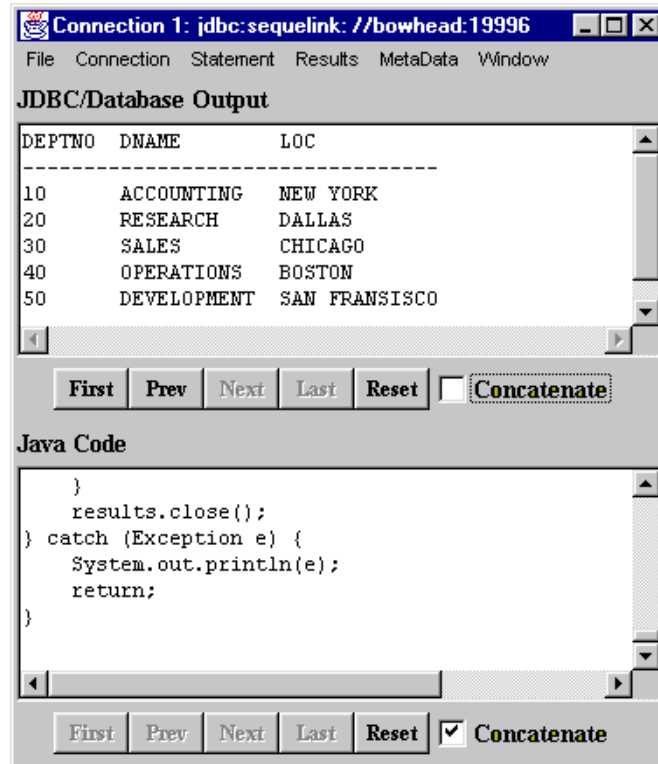
- 4 When you are finished, click **Close**.

- 5 Select **Statement / Execute Stmt Update**. The JDBC/Database Output scroll box indicates that one row has been inserted.



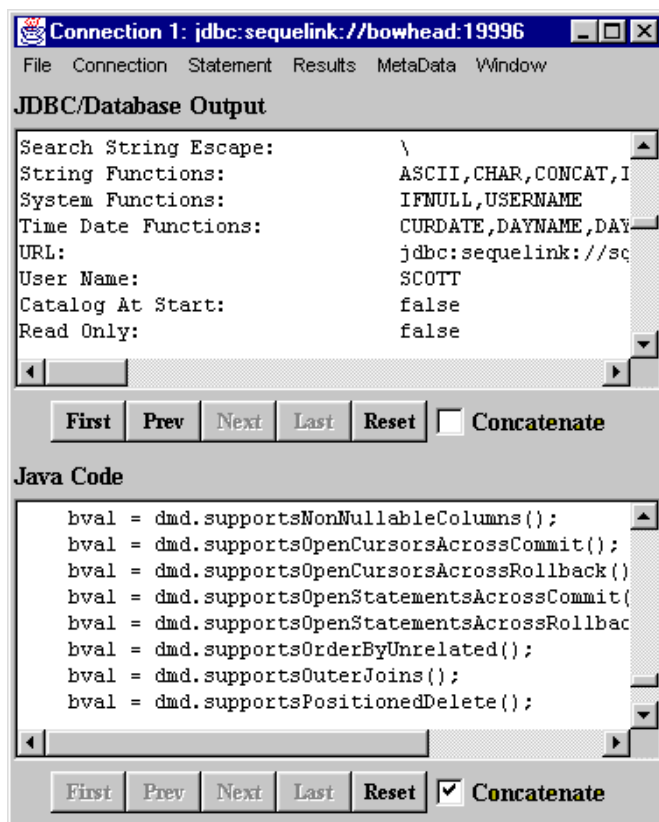
- 6 If you want to insert multiple records, repeat [Step 3](#) and [Step 5](#) for each record.

- 7 If you repeat the steps described in [“Executing a Simple Select Statement” on page 184](#), you will see that the previously inserted records are also returned.



## Retrieving Database Metadata

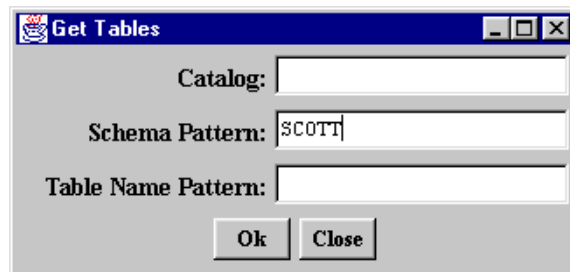
- 1 From the Connection window menu, select **Connection / Get DB Meta Data**.
- 2 Select **MetaData / Show Meta Data**. Information about the JDBC driver and the database to which you are connected is returned.



- 3 Scroll through the Java code in the Java Code scroll box to find out which JDBC calls have been implemented by JDBCTest.

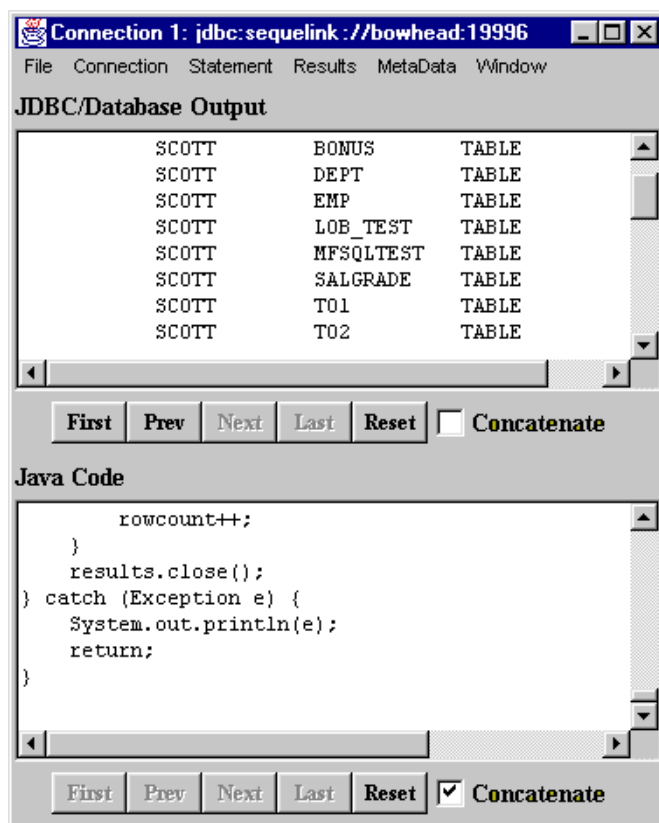
The metadata also allows you to query the database catalog (enumerate the tables in the database, for example). In this example, we will query all tables that are owned by the user SCOTT.

- 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**. JDBCTest prompts for a result set type and concurrency.

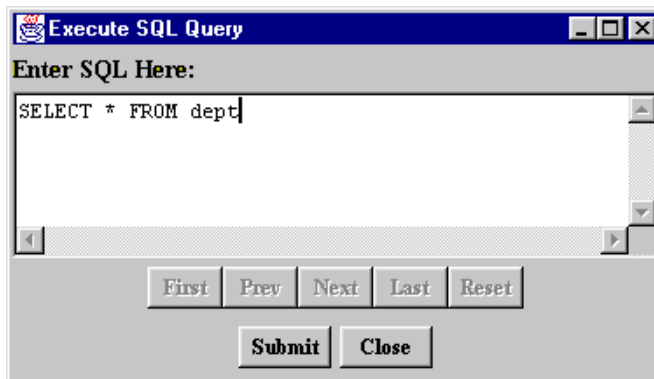


- 2 In the resultSetType field, select **TYPE\_SCROLL\_SENSITIVE**. In the resultSetConcurrency field, select **CONCUR\_READ\_ONLY**.



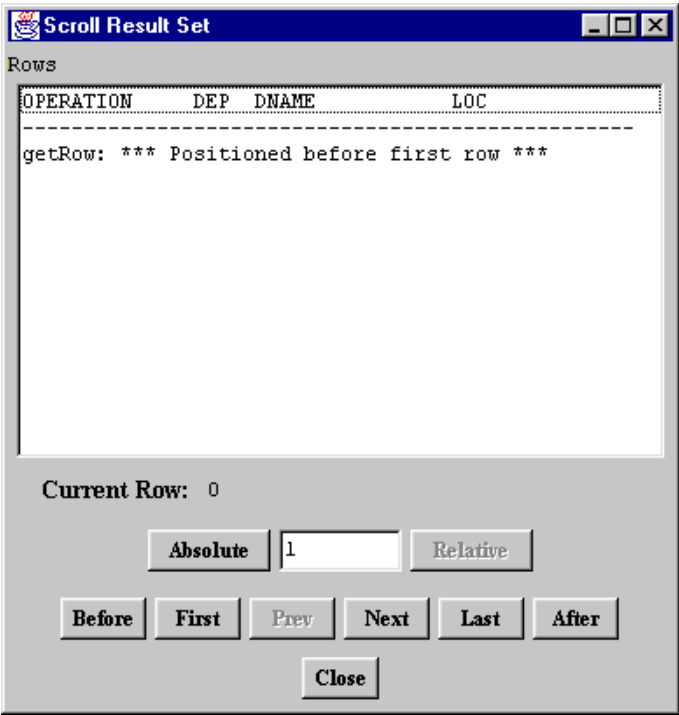
Click **Submit**; then, click **Close**.

- 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.

The screenshot shows a window titled "Scroll Result Set" with a table of data and navigation controls. The table has four columns: OPERATION, DEP, DNAME, and LOC. The data is as follows:

OPERATION	DEP	DNAME	LOC
next:1	10	ACCOUNTING	NEW YORK
next:2	20	RESEARCH	DALLAS
first:1	10	ACCOUNTING	NEW YORK
last:5	50	DEVELOPMENT	SAN FRANCISCO
previous:5	50	DEVELOPMENT	SAN FRANCISCO
previous:4	40	OPERATIONS	BOSTON
previous:3	30	SALES	CHICAGO

Below the table, the "Current Row" is displayed as 3. At the bottom, there are buttons for "Absolute", "Relative", "Before", "First", "Prev", "Next", "Last", "After", and "Close". The "Prev" button is currently selected.

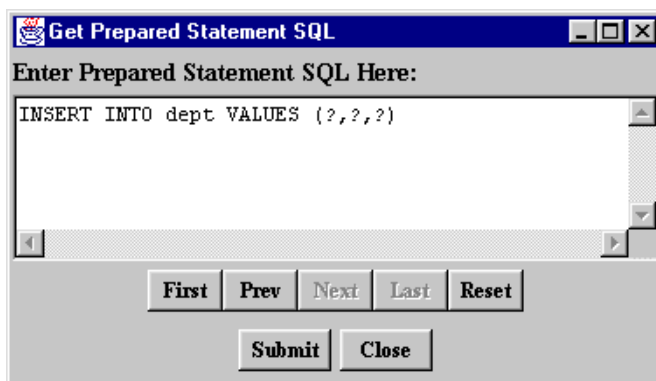
- 7 Click Close.

## Batch Execution on a Prepared Statement

Batch execution on a prepared statement allows you to update or insert multiple records simultaneously. In some cases, this can significantly improve system performance because fewer round-trips to the database are required.

NOTE: Batch execution on a prepared statement is supported by the JDBC 2.0 and higher specifications and requires a Java 2 Platform (JDK 1.2 or higher)-compatible Java Virtual Machine.

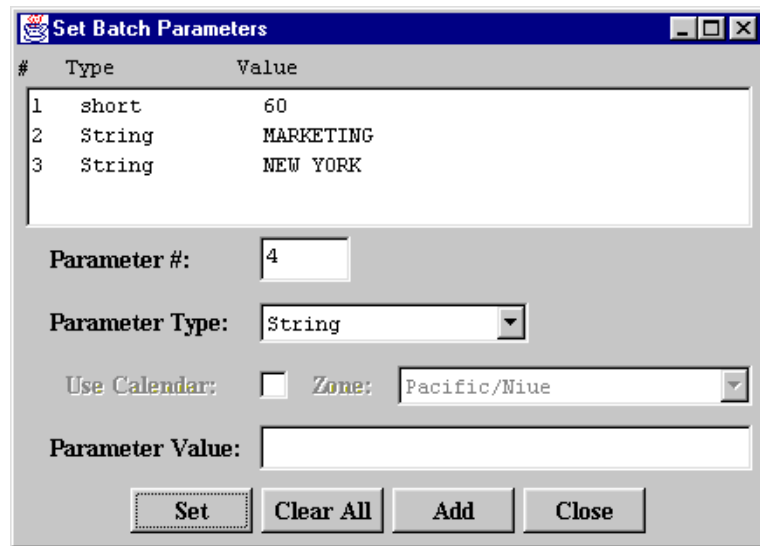
- 1 From the Connection window menu, select **Connection / Create Prepared Statement**.
- 2 Specify the Insert statement that you want to execute.



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**.



The dialog box titled "Set Batch Parameters" contains a table with three columns: #, Type, and Value. The table has three rows: 1 short 60, 2 String MARKETING, and 3 String NEW YORK. Below the table are input fields for Parameter # (4), Parameter Type (String), Use Calendar (unchecked), Zone (Pacific/Niue), and Parameter Value. At the bottom are buttons for Set, Clear All, Add, and Close.

#	Type	Value
1	short	60
2	String	MARKETING
3	String	NEW YORK

Parameter #:

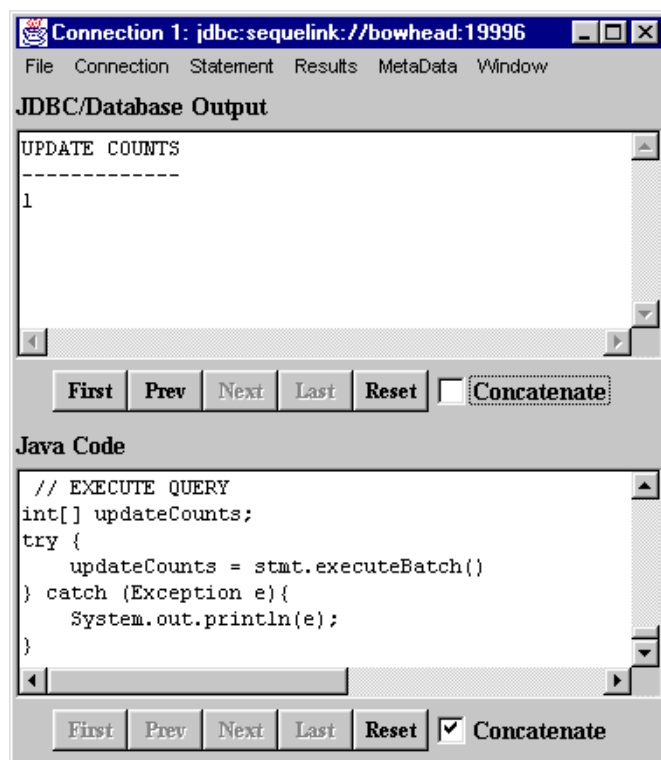
Parameter Type:

Use Calendar: ☐ Zone:

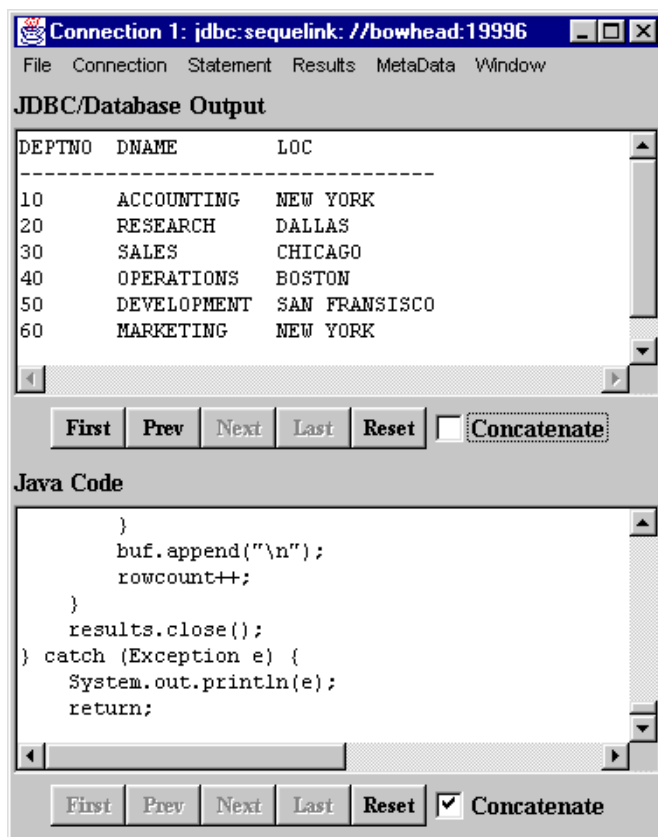
Parameter Value:

- 5 Click **Add** to add the specified set of parameters to the batch. To add multiple parameter sets to the batch, repeat [Step 3](#) through Step 5 as many times as necessary. When you are finished adding parameter sets to the batch, click **Close**.

- 6 Select **Statement / Execute Stmt Batch**. JDBCTest displays the rowcount for each of the elements in the batch.



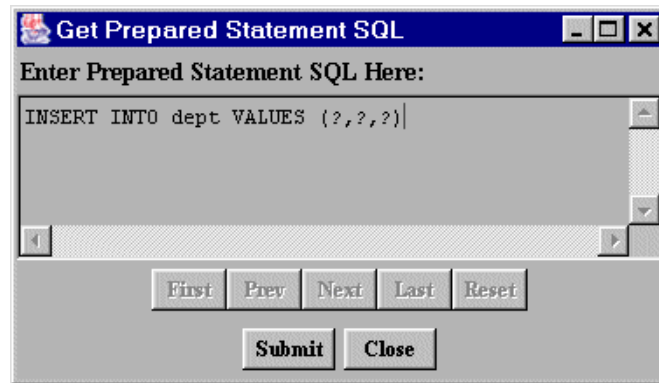
- 7 If you re-execute the Select statement from “[Executing a Simple Select Statement](#)” on page 184, you see that the previously inserted records are returned.



## Returning ParameterMetaData

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

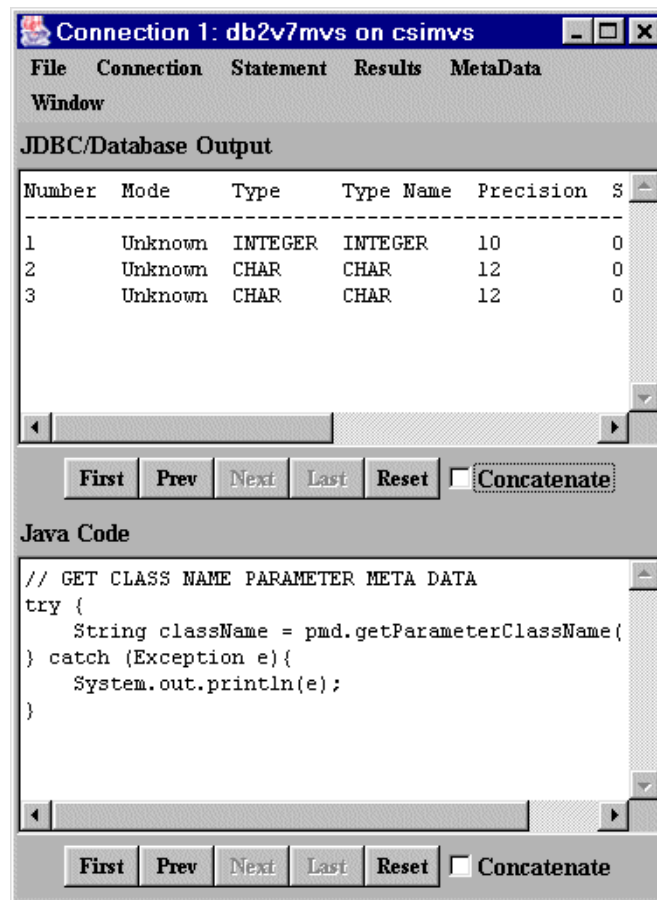
- 1 From the Connection window menu, select **Connection / Create Prepared Statement**.
- 2 Specify the prepared statement that you want to execute.



Click **Submit**; then, click **Close**.



- 3 Select **Statement / Get ParameterMetaData**. The Connection window displays ParameterMetaData:

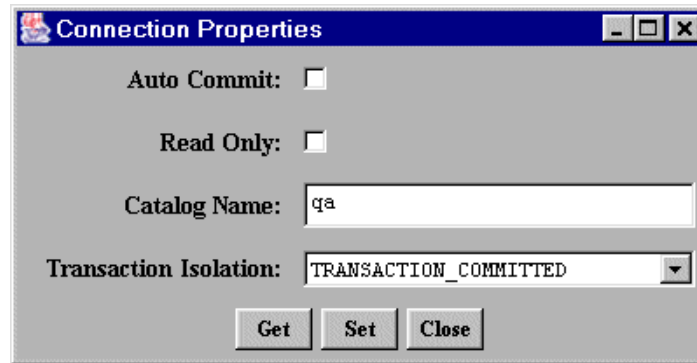


## Establishing Savepoints

NOTE: Savepoints is a JDBC 3.0 feature and requires a JDK 1.4 Java Virtual Machine.

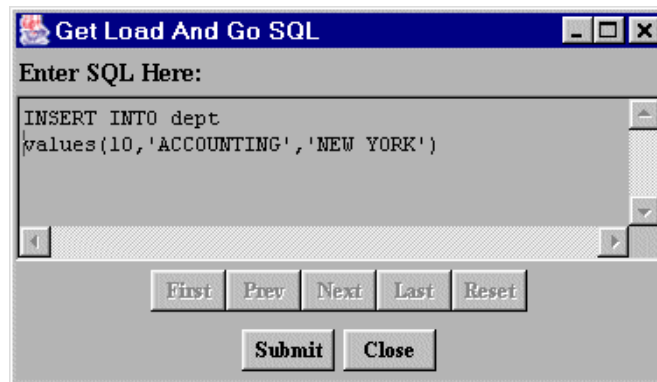
- 1 From the Connection window menu, select **Connection / Connection Properties**.

- 2 Select TRANSACTION\_COMMITTED from the Transaction Isolation drop-down list. Do not select the Auto Commit check box.



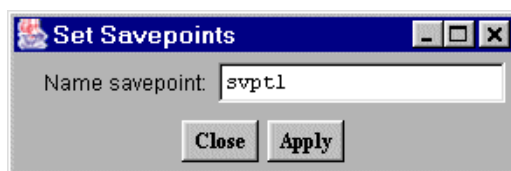
Click **Set**; then, click **Close**.

- 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.

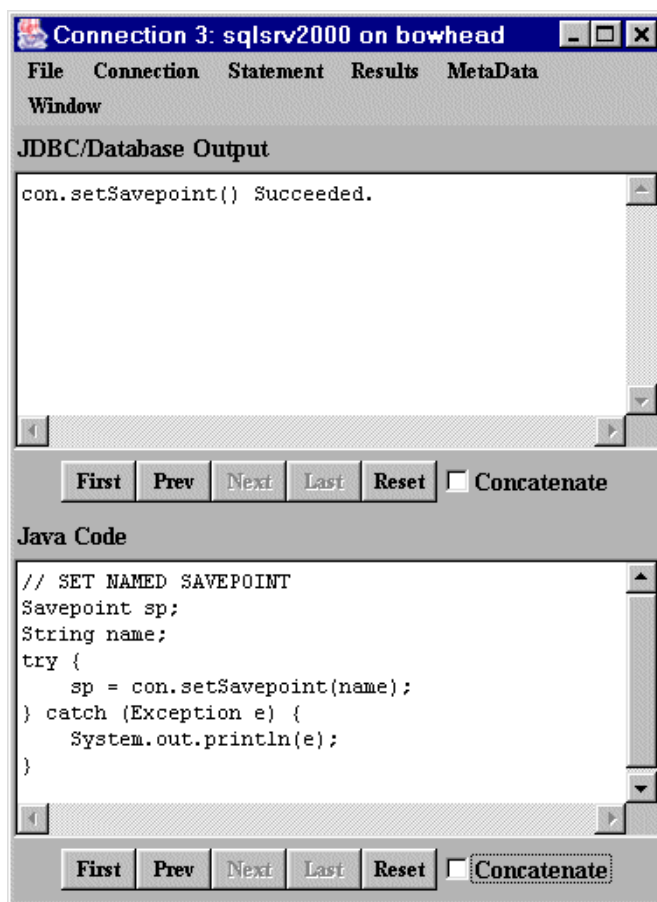


Click **Submit**.

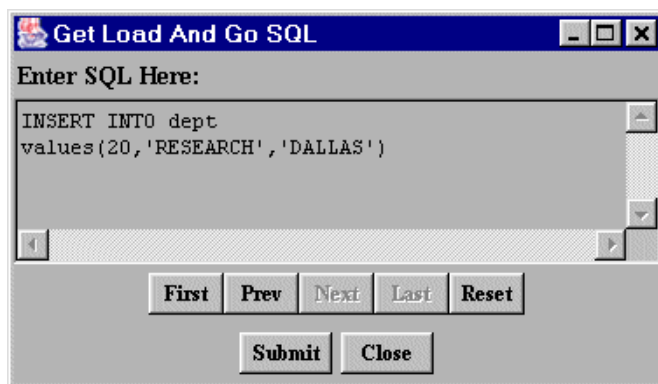
- 5 Select **Connection / Set Savepoint**. In the Set Savepoints window, specify a savepoint name.



Click **Apply**; then, click **Close**. The Connection window indicates whether or not the savepoint succeeded.

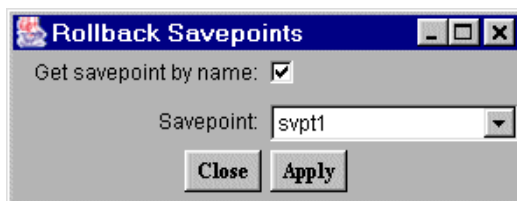


- 6 Return to the Get Load and Go SQL window and specify another statement.

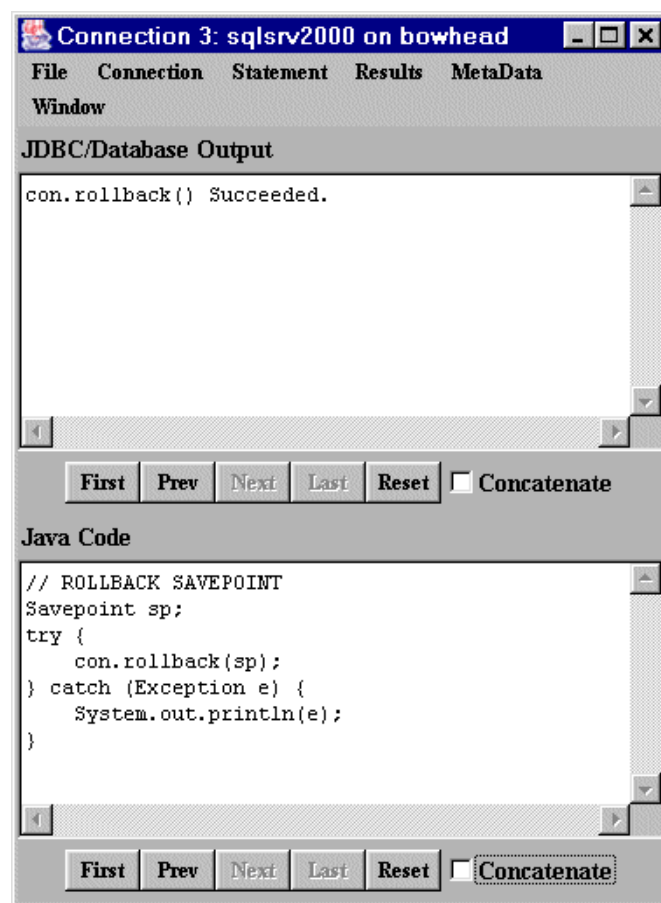


Click **Submit**.

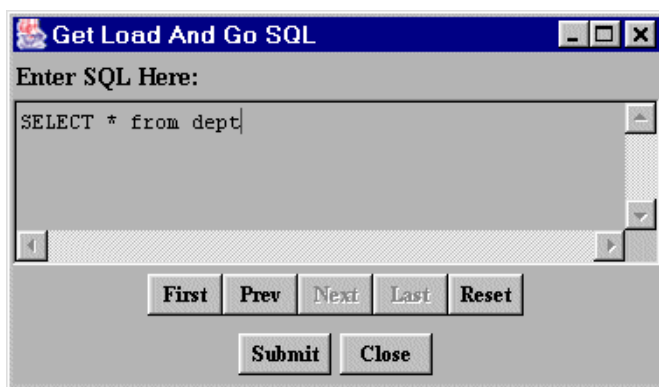
- 7 Select **Connection / Rollback Savepoint**. In the Rollback Savepoints window, specify the savepoint name.



Click **Apply**; then, click **Close**. The Connection window indicates whether or not the savepoint rollback succeeded.

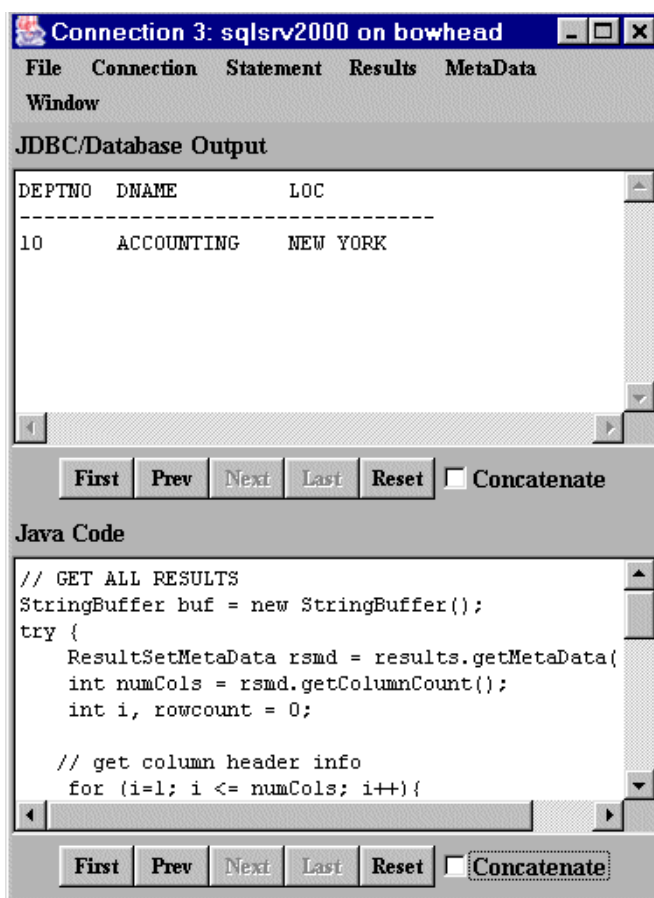


- 8 Return to the Get Load and Go SQL window and specify another statement.



Click **Submit**; then, click **Close**.

The Connection window displays data that was inserted before the first Savepoint. The second insert was rolled back.

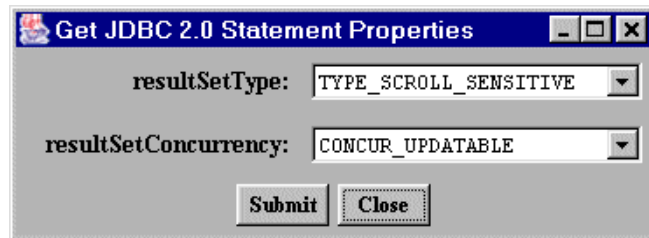


## Updatable Result Sets

The following examples illustrate Updatable result sets by deleting, inserting, and updating a row.

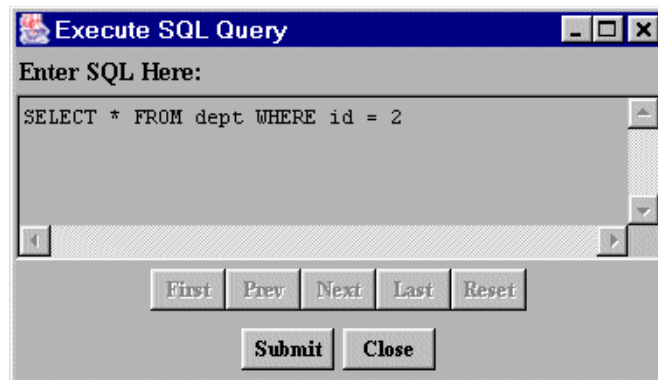
### *Deleting a Row*

- 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.

Col#	Name	Type
1	DEPTNO	int
2	DNAME	char
3	LOC	char
4	id	numeric identity

Current Row: 1

Retrieve By Column: Index

Data Type: String Use Scale: 0

Use Calendar: ☐ Zone: Etc/GMT+12

Get Cell Value:

Set Cell Value:

Auto Traverse: ☒

Absolute 1 Relative

Before First Prev Next Last After

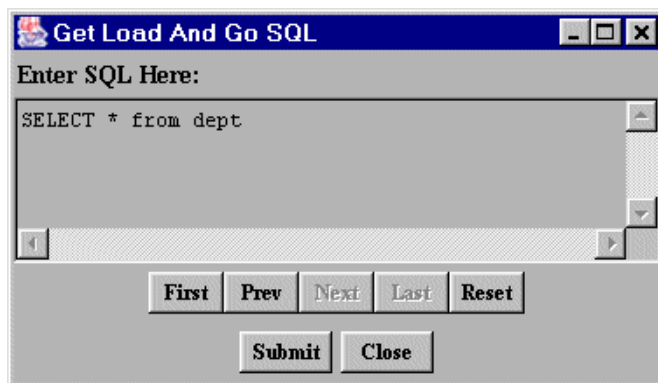
Get Cell Set Cell

Update Row Delete Row Insert Row Move to insert row Move to current row

Close

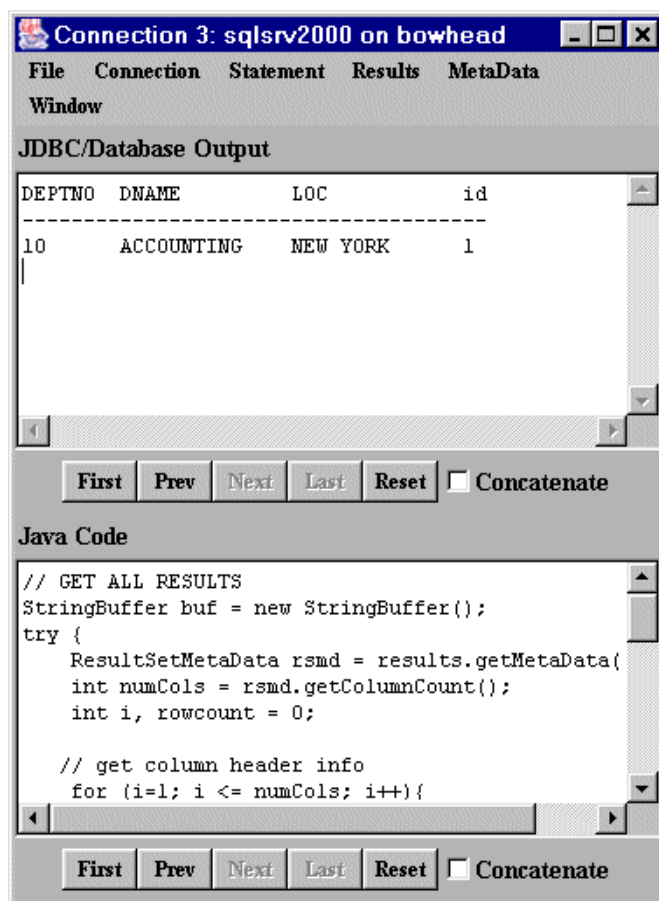
- 6 Click **Next**. Current Row changes to 1.
- 7 Click **Delete Row**.

- 8 To verify the result, return to the Connection menu and select **Connection / Load and Go**. The Get Load and Go SQL window appears.
- 9 Specify the statement that you want to execute.



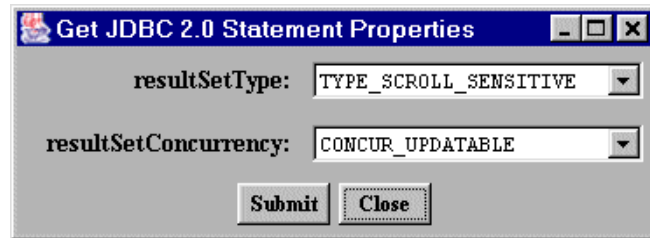
Click **Submit**; then, click **Close**.

10 The Connection window shows one row returned.



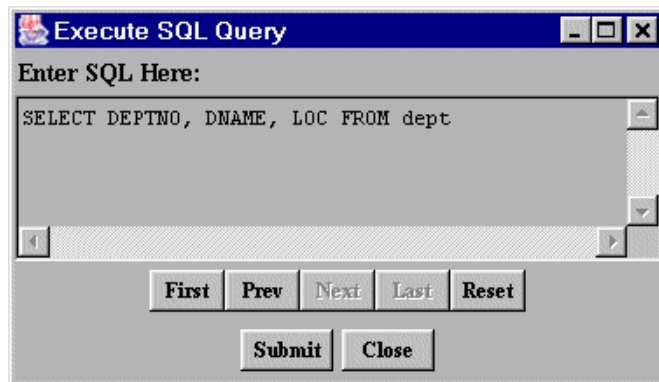
## *Inserting a Row*

- 1 From the Connection window menu, select **Connection / Create JDBC 2.0 Statement**.
- 2 In the resultSetType field, select **TYPE\_SCROLL\_SENSITIVE**. In the resultSetConcurrency field, select **CONCUR\_UPDATABLE**.



Click **Submit**; then, click **Close**.

- 3 Select **Statement / Execute Stmt Query**.
- 4 Specify the Select statement that you want to execute.



Click **Submit**; then, click **Close**.

- 5 Select **Results / Inspect Results**. The Inspect Result Set window is displayed.

**Inspect Result Set**

Col#	Name	Type
1	DEPTNO	int
2	DNAME	char
3	LOC	char

Current Row: Insert row

Retrieve By Column: Index

Data Type: String Use Scale: ☐ 0

Use Calendar: ☐ Zone: Etc/GMT+12

Get Cell Value: [Text Box]

Set Cell Value: DALLAS

Auto Traverse: ☒

Absolute 1 Relative

Before First Prev Next Last After

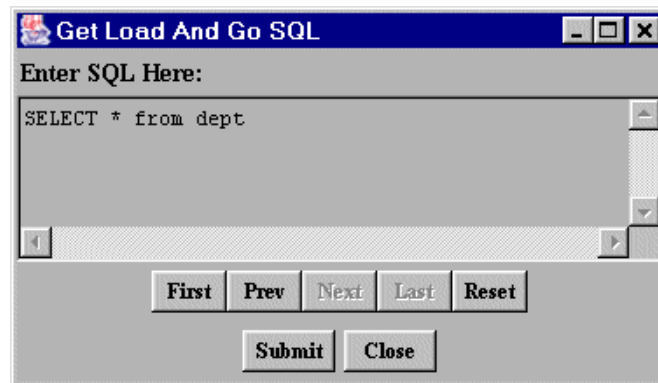
Get Cell Set Cell

Update Row Delete Row Insert Row Move to insert row Move to current row

Close

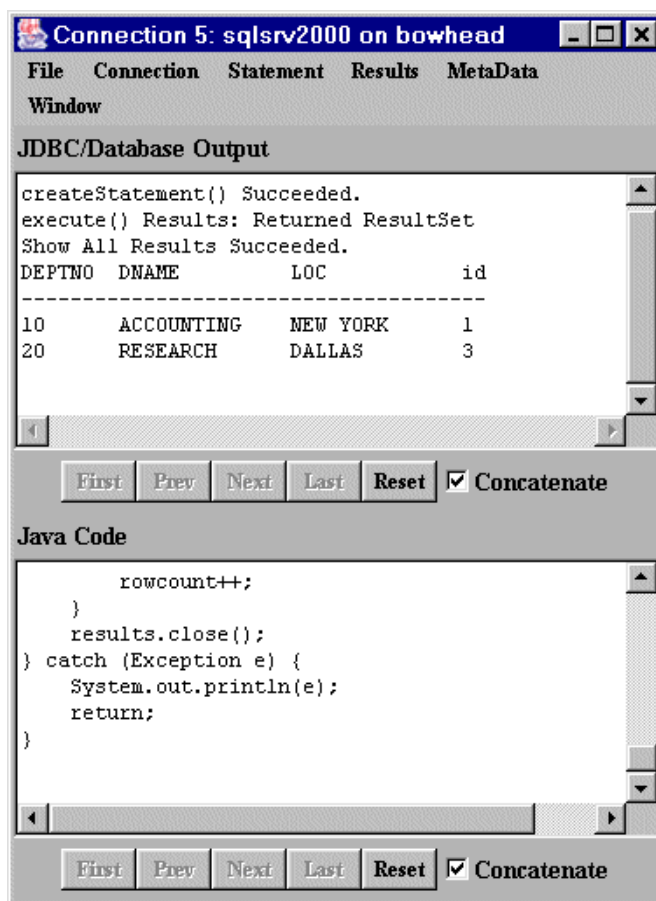
- 6 Click **Move to insert row**; Current Row is now Insert row.

- 7 Change Data Type to int. In Set Cell Value, enter 20. Click **Set Cell**.
- 8 Select the second row in the top pane. Change the Data Type to String. In Set Cell Value, enter RESEARCH. Click **Set Cell**.
- 9 Select the third row in the top pane. In Set Cell Value, enter DALLAS. Click **Set Cell**.
- 10 Click **Insert Row**.
- 11 To verify the result, return to the Connection menu and select **Connection / Load and Go**. The Get Load and Go SQL window appears.
- 12 Specify the statement that you want to execute.



Click **Submit**; then, click **Close**.

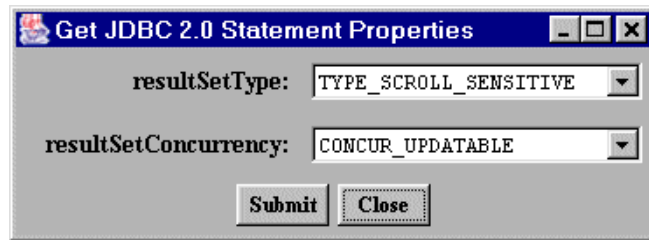
13 The Connection window shows two rows returned.



Note that the ID will be 3 for the row just inserted, because it is an auto increment column.

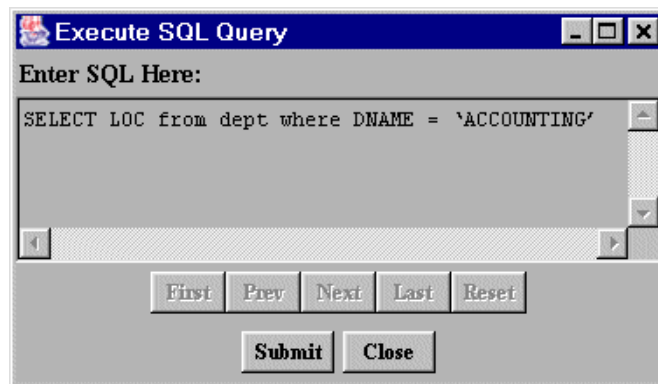
## Updating a Row

- 1 From the Connection window menu, select **Connection / Create JDBC 2.0 Statement**.
- 2 In the resultSetType field, select **TYPE\_SCROLL\_SENSITIVE**. In the resultSetConcurrency field, select **CONCUR\_UPDATABLE**.



Click **Submit**; then, click **Close**.

- 3 Select **Statement / Execute Stmt Query**.
- 4 Specify the Select statement that you want to execute.



Click **Submit**; then, click **Close**.



- 5 Select **Results / Inspect Results**. The Inspect Result Set window is displayed.

**Inspect Result Set**

Col#	Name	Type
1	LOC	char

Current Row: 1

Retrieve By Column: Index

Data Type: String Use Scale: ☐ 0

Use Calendar: ☐ Zone: Etc/GMT+12

Get Cell Value:

Set Cell Value:

Auto Traverse: ☒

Absolute 1 Relative

Before First Prev **Next** Last After

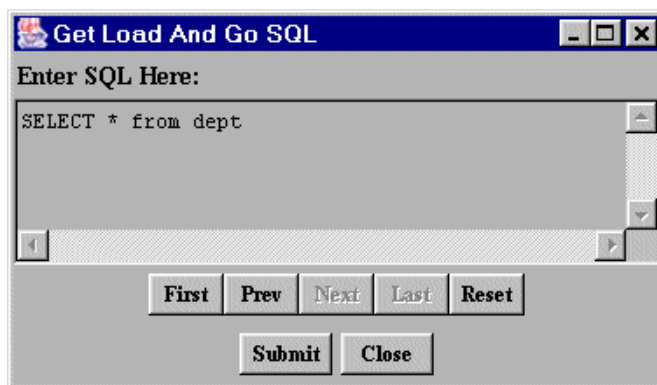
Get Cell Set Cell

Update Row Delete Row Insert Row Move to insert row Move to current row

Close

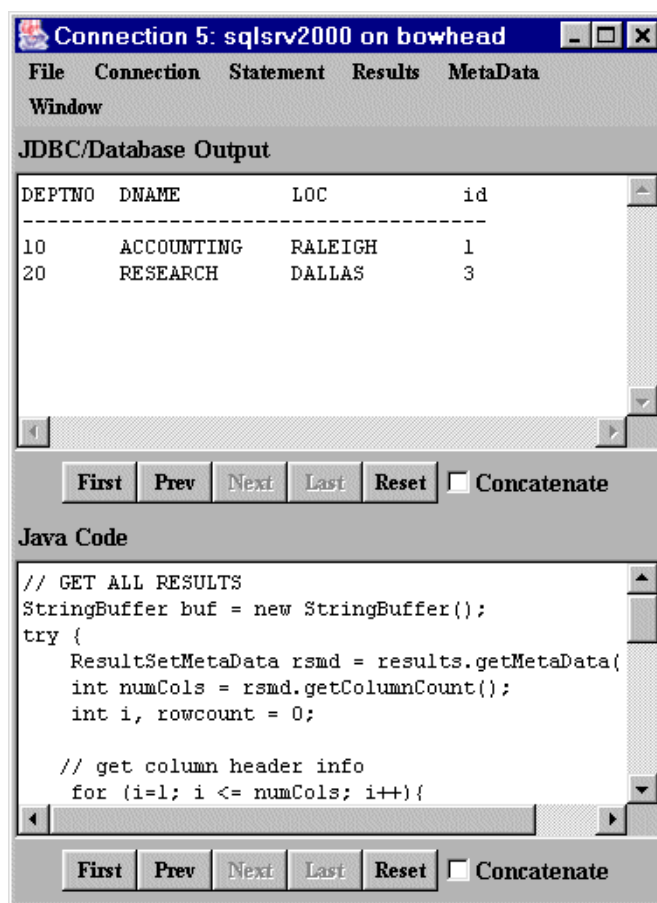
- 6 Click **Next**. Current Row changes to 1.
- 7 In Set Cell Value, enter RALEIGH. Click **Set Cell**.

- 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.

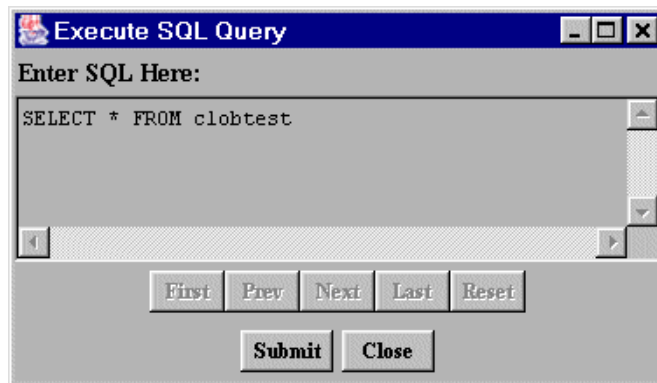


## LOB Support

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.

**Inspect Result Set**

Col#	Name	Type
1	CLOBCOL	clob

Current Row: 1

Retrieve By Column: Index

Data Type: String Use Scale: ☐ 0

Use Calendar: ☐ Zone: Etc/GMT+12

Get Cell Value:

Set Cell Value:

Auto Traverse: ☒

Absolute 1 Relative

Before First Prev Next Last After

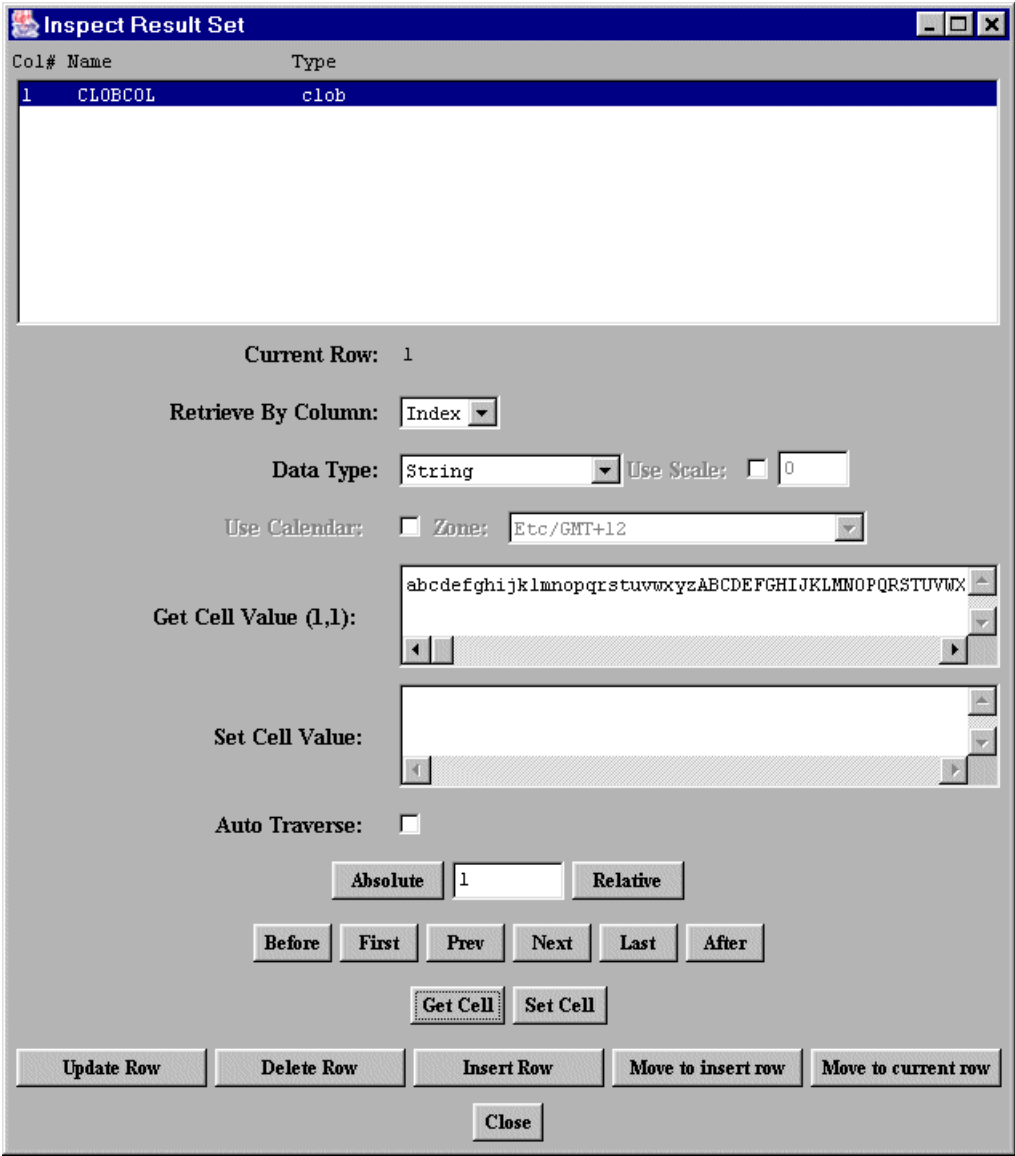
Get Cell Set Cell

Update Row Delete Row Insert Row Move to insert row Move to current row

Close

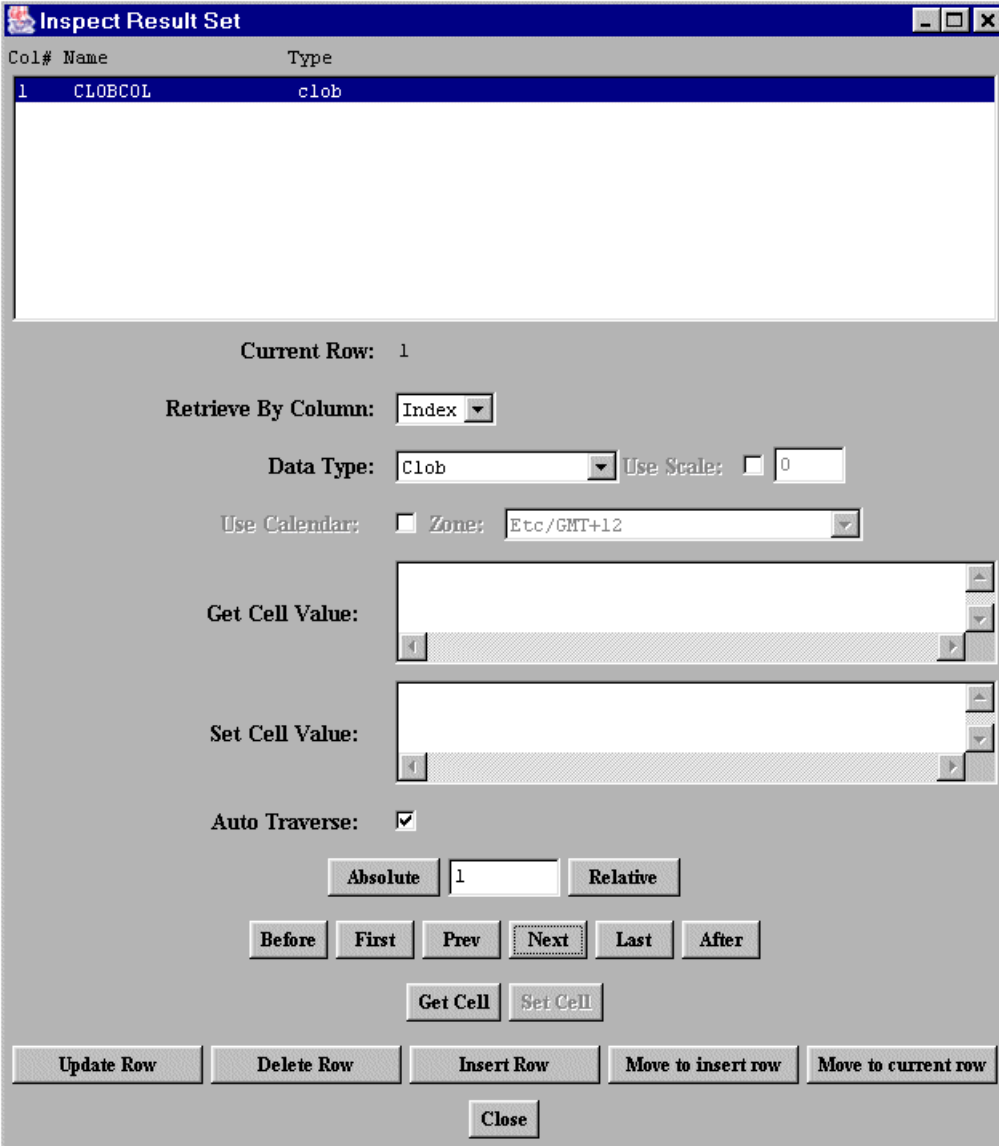
- 6 Deselect Auto Traverse. This disables automatic traversal to the next row.

7 Click **Get Cell**.



8 Values are returned in the Get Cell Value field.

## 9 Change the Data Type to Clob.



The **Inspect Result Set** dialog box is used for inspecting and modifying data in a result set. It features a table with columns **Col#**, **Name**, and **Type**. The first row shows **1**, **CLOBCOL**, and **clob**. Below the table, the **Current Row** is set to **1**. The **Retrieve By Column** dropdown is set to **Index**. The **Data Type** dropdown is set to **Clob**, and the **Use Scale** checkbox is unchecked with a value of **0**. The **Use Calendar** checkbox is unchecked, and the **Zone** dropdown is set to **Etc/GMT+12**. There are two large text input fields for **Get Cell Value** and **Set Cell Value**, each with a small grid icon to its left. The **Auto Traverse** checkbox is checked. Below these are buttons for **Absolute** and **Relative** traversal, with a text input field containing **1**. A row of navigation buttons includes **Before**, **First**, **Prev**, **Next** (highlighted with a dotted border), **Last**, and **After**. Below these are **Get Cell** and **Set Cell** buttons. At the bottom are buttons for **Update Row**, **Delete Row**, **Insert Row**, **Move to insert row**, **Move to current row**, and a **Close** button.

Col#	Name	Type
1	CLOBCOL	clob

Current Row: 1

Retrieve By Column: Index

Data Type: Clob Use Scale: ☐ 0

Use Calendar: ☐ Zone: Etc/GMT+12

Get Cell Value:

Set Cell Value:

Auto Traverse: ☒

Absolute 1 Relative

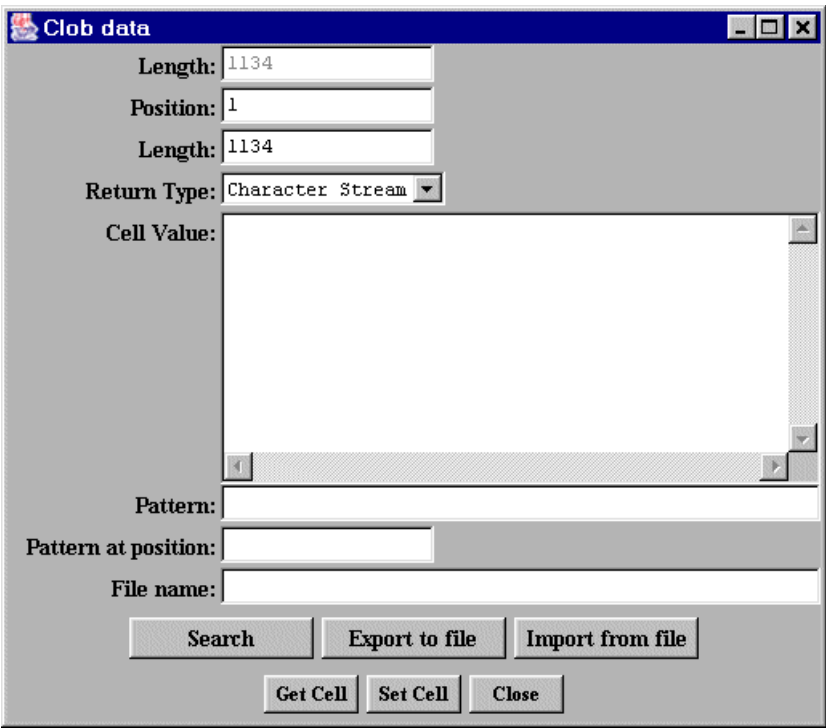
Before First Prev Next Last After

Get Cell Set Cell

Update Row Delete Row Insert Row Move to insert row Move to current row

Close

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





11 Click Get Cell.

**Clob data**

Length: 1134

Position: 1

Length: 1134

Return Type: Character Stream

Cell Value: abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ

Pattern:

Pattern at position:

File name:

Search Export to file Import from file

Get Cell Set Cell Close

12 Values are returned in the Get Cell Value field.



## 6 Using the SequeLink Java Client

This chapter provides information about using JDBC applications with the SequeLink Java Client.

---

### About the SequeLink Java Client

The SequeLink Java 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 SequeLink Java 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 SequeLink Java Client:

- SequeLink JDBC Driver
- SequeLink Proxy Server
- JDBC Spy
- JDBC Test
- jXTransformer
- DataDirect Connection Pool Manager
- J2EE Connector Architecture (JCA) resource adapters

## SequeLink JDBC Driver

The SequeLink JDBC driver is compliant with the JDBC 3.0 specification. The SequeLink 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

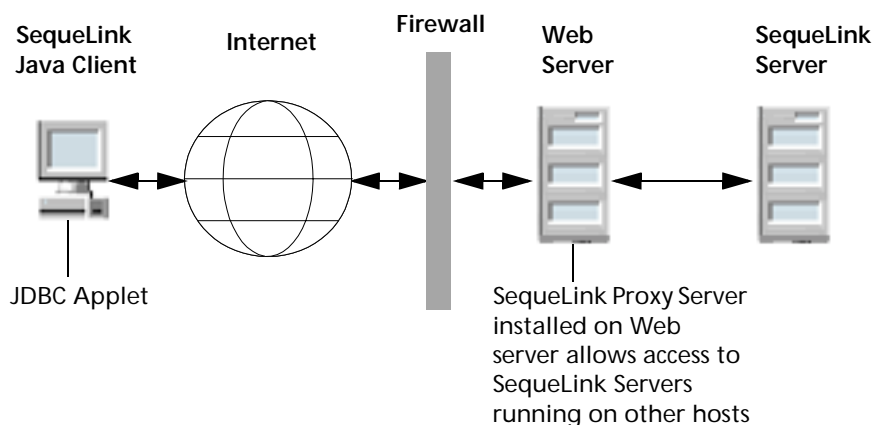
## SequeLink Proxy Server

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 6-1](#).

---

**Figure 6-1. 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 SequeLink Java 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*.

## JDBCSpy

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:

- Logging is JDBC 1.22- and JDBC 3.0-compliant, including support for the JDBC 2.1 Optional Package.
- Logging is consistent, regardless of the JDBC driver used.
- All parameters and function results for JDBC calls can be logged.
- Even if your JDBC driver does not support logging, you can still log JDBC calls.
- Logging can be enabled, without changing the application, using the SequeLinkDataSource object.
- Spy can only be used with the SequeLink JDBC Driver and the DataDirect Technologies Connect JDBC driver.

## JDBCTest

JDBCTest 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

## jXTransformer

jXTransformer is a development software component that defines and implements a grammar that allows you to:

- Read data from relational databases and transform that data into complex XML structures
- Write data stored in XML documents to relational databases

The jXTransformer grammar for reading data is similar to the SQL99 database language; the jXTransformer grammar for writing data is similar to a combination of the SQL99 database language and XML Path Language (XPath). jXTransformer uses a JDBC connection to communicate with the database. Refer to the *jXTransformer User's Guide* for more information.

## 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, “Connection Pool Manager,” on page 383](#) 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 SequeLink 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 SequeLink JDBC Driver 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. For more information about using resource adapters with SequeLink, see [“J2EE Connector Architecture \(JCA\) Resource Adapter” on page 244](#).

# SequeLink Java Client Directory Structure

Table 6-1 shows the SequeLink Java Client directory after installation and provides a description of the files.

**Table 6-1. SequeLink Java 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	Contains Java source files that provide examples of certificate checkers.
driver/examples/JNDI_FILESYSTEM_Example.java driver/examples/JNDI_LDAP_Example.java	Contains 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 SequeLink JDBC driver implementing the JDBC 2.1 Core API. To load the driver, add this path to your CLASSPATH variable. This JAR file also contains all classes of the SequeLink 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 SequeLink JDBC driver that implement Secure Socket Layer (SSL) encryption.



**Table 6-1. SequeLink Java Client Directory and Files** (cont.)

Directories and Files	Description
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 SequeLink JDBC driver that implement Secure Socket Layer (SSL) encryption.
driver/lib/sljc.rar	Resource Archive for use with J2EE and JCA Connectors.
help/*.*	Files for the HTML-based online help for the SequeLink JDBC driver.
jdbctest/lib/jdbctest.jar	Contains all the JDBCTest classes. To use JDBCTest, add this path to your CLASSPATH variable.
jdbctest/jdbctest.bat	Batch file that starts JDBCTest.
jdbctest/jdbctest.sh	UNIX shell script that starts JDBCTest.
jxtr/*.*	Contains the files containing classes used by JXTransformer.
pool/lib/pool.jar	Contains all the classes for the DataDirect Connection Pool Manager.
proxy/cmdsrv.exe	Executable that registers the SequeLink Proxy Server as a Windows service.
proxy/decrypt.bat proxy/encrypt.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	Batch file that starts the SequeLink Proxy Server.

**Table 6-1. SequeLink Java Client Directory and Files** (cont.)

Directories and Files	Description
proxy/proxyserver.sh	UNIX shell script that starts the SequeLink Proxy Server.
proxy/cert/	Contains demo certificates.
proxy/demos/com/ddtek/sequelink/demo/ demo.properties proxy/demos/com/ddtek/sequelink/demo/ GenerateDemoCertificates\$DN.class proxy/demos/com/ddtek/sequelink/demo/ GenerateDemoCertificates.class	Contains 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 6-1. SequeLink Java 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 SequeLink JDBC Driver

To use the SequeLink JDBC driver, you first must register it with the JDBC Driver Manager. You can register the SequeLink 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 SequeLink 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 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*.

<i>port</i>	is the TCP/IP port on which the SequeLink server is listening. A default installation of SequeLink Server uses the port 19996.
<i>key=value</i>	specifies connection properties. For a list of connection properties and their valid values, see <a href="#">“JDBC Connection Properties” on page 249</a> .

### JDBC Connection URL Examples:

The following examples show some typical SequeLink 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 239](#), 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 SequeLink Java 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 SequeLink Java 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 SequeLink 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.

For instructions on creating JDBC data sources, see [“Creating and Managing JDBC Data Sources” on page 239](#).

## 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 240](#) and [“Example B: With Connection Pooling” on page 241](#).

### 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.

For instructions on creating JDBC data sources, see [“Creating and Managing JDBC Data Sources” on page 239](#). For more information on the DataDirect Connection Pool Manager, see [Appendix D, “Connection Pool Manager,” on page 383](#).

# Using the Java Transaction API

[Table 6-2](#) lists which databases are supported for the Java Transaction API (JTA) by the SequeLink Java Client.

**Table 6-2. Support for the Java Transaction API (JTA) by the SequeLink Java Client**

Database	JTA Supported?
Oracle8, Oracle9i	Yes
Informix 9	Yes
DB2 V5, V6, V7R2 on OS/390	Yes
DB2 V5, V6.1, 7.1 on UNIX, Windows NT, Windows 2000, Windows XP	Yes
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.

For instructions on creating JDBC data sources, see [“Creating and Managing JDBC Data Sources”](#) on page 239.

---

## 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 SequeLink 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.

```
import java.util.Hashtable;
import java.sql.Connection;
import javax.sql.DataSource;
import javax.naming.*;

import com.ddtek.resource.jdbc.JCAConnectionFactory;
import com.ddtek.resource.jdbc.spi.*;

public class RAExample {
```

```

static public void main(String[] args) {
    try {

        // Create a connection factory instance
        ManagedConnectionFactory managedFactory =
            new SequeLinkManagedConnectionFactory();

        managedFactory.setServerName("MyOracleServer");
        managedFactory.setPortNumber("1521");
        managedFactory.setSID("TEST");

        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("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. For a list of the connection properties, see [“JDBC Connection Properties” on page 249](#).

### 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

### 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`



- JDBC DataSource object
- Server data sources specified using the SequeLink Manager

---

## JDBC Connection Properties

[Table 6-3](#) lists the JDBC connection properties supported by the SequeLink JDBC driver, describes each property, and specifies the methods with which it can be specified.

---

**Table 6-3. JDBC Properties**

---

Property	Description
ApplicationName	<p>Identifies the application that is establishing the connections. When the application does not provide a value, the default value is SequeLink JDBC Application.</p> <p>This property can be specified using:</p> <ul style="list-style-type: none"> <li>■ JDBC data source</li> <li>■ URL</li> <li>■ <code>java.util.properties</code></li> </ul>

---

Table 6-3. *JDBC Properties* (cont.)

Property	Description
blockFetchForUpdate	<p>BlockFetchForUpdate={0   1}. Specifies a workaround connection attribute. When the isolation level is Read Committed and a SELECT FOR UPDATE statement is issued against some data stores, the SequeLink Java Client does not lock the expected row.</p> <p>When set to 0, the appropriate row is locked.</p> <p>When set to 1 (the initial default), the appropriate row is not locked.</p> <p>IMPORTANT: Specifying 0 will degrade performance for SELECT FOR UPDATE statements because rows will be fetched one at a time.</p> <p>This property can be specified using:</p> <ul style="list-style-type: none"><li>■ JDBC data source</li><li>■ URL</li><li>■ java.util.properties</li><li>■ server data source</li></ul>
certificateChecker	<p>The fully qualified class name of a user-defined server certificate checker class. When the SequeLink Client and SequeLink Server have agreed on an SSL cipher suite that requires a server certificate, this class is used to verify the server certificate on behalf of the client. The class must be an implementation of the com.ddtek.sequelink.cert.CertificateCheckerInterface interface.</p> <p>This property can be specified using:</p> <ul style="list-style-type: none"><li>■ JDBC data source</li><li>■ URL</li><li>■ java.util.properties</li></ul> <p>For more information about certificate checker classes, refer to the <i>SequeLink Administrator's Guide</i>.</p>

**Table 6-3. JDBC Properties** (cont.)

Property	Description
cipherSuites	<p>The Secure Socket Layer (SSL) cipher suites with which the SequeLink Java Client can use to connect. This property is required when networkProtocol=ssl.</p> <p>For a list of supported cipher suites, refer to the <i>SequeLink Administrator's Guide</i>.</p> <p>This property can be specified using:</p> <ul style="list-style-type: none"> <li>■ JDBC data source</li> <li>■ URL</li> <li>■ java.util.properties</li> </ul>
databaseName	<p>The name of the data store to which you want to connect.</p> <p>This property can be specified using:</p> <ul style="list-style-type: none"> <li>■ JDBC data source</li> <li>■ URL</li> <li>■ java.util.properties</li> <li>■ server data source</li> </ul>
DBPassword	<p>The data store password, which may be required depending on the server configuration.</p> <p>This property can be specified using:</p> <ul style="list-style-type: none"> <li>■ JDBC data source</li> <li>■ URL</li> <li>■ java.util.properties</li> </ul>
DBUser	<p>The data store user name, which may be required depending on the server configuration.</p> <p>This property can be specified using:</p> <ul style="list-style-type: none"> <li>■ JDBC data source</li> <li>■ URL</li> <li>■ java.util.properties</li> </ul>
describeParam	<p>A description of the parameter.</p> <p>This property only can be specified using a server data source.</p>
description	A description of the connection or data source.

**Table 6-3. JDBC Properties** (cont.)

Property	Description
HPassword	<p>The host password, which may be required depending on the server configuration.</p> <p>This property can be specified using:</p> <ul style="list-style-type: none"><li>■ JDBC data source</li><li>■ URL</li><li>■ java.util.properties</li></ul>
HUser	<p>The host user name, which may be required depending on the server configuration.</p> <p>This property can be specified using:</p> <ul style="list-style-type: none"><li>■ JDBC data source</li><li>■ URL</li><li>■ java.util.properties</li></ul>
MSSMapLongtoDecimal	<p>Turns on client-side workarounds that allow you to take full advantage of the SequeLink JDBC driver with JDBC applications that require non-standard or extended behavior. For more information, refer to the <i>SequeLink Administrator's Guide</i>.</p> <p>This property can be specified using:</p> <ul style="list-style-type: none"><li>■ JDBC data source</li><li>■ URL</li><li>■ java.util.properties</li><li>■ server data source</li></ul>
networkProtocol	<p>networkProtocol={socket   ssl}. Specifies the protocol to be used.</p> <p>When set to socket (the initial default), SSL encryption is not used.</p> <p>When set to ssl, SSL encryption is used.</p> <p>This property can be specified using:</p> <ul style="list-style-type: none"><li>■ JDBC data source</li><li>■ URL</li><li>■ java.util.properties</li></ul>

**Table 6-3. JDBC Properties** (cont.)

Property	Description
newPassword	<p>The new host password to be used. If specified and applicable to the connection, the SequeLink password change mechanism is invoked. When the password has been changed successfully, the following warning is returned:</p> <pre>[DataDirect][SequeLink JDBC driver][SequeLink Server] The user password was changed successfully</pre> <p>This property can be specified using:</p> <ul style="list-style-type: none"> <li>■ JDBC data source</li> <li>■ URL</li> <li>■ java.util.properties</li> </ul> <p>For more information about the SequeLink password change mechanism, refer to the <i>SequeLink Administrator's Guide</i>.</p>
ORANumberOfIsNumeric	<p>Turns on client-side workarounds that allow you to take full advantage of the SequeLink JDBC driver with JDBC applications that require non-standard or extended behavior. For more information, refer to the <i>SequeLink Administrator's Guide</i>.</p> <p>This property can be specified using:</p> <ul style="list-style-type: none"> <li>■ JDBC data source</li> <li>■ URL</li> <li>■ java.util.properties</li> <li>■ server data source</li> </ul>
password	<p>The host or data store password, which may be required depending on the server configuration.</p> <p>This property can be specified using:</p> <ul style="list-style-type: none"> <li>■ getConnection</li> <li>■ JDBC data source</li> <li>■ URL</li> <li>■ java.util.properties</li> </ul>

**Table 6-3. JDBC Properties** *(cont.)*

Property	Description
portNumber	<p>The TCP/IP port on which the SequeLink service is listening.</p> <p>This property can be specified using:</p> <ul style="list-style-type: none"><li>■ JDBC data source</li><li>■ URL</li></ul>
serverDataSource	<p>A property that specifies a string to identify the server data source to be used for the connection. If unspecified, the configuration of the default server data source will be used for the connection.</p> <p>This property can be specified using:</p> <ul style="list-style-type: none"><li>■ JDBC data source</li><li>■ URL</li><li>■ java.util.properties</li></ul>
serverName	<p>The TCP/IP address of the SequeLink server in dotted format or host name format.</p> <p>This property can be specified using:</p> <ul style="list-style-type: none"><li>■ JDBC data source</li><li>■ URL</li></ul>
SLKStaticCursorLongColBufLen	<p>The amount of data (in KB) that is buffered for SQL_LONGVARCHAR and SQL_LONGVARBINARY columns with an insensitive result set.</p> <p>The default is 4.</p> <p>This property can be specified using:</p> <ul style="list-style-type: none"><li>■ JDBC data source</li><li>■ URL</li><li>■ java.util.properties</li><li>■ server data source</li></ul>
spyAttributes	<p>A property that enables Spy when making connections with a JDBC data source.</p> <p>This property only can be specified using a JDBC data source.</p>

---

**Table 6-3. JDBC Properties** (cont.)

---

Property	Description
user	<p>The host or data store user name, which may be required depending on the server configuration.</p> <p>This property can be specified using:</p> <ul style="list-style-type: none"><li>■ getConnection</li><li>■ JDBC data source</li><li>■ URL</li><li>■ java.util.properties</li></ul>

---

---

## Testing SequeLink JDBC Connections

For instructions on connecting with the SequeLink Java Client using JDBCTest, see [Chapter 5 “Using JDBCTest” on page 175](#).

---

## Using the SequeLink Java Client on a Java 2 Platform

When using the SequeLink 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 SequeLink Java Client code base only:

```
// permissions granted to the SequeLink Java 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 Java 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 SequeLink Java 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.

# 7 Tracking JDBC Calls

This chapter introduces 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:

- Logging is JDBC 1.22- and JDBC 3.0-compliant, including support for the JDBC 2.1 Optional Package.
- Logging is consistent, regardless of the JDBC driver used.
- All parameters and function results for JDBC calls can be logged.
- Even if your JDBC driver does not support logging, you can still log JDBC calls.
- Logging can be enabled, without changing the application, using the `SequeLinkDataSource` object.
- Spy can only be used with the SequeLink JDBC driver and the DataDirect Technologies Connect JDBC driver.

---

## 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:  
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.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:

<code>log=System.out</code>	Redirects logging to the Java output standard.
<code>log=(file)<i>filename</i></code>	Redirects logging to the file specified by <i>filename</i> . By default, Spy will use the stream specified in <code>DriverManager.setLogStream()</code> .
<code>load=<i>classname</i></code>	Loads the driver specified by <i>classname</i> . The default value is <code>com.ddtek.sequellink.jdbc.SequelLinkDriver</code> .
<code>linelimit=<i>numberofchars</i></code>	The maximum number of characters, specified by <i>numberofchars</i> , that Spy will log on one line. The default is 0 (no maximum limit).
<code>logIS={yes   no   nosingleread}</code>	Specifies whether Spy logs activity on <code>InputStreams</code> .  <code>nosingleread</code> =turns on logging for <code>InputStreams</code> (but not <code>InputStream.read()</code> messages), allowing logging on <code>InputStreams</code> 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.

---

## Using Spy with JDBC Data Sources

The SequeLink JDBC driver implements the following features defined by the JDBC 2.1 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 261](#). For more information about configuring JDBC data sources, see [“JDBC Connection URL Examples:” on page 237](#).

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 Examples

## Example A:

```
jdbc:spy:{jdbc:odbc:Oracle8};load=sun.jdbc.odbc.JdbcOdbcDriver;  
log=(file)C:\temp\spy.log
```

Using this example, Spy would:

- 1 Load the JDBC-ODBC bridge.
- 2 Log all JDBC activity to the file c:\temp\spy.log.

## Example B:

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

Using this example, Spy would:

- 1 Load the SequeLink 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.<sup>1</sup>

```
registerDriver:driver[className=com.ddtek.jdbcspy.SpyDriver,
context=null,com.ddtek.jdbcspy.SpyDriver@1ec49f]2
```

```
*Driver.connect(jdbc:spy:{jdbc:sequelink://QANT:4003;databaseName=Oracle;})
    trying driver[className=com.ddtek.jdbcspy.SpyDriver,
context=null,com.ddtek.jdbcspy.SpyDriver@1ec49f]3
```

```
spy>> Driver.connect(String url, Properties info)
spy>> url = jdbc:spy:{jdbc:sequelink://QANT:4003;databaseName=Oracle;
OSUser=gauser;OSPassword=null12}
spy>> info = {password=tiger, user=scott}
spy>> OK (Connection[1])4
```

```
getConnection returning driver[className=com.ddtek.jdbcspy.SpyDriver,
context=null,com.ddtek.jdbcspy.SpyDriver@1ec49f]5
```

```
spy>> Connection[1].getWarnings()
spy>> OK6
```

```
spy>> Connection[1].createStatement
spy>> OK (Statement[1])7
```

```
spy>> Statement[1].executeQuery(String sql)
spy>> sql = select empno,ename,job from emp where empno=7369
spy>> OK (ResultSet[1])8
```

```
spy>> ResultSet[1].getMetaData()
spy>> OK (ResultSetMetaData[1])9
```

```
spy>> ResultSetMetaData[1].getColumnCount()
spy>> OK (3)10
```



```
spy>> ResultSetMetaData[1].getColumnLabel(int column)
spy>> column = 1
spy>> OK (EMPNO)11

spy>> ResultSetMetaData[1].getColumnLabel(int column)
spy>> column = 2
spy>> OK (ENAME)12

spy>> ResultSetMetaData[1].getColumnLabel(int column)
spy>> column = 3
spy>> OK (JOB)13

spy>> ResultSet[1].next()
spy>> OK (true)14

spy>> ResultSet[1].getString(int columnIndex)
spy>> columnIndex = 1
spy>> OK (7369)15

spy>> ResultSet[1].getString(int columnIndex)
spy>> columnIndex = 2
spy>> OK (SMITH)16

spy>> ResultSet[1].getString(int columnIndex)
spy>> columnIndex = 3
spy>> OK (CLERK)17

spy>> ResultSet[1].next()
spy>> OK (false)18

spy>> ResultSet[1].close()
spy>> OK19

spy>> Connection[1].close()
spy>> OK20
```

## NOTES:

- 1: The Spy driver is registered. The spy>> prefix indicates that this line has been logged by Spy.
- 2: The JDBC Driver Manager logs a message each time a JDBC driver is registered.
- 3: This is the logging of the JDBC Driver Manager. It logs a message each time a JDBC application makes a connection.
- 4: The application connects with the specified URL. The User Name and Password are specified using properties.
- 5: This is the logging of the JDBC Driver Manager. It logs a message each time a successful connection is made.
- 6: The application checks to see if there are any warnings. In this example, no warnings are present.
- 7 and 8: The statement "select empno,ename,job from emp where empno=7369" is created.
- 9, 10, 11, 12, and 13: Some metadata is requested.
- 14, 15, 16, and 17: The first row is fetched.
- 18: The application attempts to fetch the second row, but the database returned only one row for this query.
- 19: After fetching all data, the result set is closed.
- 20: The application finishes and disconnects.

## 8 Developing JDBC Applications

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

- [“JDBC 3.0 Support” on page 268](#)
- [“SQL Escape Sequences” on page 269](#)
- [“Data Types and Isolation Levels” on page 269](#)
- [“Threading” on page 270](#)
- [“Using Scrollable Cursors” on page 272](#)
- [“Specifying Application IDs” on page 275](#)
- [“Error Handling” on page 276](#)
- [“Fine-Tuning JDBC Application Performance” on page 278](#)

# JDBC 3.0 Support

The SequeLink 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 JDBC driver supports.

**Table 8-1. Supported JDBC Features**

	DB2/390 V6, 7	DB2 UDB	Informix	Oracle8, 9i	SQL Server 7, 2000	Sybase
Blob interfaces (JDBC 2.0 methods)	X	X	X	X		
Clob interfaces (JDBC 2.0 methods)	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.

For information on the methods that SequeLink supports and the compatibility between the JDBC application versions, see [Appendix C “JDBC Support” on page 331](#).

---

## 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 244](#) for information about using the SequeLink JDBC driver as a JCA resource adapter.

---

## SQL Escape Sequences

See [Appendix A “SQL Escape Sequences for ODBC and JDBC” on page 297](#) for information about the SQL escape sequences supported by the SequeLink JDBC driver.

---

## Data Types and Isolation Levels

The data types and isolation levels supported by the SequeLink JDBC driver depend on the data store to which you are connecting. See [Appendix B “Data Types and Isolation Levels” on page 317](#) for database-specific information about data types and isolation levels.

---

## Threading

The SequeLink 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 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 effect 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, V6R1, V7R1 on OS/390	Ignored
DB2 V6, V7R2 on Windows	OK
DB2 V6, V7R2 on UNIX	Ignored
Informix 9, 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
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 OS/390	X	
Informix	X	X
Microsoft SQL Server (see note)	X	X
ODBC Socket Server	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 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 JDBC driver supports updatable result sets.

NOTE: When the SequeLink 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 JDBC driver, immediately invoke `setApplicationId`. The `setApplicationId` method is defined on the interface `com.ddtek.jdbc.extensions.SIExtensionInterface`, 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.SIExtensionInterface;

...
Connection con = DriverManager.getConnection(...);

String appId = "myAppID";
if (con instanceof SIExtensionInterface)
{
    SIExtensionInterface slCon = (SIExtensionInterface)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*.

---

## Error Handling

The SequeLink JDBC driver reports errors to the calling application by returning `SQLExceptions`. Errors can be generated by the following components:

- SequeLink JDBC driver
- SequeLink Server
- Database

### SequeLink JDBC Driver Errors

An error generated by the SequeLink 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 JDBC driver is packaged into the following JAR files:

- sljc.jar contains all classes of the SequeLink JDBC driver.
- sslsl.jar contains all classes of the SequeLink JDBC driver implementation of SSL. This file is only required if you will be using SSL encryption.

To use the SequeLink 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,sslsl.jar>
<param name=ConfigFile value=Config.txt>
</applet>
```

## Fetching BigDecimal Objects

JDBC 1.22 defines `getBigDecimal()` with a scale parameter. When the SequeLink 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 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", "john", "WSTable", "TABLE");
```

Sometimes, little information is known about the object for which you are requesting information. Any information that the application can send the driver when calling database metadata methods can result in improved performance and reliability.

## ***Using a Dummy Query to Determine Table Characteristics***

Avoid using `getColumns` to determine characteristics about a table. Instead, use a dummy query with `getMetadata`.

Consider an application that allows the user to choose columns. Should the application use `getColumns` to return information about the columns to the user or instead prepare a dummy query and call `getMetadata`?

***Case 1: GetColumns Method***

```

ResultSet WSrc = WSc.getColumns (... "UnknownTable" ...);
// This call to getColumns will generate a query to
// the system catalogs... possibly a join
// which must be prepared, executed, and produce
// a result set
. . .
WSrc.next();
string Cname = getString(4);
. . .
// user must retrieve N rows from the server
// N = # result columns of UnknownTable
// result column information has now been obtained

```

***Case 2: GetMetadata Method***

```

// prepare dummy query
PreparedStatement WSps = WSc.prepareStatement
    (... "SELECT * from UnknownTable WHERE 1 = 0" ...);
// query is never executed on the server -
// only prepared
ResultSetMetaData WSsmd=wsps.getMetaData();
int numcols = WSsmd.getColumnCount();
...
int ctype = WSsmd.getColumnType(n)
...
// result column information has now been obtained
// Note we also know the column ordering within the table!
// This information cannot be assumed from the getColumns example.

```

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

To somewhat complicate this discussion, let us consider a DBMS server that does not natively support preparing a SQL statement. The performance of Case 1 does not change but Case 2 increases minutely because the dummy query must be evaluated instead of only prepared. Because the Where clause of the query always evaluates to FALSE, the query generates no result rows and

should execute without accessing table data. For this situation, Case 2 still outperforms Case 1.

## Retrieving Data

This section provides general guidelines for retrieving data with JDBC applications.

### *Retrieving Long Data*

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

Most users don't want to see long data. If the user does need to see these result items, the application can query the database again, specifying only the long columns in the select list. This method allows the average user to retrieve result sets without having to pay a high performance penalty for network traffic.

Although the best method is to exclude long data from the select list, some applications do not formulate the select list before sending the query to the JDBC driver (for example, some applications `select * from <table name> ...`). If the select list contains long data, the driver must retrieve that data at fetch time, even if the application does not get the long data in the result set. When possible, the application developer should use a method that does not retrieve all columns of the table.

Additionally, although the `getClob` and `getBlob` methods allow the application to control how long data is retrieved in the application, the designer must realize that in many cases, the JDBC driver emulates these methods due to the lack of true locator support in the DBMS. In such cases, the driver must retrieve all of the long data across the network before exposing the `getClob` and `getBlob` methods.

Sometimes long data must be retrieved. When this is the case, remember that most users do not want to see 100 KB, or more, of text on the screen.

### ***Reducing the Size of Data Retrieved***

To reduce network traffic and improve performance, you can reduce the size of data being retrieved to a manageable limit by calling `setMaxRows`, `setMaxFieldSize`, and the driver-specific `SetFetchSize`. Another method of reducing the size of the data being retrieved is to decrease the column size. If the driver allows you to define the packet size, use the smallest packet size that will meet your needs.

In addition, be careful to return only the rows you need. If you return five columns when you only need two columns, performance is decreased, especially if the unnecessary rows include long data.

### ***Choosing the Right Data Type***

Retrieving and sending certain data types can be expensive. When you design a schema, select the data type that can be processed most efficiently. For example, integer data is processed faster than floating-point data. Floating-point data is defined according to internal database-specific formats, usually in a compressed format. The data must be decompressed and converted into a different format so that it can be processed by the wire protocol.

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

## Selecting JDBC Objects and Methods

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

### *Using Parameter Markers as Arguments to Stored Procedures*

When calling stored procedures, always use parameter markers for the argument markers instead of using literal arguments. JDBC drivers can call stored procedures on the database server either by executing the procedure as any other SQL query, or by optimizing the execution by invoking a Remote Procedure Call (RPC) directly into the database server. Executing the stored procedure as a SQL query results in the database server parsing the statement, validating the argument types, and converting the arguments into the correct data types. Remember that SQL is always sent to the database server as a character string, for example, "{call getCustName (12345)}". In this case, even though the application programmer might assume that the only argument to getCustName is an integer, the argument is actually passed inside a character string to the server. The database server would parse the SQL query, isolate the single argument value 12345, then convert the string '12345' into an integer value.

By invoking an RPC inside the database server, the overhead of using a SQL character string is avoided. Instead, the procedure is called only by name with the argument values already encoded into their native data types.

**Case 1**

Stored Procedure cannot be optimized to use a server-side RPC. The database server must parse the statement, validate the argument types, and convert the arguments into the correct data types. The database server must parse the statement, validate the argument types, and convert the arguments into the correct data types.

```
CallableStatement cstmt = conn.prepareCall ("call getCustName (12345)");
ResultSet rs = cstmt.executeQuery ();
```

**Case 2**

Stored Procedure can be optimized to use a server-side RPC. Because the application calls the procedure by name and the argument values are already encoded, the load on the database server is less.

```
CallableStatement cstmt = conn.prepareCall ("Call getCustName (?)");
cstmt.setLong (1,12345);
ResultSet rs = cstmt.executeQuery();
```

## ***Using the Statement Object instead of the PreparedStatement Object***

JDBC drivers are optimized based on the perceived use of the functions that are being executed. Choose between the PreparedStatement object and the Statement object depending on the planned use. The Statement object is optimized for a single execution of a SQL statement. In contrast, the PreparedStatement object is optimized for SQL statements that will be executed two or more times.

The overhead for the initial execution of a PreparedStatement object is high. The advantage comes with subsequent executions of the SQL statement.

## ***Choosing the Right Cursor***

Choosing the appropriate type of cursor allows maximum application flexibility. This section summarizes the performance issues of three types of cursors.

A forward-only cursor provides excellent performance for sequential reads of all of the rows in a table. However, it cannot be used when the rows to be returned are not sequential.

Insensitive cursors used by JDBC drivers are ideal for applications that require high levels of concurrency on the database server and require the ability to scroll forwards and backwards through result sets. The first request to an insensitive cursor fetches all of the rows and stores them on the client. Thus, the first request is very slow, especially when long data is retrieved. Subsequent requests do not require any network traffic and are processed quickly. Because the first request is processed slowly, insensitive cursors should not be used for a single request of one row. Designers should also avoid using insensitive cursors when long data is returned, because memory can be exhausted. Some insensitive cursor implementations cache the data in a temporary table on the database server and avoid the performance issue.

Sensitive cursors, sometimes called keyset-driven cursors, use identifiers, such as a ROWID, that already exist in your database. When you scroll through the result set, the data for the identifiers is retrieved. Because each request generates network traffic, performance can be very slow. However, returning nonsequential rows does not further affect performance. Sensitive cursors are the preferred scrollable cursor model for dynamic situations, when the application cannot afford to buffer the data from an insensitive cursor.

## 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 updateXXX methods of the ResultSet object allows the developer to update data without building a complex SQL statement. Instead, the developer simply supplies the column in the result set that is to be updated and the data that is to be changed. Then, before moving the cursor from the row in the result set, the updateRow method must be called to update the database as well.

In the following code fragment, the value of the Age column of the Resultset object rs is retrieved using the method `getInt`, and the method `updateInt` is used to update the column with an int value of 25. The method `updateRow` is called to update the row in the database that contains the modified value.

```
int n = rs.getInt("Age");
// n contains value of Age column in the resultset rs
. . .
rs.updateInt("Age", 25);
rs.updateRow();
```

In addition to making the application more easily maintainable, programmatic updates usually result in improved performance. Because the database server is already positioned on the row for the Select statement in process, performance-expensive operations to locate the row to be changed are not needed. If the row must be located, the server usually has an internal pointer to the row available (for example, ROWID).

### ***Using `getBestRowIdentifier()`***

Use `getBestRowIdentifier()` to determine the optimal set of columns to use in the Where clause for updating data. Pseudo-columns often provide the fastest access to the data, and these columns can only be determined by using `getBestRowIdentifier()`.

Some applications cannot be designed to take advantage of positional updates and deletes. Some applications might formulate the Where clause by using all searchable result columns by calling `getPrimaryKeys()`, or by calling `getIndexInfo()` to find columns that might be part of a unique index. These methods usually work, but might result in fairly complex queries.

Consider the following example:

```
ResultSet WSrs = WSs.executeQuery
    ("SELECT first_name, last_name, ssn, address, city, state, zip
     FROM emp");
// fetchdata
...
WSs.executeUpdate ("UPDATE EMP SET ADDRESS = ?
    WHERE first_name = ? and last_name = ? and ssn = ?
    and address = ? and city = ? and state = ?
    and zip = ?");
// fairly complex query
```

Applications should call `getBestRowIdentifier()` to retrieve the optimal set of columns (possibly a pseudo-column) that identifies a specific record. Many databases support special columns that are not explicitly defined by the user in the table definition but are "hidden" columns of every table (for example, ROWID and TID). These pseudo-columns generally provide the fastest access to the data because they typically are pointers to the exact location of the record. Because pseudo-columns are not part of the explicit table definition, they are not returned from `getColumns`. To determine if pseudo-columns exist, call `getBestRowIdentifier()`.

Consider the previous example again:

```
...
ResultSet WSrowid = getBestRowIdentifier()
    (... "emp", ...);
// Suppose this returned "ROWID"
...
ResultSet WSrs = WSs.executeQuery("SELECT first_name, last_name,
    ssn, address, city, state, zip, ROWID FROM emp");
// fetch data and probably "hide" ROWID from the user
...
WSs.executeUpdate ("UPDATE emp SET address = ? WHERE ROWID = ?");
// fastest access to the data!
```

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



# Part 4: Reference

This part contains the following appendixes:

- [Appendix A “SQL Escape Sequences for ODBC and JDBC” on page 297](#) 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 317](#) lists the data types and isolation levels supported for each data store supported by SequeLink.
- [Appendix C “JDBC Support” on page 331](#) provides information about JDBC compatibility and developing JDBC applications for SequeLink environments.
- [Appendix D “Connection Pool Manager” on page 383](#) describes how your application can use connection pooling through the DataDirect Connection Pool Manager.





# A SQL Escape Sequences for ODBC and JDBC

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 following language features:

- Date, time, timestamp, and datetime interval literals
- Scalar functions such as numeric, string, and data type conversion functions
- LIKE predicate escape characters
- Outer joins
- Procedure calls

The escape sequence used by ODBC and JDBC is:

```
{extension}
```

The escape sequence is recognized and parsed by the SequeLink ODBC driver or SequeLink JDBC driver, 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 [1]
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 SequeLink ODBC driver and the SequeLink JDBC driver, as shown in [Table A-1 “Scalar Functions Supported by the SequeLink ODBC Driver and SequeLink JDBC Driver” on page 299](#).

**Example:**

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

---

**Table A-1. Scalar Functions Supported by the SequeLink ODBC Driver and SequeLink JDBC Driver**

---

<b>Data Store</b>	<b>String Functions</b>	<b>Numeric Functions</b>	<b>Time/date Functions</b>	<b>System Functions</b>
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
DB2 V5 on Windows and UNIX	CONCAT LEFT LENGTH RIGHT SUBSTRING	Not supported	CURDATE CURRENT DATE CURTIME DAYOFMONTH HOUR MINUTE MONTH NOW QUARTER SECOND YEAR	IFNULL USERNAME

**Table A-1. Scalar Functions Supported by the SequeLink ODBC Driver and SequeLink JDBC Driver** *(cont.)*

Data Store	String Functions	Numeric Functions	Time/date Functions	System Functions
DB2 V6R1, V7R1, V7R2	ASCII	ABS	CURDATE	DBNAME
	CHAR	ACOS	CURRENT_DATE	IFNULL
	CONCAT	ASIN	CURTIME	USERNAME
	DIFFERENCE	ATAN	DAYNAME	
	INSERT	ATAN2	DAYOFMONTH	
	LCASE	CEILING	DAYOFWEEK	
	LEFT	COS	DAYOFYEAR	
	LENGTH	COT	HOURL	
	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-1. Scalar Functions Supported by the SequeLink ODBC Driver and SequeLink JDBC Driver (cont.)**

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-1.   Scalar Functions Supported by the SequeLink ODBC Driver and SequeLink JDBC Driver** *(cont.)*

Data Store	String Functions	Numeric Functions	Timedate Functions	System Functions
Microsoft SQL Server	ASCII	ABS	CURDATE	DBNAME
	BITLENGTH	ACOS	CURRENT_DATE	IFNULL
	CHAR	ASIN	CURRENT_TIME	USERNAME
	CONCAT	ATAN	CURRENT_	
	DIFFERENCE	ATAN2	TIMESTAMP	
	INSERT	CEILING	CURTIME	
	LCASE	COS	DAYOFMONTH	
	LEFT	COT	DAYOFWEEK	
	LENGTH	DEGREES	DAYOFYEAR	
	LOCATE	EXP	DAYNAME	
	LOCATE2	FLOOR	EXTRACT	
	LTRIM	LOG	HOURL	
	OCTET_LENGTH	LOG10	MINUTE	
	REPEAT	MOD	MONTH	
	REPLACE	PI	MONTHNAME	
	RIGHT	POWER	NOW	
	RTRIM	RADIANS	QUARTER	
	SOUNDEX	RAND	SECOND	
	SPACE	ROUND	TIMESTAMPADD	
	SUBSTRING	SIGN	TIMESTAMPDIFF	
	UCASE	SIN	WEEK	
		SQRT	YEAR	
		TAN		
		TRUNCATE		

**Table A-1. Scalar Functions Supported by the SequeLink ODBC Driver and SequeLink JDBC Driver (cont.)**

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

\* Oracle9i only

**Table A-1.    Scalar Functions Supported by the SequeLink ODBC Driver and SequeLink JDBC Driver** *(cont.)*

Data Store	String Functions	Numeric Functions	Timedate Functions	System Functions
Sybase	ASCII	ABS	CURDATE	DBNAME
	CHAR	ACOS	CURRENT DATE	IFNULL
	CONCAT	ASIN	DAYOFMONTH	USERNAME
	DIFFERENCE	ATAN	DAYOFWEEK	
	INSERT	ATAN2	DAYOFYEAR	
	LCASE	CEILING	DAYNAME	
	LEFT	COS	HOURL	
	LENGTH	COT	MINUTE	
	LTRIM	EXP	MONTH	
	REPEAT	FLOOR	MONTHNAME	
	RIGHT	LOG	NOW	
	RTRIM	LOG10	QUARTER	
	SOUNDEX	MOD	SECOND	
	SPACE	NUM_DEGREES	TIMESTAMPADD	
	SUBSTRING	NUM_RADIANS	TIMESTAMPDIFF	
	UCASE	PI	WEEK	
		POWER	YEAR	
		RAND		
		ROUND		
		SIGN		
		SIN		
		SQRT		
		TAN		
		TRUNCATE		



# String Functions

Table A-2 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, SQL\_VARCHAR, or SQL\_LONGVARCHAR.
- *start*, *length*, and *count* can be the result of another scalar function or a literal numeric value, where the underlying data type is SQL\_TINYINT, SQL\_SMALLINT, or SQL\_INTEGER.

The string functions are one-based; that is, the first character in the string is the character 1. Character string literals must be enclosed by single quotation marks.

---

**Table A-2. Scalar String Functions**

---

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

Table A-2. Scalar String Functions (cont.)

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

**Table A-2. Scalar String Functions** (*cont.*)

Function	Returns
RTRIM( <i>string_exp</i> )	The characters of <i>string_exp</i> with trailing blanks removed.
SOUNDEX( <i>string_exp</i> )	A data-source dependent string representing the sound of the words in <i>string_exp</i> .
SPACE( <i>count</i> )	A string consisting of <i>count</i> spaces.
SUBSTRING( <i>string_exp</i> , <i>start</i> , <i>length</i> )	A string derived from <i>string_exp</i> , beginning at the character position <i>start</i> for <i>length</i> characters.
UCASE( <i>string_exp</i> )	Lowercase characters in <i>string_exp</i> converted to uppercase.

## Numeric Functions

[Table A-3](#) 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-3.    Scalar Numeric Functions**

Function	Returns
<code>ABS(<i>numeric_exp</i>)</code>	Absolute value of <i>numeric_exp</i> .
<code>ACOS(<i>float_exp</i>)</code>	Arccosine of <i>float_exp</i> as an angle in radians.
<code>ASIN(<i>float_exp</i>)</code>	Arcsine of <i>float_exp</i> as an angle in radians.
<code>ATAN(<i>float_exp</i>)</code>	Arctangent of <i>float_exp</i> as an angle in radians.
<code>ATAN2(<i>float_exp1</i>, <i>float_exp2</i>)</code>	Arctangent of the x and y coordinates, specified by <i>float_exp1</i> and <i>float_exp2</i> as an angle in radians.
<code>CEILING(<i>numeric_exp</i>)</code>	Smallest integer greater than or equal to <i>numeric_exp</i> .
<code>COS(<i>float_exp</i>)</code>	Cosine of <i>float_exp</i> as an angle in radians.
<code>COT(<i>float_exp</i>)</code>	Cotangent of <i>float_exp</i> as an angle in radians.
<code>DEGREES(<i>numeric_exp</i>)</code>	Number if degrees converted from <i>numeric_exp</i> radians.
<code>EXP(<i>float_exp</i>)</code>	Exponential value of <i>float_exp</i> .
<code>FLOOR(<i>numeric_exp</i>)</code>	Largest integer less than or equal to <i>numeric_exp</i> .
<code>LOG(<i>float_exp</i>)</code>	Natural log of <i>float_exp</i> .
<code>LOG10(<i>float_exp</i>)</code>	Base 10 log of <i>float_exp</i> .
<code>MOD(<i>integer_exp1</i>, <i>integer_exp2</i>)</code>	Remainder of <i>integer_exp1</i> divided by <i>integer_exp2</i> .
<code>PI()</code>	Constant value of pi as a floating-point number.
<code>POWER(<i>numeric_exp</i>, <i>integer_exp</i>)</code>	Value of <i>numeric_exp</i> to the power of <i>integer_exp</i> .
<code>RADIANS(<i>numeric_exp</i>)</code>	Number of radians converted from <i>numeric_exp</i> degrees.
<code>RAND([<i>integer_exp</i>])</code>	Random floating-point value using <i>integer_exp</i> as the optional seed value.
<code>ROUND(<i>numeric_exp</i>, <i>integer_exp</i>)</code>	<i>numeric_exp</i> rounded to <i>integer_exp</i> places right of the decimal (left of the decimal if <i>integer_exp</i> is negative).
<code>SIGN(<i>numeric_exp</i>)</code>	Indicator of the sign of <i>numeric_exp</i> . If <i>numeric_exp</i> < 0, -1 is returned. If <i>numeric_exp</i> = 0, 0 is returned. If <i>numeric_exp</i> > 0, 1 is returned.

**Table A-3. Scalar Numeric Functions** (cont.)

Function	Returns
<code>SIN(float_exp)</code>	Sine of <i>float_exp</i> , where <i>float_exp</i> is an angle in radians.
<code>SQRT(float_exp)</code>	Square root of <i>float_exp</i> .
<code>TAN(float_exp)</code>	Tangent of <i>float_exp</i> , where <i>float_exp</i> is an angle in radians.
<code>TRUNCATE(numeric_exp, integer_exp)</code>	<i>numeric_exp</i> truncated to <i>integer_exp</i> places right of the decimal. (If <i>integer_exp</i> is negative, truncation is to the left of the decimal.)

## Date and Time Functions

[Table A-4](#) 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-4. Scalar Time and Date Functions

Function	Returns
CURDATE()	Current date as a date value.
CURRENT_DATE()	Current date.
CURRENT_TIME[( <i>time-precision</i> )]	Current local time. The <i>time-precision</i> argument determines the seconds precision of the returned value.
CURRENT_TIMESTAMP[( <i>timestamp-precision</i> )]	Current local date and local time as a timestamp value. The <i>timestamp-precision</i> argument determines the seconds precision of the returned timestamp.
CURTIME()	Current local time as a time value.
DAYNAME( <i>date_exp</i> )	Character string containing a data-source-specific name of the day for the day portion of <i>date_exp</i> .
DAYOFMONTH( <i>date_exp</i> )	Day of the month in <i>date_exp</i> as an integer value (1–31).
DAYOFWEEK( <i>date_exp</i> )	Day of the week in <i>date_exp</i> as an integer value (1–7).
DAYOFYEAR( <i>date_exp</i> )	Day of the year in <i>date_exp</i> as an integer value (1–366).
EXTRACT( <i>extract-field</i> FROM <i>extract-source</i> )	<p><i>Extract-field</i> portion of the <i>extract-source</i>. The <i>extract-source</i> argument is a datetime or interval expression. The <i>extract-field</i> argument can be one of the following keywords:</p> <p>YEAR MONTH DAY HOUR MINUTE SECOND</p> <p>The precision scale of the returned value is 0 unless SECOND is specified, in which case, the scale is not less than the fractional seconds precision of the <i>extract-source</i> field.</p>
HOUR( <i>time_exp</i> )	Hour in <i>time_exp</i> as an integer value (0–23).

**Table A-4. Scalar Time and Date Functions** (cont.)

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

## System Functions

Table A-5 lists system functions.

Table A-5. Scalar System Functions	
Function	Returns
DATABASE()	Name of the database, corresponding to the connection handle (hdbc).
IFNULL( <i>exp,value</i> )	<i>value</i> , if <i>exp</i> is null.
USER()	Authorization name of the user.

## Like Predicate Escape Characters

In a LIKE predicate, the percent sign (%) matches zero or more of any character and the underscore (\_) matches any one character. To match an actual percent sign or underscore in a LIKE predicate, an escape character must precede the % or \_. The escape sequence that defines the LIKE predicate escape character is:

{escape 'escape-character' }

where *escape-character* is any character supported by the data source.

**Example:**

```
SELECT Name FROM Customers
WHERE Name LIKE '\%AAA%' {escape '\' }
```

returns all the customers for which the name 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-6](#) lists the outer join escape sequences supported by SequeLink for each data store.

**Table A-6. Outer Join Escape Sequences Supported by the SequeLink ODBC Driver and SequeLink JDBC Driver**

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 V6R1, V7R1, V7R2 on Windows NT, Windows 2000, 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
Legacy Server	Left outer joins Right outer joins Full outer joins Nested outer joins Unordered outer joins
Microsoft SQL Server	Left outer joins Right outer joins Full outer joins Nested outer joins
Oracle8i	Left outer joins Right outer joins Nested outer joins
Oracle9i	Left outer joins Right outer joins Full outer joins Nested outer joins Unordered outer joins Inner outer joins
Sybase	Left outer joins Nested outer joins Unordered outer joins

---

## Procedure Call Escape Sequences

A procedure is an executable object stored in the data store. Generally, it is one or more SQL statements that have been precompiled. The escape sequence for calling a procedure is:

```
{[?]=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

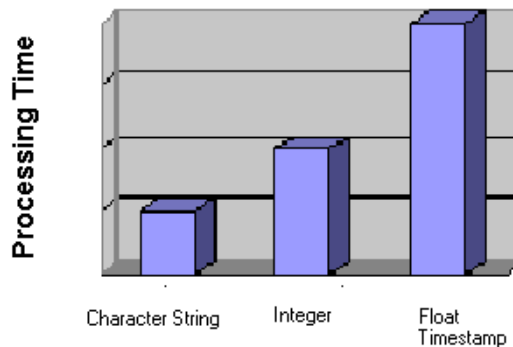
This appendix lists the data types and isolation levels supported for each data store supported by SequeLink.

---

### 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.

This section lists the ODBC, ADO/OLE DB, and JDBC data types that SequeLink supports for each data store.

## DB2 V5 on OS/390

[Table B-1](#) lists the data types supported by the SequeLink ODBC driver, SequeLink ADO provider, and SequeLink JDBC driver for DB2 V5.

<i>Table B-1. Data Types (DB2 V5)</i>			
DB2 Data Type	ODBC Data Type	OLE DB Data Type	JDBC Data Type
Char	SQL_CHAR	DBTYPE_STR	CHAR
Char() for Bit Data	SQL_BINARY	DBTYPE_BYTES	BINARY
Date	SQL__TYPE_DATE	DBTYPE_DBDATE	DATE
Decimal	SQL_DECIMAL	DBTYPE_NUMERIC	DECIMAL
Float	SQL_FLOAT	DBTYPE_R8	FLOAT
Integer	SQL_INTEGER	DBTYPE_I4	INTEGER
Long Varchar	SQL_LONGVARCHAR	DBTYPE_STR	LONGVARCHAR
Long Varchar for Bit Data	SQL_LONGVARBINARY	DBTYPE_BYTES	LONGVARBINARY
Real	SQL_REAL	DBTYPE_R4	REAL
Smallint	SQL_SMALLINT	DBTYPE_I2	SMALLINT
Time	SQL_TYPE_TIME	DBTYPE_DBTIME	TIME
Timestamp	SQL_TYPE_TIMESTAMP	DBTYPE_DBTIMESTAMP	TIMESTAMP
Varchar	SQL_VARCHAR	DBTYPE_STR	VARCHAR
Varchar() for Bit Data	SQL_VARBINARY	DBTYPE_BYTES	VARBINARY

## DB2 V6, V7

Table B-2 lists the data types supported by the SequeLink ODBC driver, SequeLink ADO provider, and SequeLink JDBC driver for DB2 V6 and V7.

**Table B-2. Data Types (DB2 V6, V7)**

DB2 Data Type	ODBC Data Type	OLE DB Data Type	JDBC Data Type
Blob*	SQL_LONGVARBINARY	DBTYPE_BYTES	BLOB
Char	SQL_CHAR	DBTYPE_STR	CHAR
Char() for Bit Data	SQL_BINARY	DBTYPE_BYTES	BINARY
Clob*	SQL_LONGVARCHAR	DBTYPE_STR	CLOB
Date	SQL__TYPE_DATE	DBTYPE_DBDATE	DATE
Decimal	SQL_DECIMAL	DBTYPE_NUMERIC	DECIMAL
Float	SQL_FLOAT	DBTYPE_R8	FLOAT
Integer	SQL_INTEGER	DBTYPE_I4	INTEGER
Long Varchar	SQL_LONGVARCHAR	DBTYPE_STR	LONGVARCHAR
Long Varchar for Bit Data	SQL_LONGVARBINARY	DBTYPE_BYTES	LONGVARBINARY*
Real	SQL_REAL	DBTYPE_R4	REAL
Rowid	SQL_VARBINARY	DBTYPE_BYTES	VARBINARY
Smallint	SQL_SMALLINT	DBTYPE_I2	SMALLINT
Time	SQL_TYPE_TIME	DBTYPE_DBTIME	TIME
Timestamp	SQL_TYPE_TIMESTAMP	DBTYPE_DBTIMESTAMP	TIMESTAMP
Varchar	SQL_VARCHAR	DBTYPE_STR	VARCHAR
Varchar() for Bit Data	SQL_VARBINARY	DBTYPE_BYTES	VARBINARY

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

# Informix 9

Table B-3 lists the data types supported by the SequeLink ODBC driver, SequeLink ADO provider, and SequeLink JDBC driver for Informix 9.

**Table B-3. Data Types (Informix 9)**

Informix Data Type	ODBC Data Type	OLE DB Data Type	JDBC Data Type
Blob	SQL_LONGVARIABLE	DBTYPE_BYTES	LONGVARIABLE
Boolean	SQL_BIT	DBTYPE_BOOL	BIT
Byte	SQL_LONGVARIABLE	DBTYPE_BYTES	LONGVARIABLE
Char	SQL_CHAR	DBTYPE_STR	CHAR
Clob	SQL_LONGVARIABLE	DBTYPE_STR	LONGVARIABLE
Date	SQL__TYPE_DATE	DBTYPE_DBDATE	DATE
Datetime hour to second	SQL__TYPE_TIME	DBTYPE_DBTIME	TIME
Datetime year to fraction(5)	SQL__TYPE_TIMESTAMP	DBTYPE_DBTIMESTAMP	TIMESTAMP
Decimal	SQL_DECIMAL	DBTYPE_NUMERIC	DECIMAL
Float	SQL_FLOAT	DBTYPE_R8	FLOAT
Integer	SQL_INTEGER	DBTYPE_I4	INTEGER
Int8	SQL_BIGINT	DBTYPE_I8	BIGINT
Lvarchar	SQL_LONGVARIABLE	DBTYPE_STR	LONGVARIABLE
Money	SQL_DECIMAL	DBTYPE_NUMERIC	DECIMAL
Numeric	SQL_NUMERIC	DBTYPE_NUMERIC	NUMERIC
Serial	SQL_INTEGER	DBTYPE_I4	INTEGER
Serial8	SQL_BIGINT	DBTYPE_I8	BIGINT
Smallfloat	SQL_REAL	DBTYPE_R4	REAL
Smallint	SQL_SMALLINT	DBTYPE_I2	SMALLINT

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



**Table B-3. Data Types (Informix 9) (cont.)**

Informix Data Type	ODBC Data Type	OLE DB Data Type	JDBC Data Type
Text	SQL_LONGVARCHAR	DBTYPE_STR	LONGVARCHAR
Varchar	SQL_VARCHAR	DBTYPE_STR	VARCHAR

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

## Microsoft SQL Server 7

[Table B-4](#) lists the data types supported by the SequeLink ODBC driver, SequeLink ADO provider, and SequeLink JDBC driver for Microsoft SQL Server 7.

**Table B-4. Data Types (Microsoft SQL Server 7)**

Microsoft SQL Server Data Type	ODBC Data Type	OLE DB Data Type	JDBC Data Type
Binary	SQL_BINARY	DBTYPE_BYTES	BINARY
Bit	SQL_BIT	DBTYPE_BOOL	BIT
Char	SQL_CHAR	DBTYPE_STR	CHAR
Datetime	SQL_DATETIME	DBTYPE_DBTIMESTAMP	TIMESTAMP
Decimal	SQL_DECIMAL	DBTYPE_NUMERIC	DECIMAL
Decimal identity	SQL_DECIMAL	DBTYPE_NUMERIC	DECIMAL
Float	SQL_FLOAT	DBTYPE_R8	FLOAT
Guid	SQL_BINARY	DBTYPE_GUID	BINARY

\* Wide characters are not fully supported. Instead, they are mapped to their single-byte character set equivalent.

**Table B-4. Data Types (Microsoft SQL Server 7) (cont.)**

Microsoft SQL Server Data Type	ODBC Data Type	OLE DB Data Type	JDBC Data Type
Image	SQL_LONGVARBINARY	DBTYPE_BYTES	LONGVARBINARY
Int	SQL_INTEGER	DBTYPE_I4	INTEGER
Int identity	SQL_INTEGER	DBTYPE_I4	INTEGER
Money	SQL_DECIMAL	DBTYPE_CY	DECIMAL
Nchar*	SQL_CHAR	DBTYPE_WSTR	CHAR
Ntext*	SQL_LONGVARCHAR	DBTYPE_WSTR	LONGVARCHAR
Numeric	SQL_NUMERIC	DBTYPE_NUMERIC	NUMERIC
Numeric identity	SQL_NUMERIC	DBTYPE_NUMERIC	NUMERIC
Nvarchar*	SQL_VARCHAR	DBTYPE_STR	VARCHAR
Real	SQL_REAL	DBTYPE_R4	REAL
Smalldatetime	SQL_DATETIME	DBTYPE_DBTIMESTAMP	TIMESTAMP
Smallint	SQL_SMALLINT	DBTYPE_I2	SMALLINT
Smallint identity	SQL_SMALLINT	DBTYPE_I2	SMALLINT
Smallmoney	SQL_DECIMAL	DBTYPE_CY	DECIMAL
Sysname	SQL_VARCHAR	DBTYPE_WSTR	VARCHAR
Text	SQL_LONGVARCHAR	DBTYPE_STR	LONGVARCHAR
Timestamp	SQL_BINARY	DBTYPE_BYTES	BINARY
Tinyint	SQL_TINYINT	DBTYPE_UI1	TINYINT
Tinyint identity	SQL_TINYINT	DBTYPE_UI1	TINYINT
Varbinary	SQL_VARBINARY	DBTYPE_BYTES	VARBINARY
Varchar	SQL_VARCHAR	DBTYPE_STR	VARCHAR

\* Wide characters are not fully supported. Instead, they are mapped to their single-byte character set equivalent.

## Microsoft SQL Server2000

Table B-5 lists the data types supported by the SequeLink ODBC driver, SequeLink ADO provider, and SequeLink JDBC driver for Microsoft SQL Server2000.

**Table B-5. Data Types (Microsoft SQL Server2000)**

Microsoft SQL Server Data Type	ODBC Data Type	OLE DB Data Type	JDBC Data Type
Bigint	SQL_BIGINT	DBTYPE_I8	BIGINT
Bigint identity	SQL_BIGINT	DBTYPE_I8	BIGINT
Binary	SQL_BINARY	DBTYPE_BYTES	BINARY
Bit	SQL_BIT	DBTYPE_BOOL	BIT
Char	SQL_CHAR	DBTYPE_STR	CHAR
Datetime	SQL_DATETIME	DBTYPE_DBTIMESTAMP	TIMESTAMP
Decimal	SQL_DECIMAL	DBTYPE_NUMERIC	DECIMAL
Decimal identity	SQL_DECIMAL	DBTYPE_NUMERIC	DECIMAL
Float	SQL_FLOAT	DBTYPE_R8	FLOAT
Image	SQL_LONGVARBINARY	DBTYPE_BYTES	LONGVARBINARY
Int	SQL_INTEGER	DBTYPE_I4	INTEGER
Int identity	SQL_INTEGER	DBTYPE_I4	INTEGER
Money	SQL_DECIMAL	DBTYPE_CY	DECIMAL
Nchar	SQL_CHAR	DBTYPE_WSTR	CHAR
Ntext*	SQL_LONGVARCHAR	DBTYPE_STR	LONGVARCHAR
Numeric	SQL_NUMERIC	DBTYPE_NUMERIC	NUMERIC
Numeric() identity	SQL_NUMERIC	DBTYPE_NUMERIC	NUMERIC
Nvarchar*	SQL_VARCHAR	DBTYPE_STR	VARCHAR
Real	SQL_REAL	DBTYPE_R4	REAL
Smalldatetime	SQL_DATETIME	DBTYPE_DBTIMESTAMP	TIMESTAMP
Smallint	SQL_SMALLINT	DBTYPE_I2	SMALLINT
Smallint identity	SQL_SMALLINT	DBTYPE_I2	SMALLINT
Smallmoney	SQL_DECIMAL	DBTYPE_CY	DECIMAL

**Table B-5. Data Types (Microsoft SQL Server2000)** *(cont.)*

Microsoft SQL Server Data Type	ODBC Data Type	OLE DB Data Type	JDBC Data Type
SQLVariant	SQL_BINARY		
Sysname	SQL_VARCHAR	DBTYPE_STR	VARCHAR
Text	SQL_LONGVARCHAR	DBTYPE_STR	LONGVARCHAR
Timestamp	SQL_BINARY	DBTYPE_BYTES	BINARY
Tinyint	SQL_TINYINT	DBTYPE_UI1	TINYINT
Tinyint identity	SQL_TINYINT	DBTYPE_UI1	TINYINT
Uniqueidentifier	SQL_BINARY	DBTYPE_BINARY	BINARY
Varbinary	SQL_VARBINARY	DBTYPE_BYTES	VARBINARY
Varchar	SQL_VARCHAR	DBTYPE_STR	VARCHAR

\* Wide characters are not fully supported. Instead, they are mapped to their single-byte character set equivalent.

## Oracle8

Table B-6 lists the data types supported by the SequeLink ODBC driver, SequeLink ADO provider, and SequeLink JDBC driver for Oracle8.

**Table B-6. Data Types (Oracle8)**

Oracle Data Type	ODBC Data Type	OLE DB Data Type	JDBC Data Type
Bfile	SQL_LONGVARBINARY	DBTYPE_BYTES	LONGVARBINARY
Blob	SQL_LONGVARBINARY	DBTYPE_BYTES	BLOB***
Char	SQL_CHAR	DBTYPE_STR	CHAR
Clob	SQL_LONGVARCHAR	DBTYPE_STR	CLOB ***
Date	SQL_TYPE_TIMESTAMP	DBTYPE_DBTIMESTAMP	TIMESTAMP
Long	SQL_LONGVARCHAR	DBTYPE_STR	LONGVARCHAR

**Table B-6. Data Types (Oracle8) (cont.)**

Oracle Data Type	ODBC Data Type	OLE DB Data Type	JDBC Data Type
Long Raw	SQL_LONGVARBINARY	DBTYPE_BYTES	LONGVARBINARY
Nchar(size)*	SQL_CHAR	DBTYPE_STR	CHAR
Nclob*	SQL_LONGVARCHAR	**	LONGVARCHAR
Number	SQL_FLOAT	DBTYPE_R8	FLOAT
Number(p,s)	SQL_DECIMAL	DBTYPE_NUMERIC	DECIMAL
NVarchar2	SQL_VARCHAR	**	VARCHAR
Raw	SQL_VARBINARY	DBTYPE_BYTES	VARBINARY
Rowid	SQL_VARCHAR	DBTYPE_STR	VARCHAR
Varchar2	SQL_VARCHAR	DBTYPE_STR	VARCHAR

\* Nchar, Nclob, and Nvarchar2 are partially supported.

\*\* No corresponding data type.

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

## Oracle9i

[Table B-7](#) lists the data types supported by the SequeLink ODBC driver, SequeLink ADO provider, and SequeLink JDBC driver for Oracle9i.

**Table B-7. Data Types (Oracle9i)**

Oracle Data Type	ODBC Data Type	OLE DB Data Type	JDBC Data Type
Bfile	SQL_LONGVARBINARY	DBTYPE_BYTES	LONGVARBINARY
Blob	SQL_LONGVARBINARY	DBTYPE_BYTES	BLOB
Char	SQL_CHAR	DBTYPE_STR	CHAR

**Table B-7. Data Types (Oracle9i) (cont.)**

Oracle Data Type	ODBC Data Type	OLE DB Data Type	JDBC Data Type
Clob	SQL_LONGVARCHAR	DBTYPE_STR	CLOB
Date	SQL_TYPE_TIMESTAMP	DBTYPE_DBTIMESTAMP	TIMESTAMP
Long	SQL_LONGVARCHAR	DBTYPE_STR	LONGVARCHAR
Long Raw	SQL_LONGVARBINARY	DBTYPE_BYTES	LONGVARBINARY
Nchar(size) Note 1	SQL_CHAR	DBTYPE_STR	CHAR
Nclob Note 1	SQL_LONGVARCHAR	Note 2	LONGVARCHAR
Number	SQL_FLOAT	DBTYPE_R8	FLOAT
Number(p,s)	SQL_DECIMAL	DBTYPE_NUMERIC	DECIMAL
Nvarchar2 Note 1	SQL_VARCHAR	Note 2	VARCHAR
Raw	SQL_VARBINARY	DBTYPE_BYTES	VARBINARY
Rowid	SQL_VARCHAR	DBTYPE_STR	VARCHAR
timestamp	SQL_TYPE_TIMESTAMP	DBTYPE_TIMESTAMP	TIMESTAMP
timestamp with local time zone	SQL_TYPE_TIMESTAMP	DBTYPE_TIMESTAMP	TIMESTAMP
timestamp with time zone Note 3	SQL_TYPE_TIMESTAMP	DBTYPE_TIMESTAMP	TIMESTAMP
Varchar2	SQL_VARCHAR	DBTYPE_STR	VARCHAR

NOTES:

- 1 Nchar, Nclob, and Nvarchar2 are partially supported.
- 2 No corresponding data type.
- 3 The value will not contain the time zone. Instead, it will be changed to the client time zone.
- 4 The LOB data types will be handled in JDBC as LOB objects or as Longvar data types, depending on the setting for the DataSourceReportLobsAsLongvar service attribute. Refer to the *SequeLink Administrator's Guide* for more information

## Sybase

[Table B-8](#) lists the data types supported by the SequeLink ODBC driver, SequeLink ADO provider, and SequeLink JDBC driver for Sybase 11 and 12.

**Table B-8. Data Types (Sybase 11 and 12)**

Sybase Data Type	ODBC Data Type	OLE DB Data Type	JDBC Data Type
Binary	SQL_BINARY	DBTYPE_BYTES	BINARY
Bit	SQL_BIT	DBTYPE_BOOL	BIT
Char	SQL_CHAR	DBTYPE_STR	CHAR
Datetime	SQL__TYPE_DATETIME	DBTYPE_DBTIMESTAMP	TIMESTAMP
Decimal	SQL_DECIMAL	DBTYPE_NUMERIC	DECIMAL
Float	SQL_FLOAT	DBTYPE_R8	FLOAT
Image	SQL_LONGVARBINARY	DBTYPE_BYTES	LONGVARBINARY
Int	SQL_INTEGER	DBTYPE_I4	INTEGER
Money	SQL_DECIMAL	DBTYPE_CY	DECIMAL
Numeric	SQL_NUMERIC	DBTYPE_NUMERIC	NUMERIC
Real	SQL_REAL	DBTYPE_R4	REAL
Smalldatetime	SQL__TYPE_DATETIME	DBTYPE_DBTIMESTAMP	TIMESTAMP
Smallint	SQL_SMALLINT	DBTYPE_I2	SMALLINT
Smallmoney	SQL_DECIMAL	DBTYPE_CY	DECIMAL
Sysname	SQL_VARCHAR	DBTYPE_STR	VARCHAR
Text	SQL_LONGVARCHAR	DBTYPE_STR	LONGVARCHAR
Timestamp	SQL_BINARY	DBTYPE_BYTES	BINARY
Tinyint	SQL_TINYINT	DBTYPE_UI1	TINYINT
Varbinary	SQL_VARBINARY	DBTYPE_BYTES	VARBINARY
Varchar	SQL_VARCHAR	DBTYPE_STR	VARCHAR

## SequeLink Legacy Server

[Table B-9](#) lists the data types supported by the SequeLink ODBC driver, SequeLink ADO provider, and SequeLink JDBC driver for Legacy Server.

**Table B-9. Data Types (Legacy Server)**

Legacy Server Data Type	ODBC Data Type	OLE DB Data Type	JDBC Data Type
Char(size)	SQL_CHAR	DBTYPE_STR	CHAR
Decimal	SQL_DECIMAL	DBTYPE_DECIMAL	DECIMAL
Float (n)22<=n<=53	SQL_DOUBLE	DBTYPE_R8	DOUBLE
Float(n)n<=22	SQL_REAL	DBTYPE_R4	REAL
Integer	SQL_INTEGER	DBTYPE_I4	INTEGER
Long varchar	SQL_LONGVARCHAR	DBTYPE_STR	SQL_LONGVARCHAR
Smallint	SQL_SMALLINT	DBTYPE_I2	SMALLINT
Varchar(size)	SQL_VARCHAR	DBTYPE_STR	VARCHAR

## Isolation Levels

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



**Table B-10. Isolation Levels**

<b>Database</b>	<b>Isolation Levels</b>	<b>Default</b>
DB2 V5, V6, V7	Read uncommitted Read committed Repeatable read Serializable	Read committed
Informix	Read uncommitted Read committed Repeatable read	Read committed
Legacy Server	Read uncommitted	Read uncommitted
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

[Table C-1](#) shows compatibility among the JDBC specification versions, Java Virtual Machines, and theSequeLink JDBC driver. TheSequeLink JDBC driver does not support Java Virtual Machines 1.0.2 or 1.1.8.

---

***Table C-1.    JDBC Compatibility***

---

JDBC Version*	JDK	Driver Compatible?
1.22	1.2	Yes
2.0	1.2	Yes
2.0	1.3	Yes
3.0	1.2	No
3.0	1.3	No
3.0	1.4	Yes

---

\*Refers to whether the application is using JDBC 1.22, JDBC 2.0 or JDBC 3.0 features.

---

# 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
java.io.InputStream getBinaryStream ()	2.0 Core	Yes	Not supported in DB2 OS/390 V5, SQL Server 2000, Sybase, Informix, and Legacy drivers.
byte[] getBytes (long, int)	2.0 Core	Yes	Not supported in DB2 OS/390 V5, SQL Server 2000, Sybase, Informix, and Legacy drivers.
long length ()	2.0 Core	Yes	Not supported in DB2 OS/390 V5, SQL Server 2000, Sybase, Informix, and Legacy drivers.
long position (byte, long)	2.0 Core	Yes	Not supported in DB2 OS/390 V5, SQL Server 2000, Sybase, Informix, and Legacy drivers.
long position (Blob, long)	2.0 Core	Yes	Not supported in DB2 OS/390 V5, SQL Server 2000, Sybase, Informix, and Legacy drivers.
java.io.OutputStream setBinaryStream (long)	3.0 Core	No	Throws "unsupported method" exception.
int setBytes (long, byte[])	3.0 Core	No	Throws "unsupported method" exception.
int setBytes (long, byte[], int, int)	3.0 Core	No	Throws "unsupported method" exception.

<b>Blob Object</b> <i>(cont.)</i> <b>Methods</b>	<b>Version Introduced</b>	<b>Supported</b>	<b>Comments</b>
void truncate (long)	3.0 Core	No	Throws "unsupported method" exception.

<b>CallableStatement Object Methods</b>	<b>Version Introduced</b>	<b>Supported</b>	<b>Comments</b>
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.
boolean execute (String, String[])	3.0	No	Throws "invalid method call" exception.

<b>CallableStatement Object</b> <i>(cont.)</i>	<b>Version Introduced</b>	<b>Supported</b>	<b>Comments</b>
<b>Methods</b>			
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 “invalid method call” exception.
int executeUpdate (String, int[])	3.0	No	Throws “invalid method call” exception.
int executeUpdate (String, String[])	3.0	No	Throws “invalid method call” exception.
Array getArray (int)	2.0 Core	No	Throws “unsupported method” exception.
Array getArray (String)	3.0	No	Throws “unsupported method” exception.
java.math.BigDecimal getBigDecimal (int)	2.0 Core	Yes	
java.math.BigDecimal getBigDecimal (int, int)	1.0	Yes	
java.math.BigDecimal getBigDecimal (String)	3.0	No	Throws “unsupported method” exception.
Blob getBlob (int)	2.0 Core	Yes	Not supported in DB2 OS/390 V5, SQL Server 2000, Sybase, Informix, and Legacy drivers.
Blob getBlob (String)	3.0	No	Throws “unsupported method” exception.
boolean getBoolean (int)	1.0	Yes	

<b>CallableStatement</b>			
<b>Object</b> <i>(cont.)</i>	<b>Version</b>		
<b>Methods</b>	<b>Introduced</b>	<b>Supported</b>	<b>Comments</b>
boolean getBoolean (String)	3.0	No	Throws “unsupported method” exception.
byte getByte (int)	1.0	Yes	
byte getByte (String)	3.0	No	Throws “unsupported method” exception.
byte [] getBytes (int)	1.0	Yes	
byte [] getBytes (String)	3.0	No	Throws “unsupported method” exception.
Clob getClob (int)	2.0 Core	Yes	
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, java.util.Calendar)	2.0 Core	Yes	
Date getDate (String)	3.0	No	Throws “unsupported method” exception.
Date getDate (String, java.util.Calendar)	3.0	No	Throws “unsupported method” exception.
double getDouble (int)	1.0	Yes	
double getDouble (String)	3.0	No	Throws “unsupported method” exception.
int getFetchDirection ()	2.0 Core	Yes	
int getFetchSize ()	2.0 Core	Yes	
float getFloat (int)	1.0	Yes	
float getFloat (String)	3.0	No	Throws “unsupported method” exception.

<b>CallableStatement</b>			
<b>Object</b> <i>(cont.)</i>	<b>Version</b>	<b>Supported</b>	<b>Comments</b>
<b>Methods</b>	<b>Introduced</b>		
ResultSet getGeneratedKeys()	3.0	No	Throws “unsupported method” exception.
int getInt (int)	1.0	Yes	
int getInt (String)	3.0	No	Throws “unsupported method” exception.
long getLong (int)	1.0	Yes	
long getLong (String)	3.0	No	Throws “unsupported method” exception.
int getMaxFieldSize ()	1.0	Yes	
int getMaxRows ()	1.0	Yes	
ResultSetMetaData getMetaData ()	2.0 Core	Yes	
boolean getMoreResults ()	1.0	Yes	
boolean getMoreResults (int)	3.0	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.
int getQueryTimeout ()	1.0	Yes	Returns 0 for DB2 and Informix.
Ref getRef (int)	2.0 Core	No	Throws “unsupported method” exception.
Ref getRef (String)	3.0	No	Throws “unsupported method” exception.
ResultSet getResultSet ()	1.0	Yes	



<b>CallableStatement Object</b> <i>(cont.)</i>	<b>Version Introduced</b>	<b>Supported</b>	<b>Comments</b>
<b>Methods</b>			
int getResultSetConcurrency ()	2.0 Core	Yes	
int getResultSetHoldability ()	3.0	No	Throws “unsupported method” exception.
int getResultSetType ()	2.0 Core	Yes	
short getShort (int)	1.0	Yes	
short getShort (String)	3.0	No	Throws “unsupported method” exception.
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, java.util.Calendar)	2.0 Core	Yes	
Time getTime (String)	3.0	No	Throws “unsupported method” exception.
Time getTime (String, java.util.Calendar)	3.0	No	Throws “unsupported method” exception.
Timestamp getTimestamp (int)	1.0	Yes	
Timestamp getTimestamp (int, java.util.Calendar)	2.0 Core	Yes	
Timestamp getTimestamp (String)	3.0	No	Throws “unsupported method” exception.
Timestamp getTimestamp (String, java.util.Calendar)	3.0	No	Throws “unsupported method” exception.
int getUpdateCount ()	1.0	Yes	
java.net.URL getURL (int)	3.0	No	Throws “unsupported method” exception.

<b>CallableStatement</b>			
<b>Object</b> <i>(cont.)</i>	<b>Version</b>		
<b>Methods</b>	<b>Introduced</b>	<b>Supported</b>	<b>Comments</b>
java.net.URL getURL (String)	3.0	No	Throws “unsupported method” exception.
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, 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 (int, int, int)	1.0	Yes	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, String)	3.0	No	Throws “unsupported method” exception.
void registerOutParameter (String, int, int)	3.0	No	Throws “unsupported method” exception.
SQLWarning getWarnings ()	1.0	Yes	
void setArray (int, Array)	2.0 Core	No	Throws “unsupported method” exception.
void setAsciiStream (int, java.io.InputStream, int)	1.0	Yes	
void setAsciiStream (String, InputStream, int)	3.0	No	Throws “unsupported method” exception.
void setBigDecimal (int, java.math.BigDecimal)	1.0	Yes	

<b>CallableStatement</b>			
<b>Object</b> <i>(cont.)</i>	<b>Version</b>		
<b>Methods</b>	<b>Introduced</b>	<b>Supported</b>	<b>Comments</b>
void setBigDecimal (String, java.math.BigDecimal)	3.0	No	Throws "unsupported method" exception.
void setBinaryStream (int, java.io.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	Not supported in DB2 OS/390 V5, SQL Server 2000, Sybase, Informix, and Legacy drivers.
void setBoolean (int, boolean)	1.0	Yes	
void setBoolean (String, boolean)	3.0	No	Throws "unsupported method" exception.
void setByte (int, byte)	1.0	Yes	
void setByte (String, byte)	3.0	No	Throws "unsupported method" exception.
void setBytes (int, byte [])	1.0	Yes	
void setBytes (String, byte [])	3.0	No	Throws "unsupported method" exception.
void setCharacterStream (int, java.io.Reader, int)	2.0 Core	Yes	
void setCharacterStream (String, java.io.Reader, int)	3.0	No	Throws "unsupported method" exception.
void setClob (int, Clob)	2.0 Core	Yes	Not supported in DB2 OS/390 V5, SQL Server 2000, Sybase, Informix, and Legacy drivers.
void setCursorName (String)	1.0	No	Throws "unsupported method" exception.
void setDate (int, Date)	1.0	Yes	SQLServer throws "optional feature not implemented" exception.

<b>CallableStatement</b>			
<b>Object</b> <i>(cont.)</i>	<b>Version</b>		
<b>Methods</b>	<b>Introduced</b>	<b>Supported</b>	<b>Comments</b>
void setDate (String, Date)	3.0	No	Throws "unsupported method" exception.
void setDate (int, Date, java.util.Calendar)	2.0 Core	Yes	SQLServer throws "optional feature not implemented" exception.
void setDate (String, Date, java.util.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	
void setFetchDirection (int)	2.0 Core	Yes	SequeLink supports this method, but will simply ignore this hint provided by the application.
void setFetchSize (int)	2.0 Core	Yes	SequeLink supports this method, but will simply 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.

<b>CallableStatement Object</b> <i>(cont.)</i>			
<b>Methods</b>	<b>Version Introduced</b>	<b>Supported</b>	<b>Comments</b>
void setLong (int, long)	1.0	Yes	SQLServer 7 throws "optional feature not implemented" exception. However, there is a workaround available 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 (String, int)	3.0	No	Throws "unsupported method" exception.
void setNull (int, int, String)	2.0 Core	Yes	SequeLink does not support the Advanced Data Type functionality. This method is always mapped to setNull (i, sqlType).
void setNull (String, int, String)	3.0	No	Throws "unsupported method" exception.
void setObject (int, Object)	1.0	Yes	
void setObject (String, Object)	3.0	No	Throws "unsupported method" exception.
void setObject (int, Object, int)	1.0	Yes	
void setObject (String, Object, int)	3.0	No	Throws "unsupported method" exception.
void setObject (int, Object, int, int)	1.0	Yes	

<b>CallableStatement</b>			
<b>Object</b> <i>(cont.)</i>	<b>Version</b>	<b>Supported</b>	<b>Comments</b>
<b>Methods</b>	<b>Introduced</b>		
void setObject (String, Object, int, int)	3.0	No	Throws "unsupported method" exception.
void setQueryTimeout (int)	1.0	Yes	Not supported on DB2 OS/390 and Legacy 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.
void setTime (String, Time)	3.0	No	Throws "unsupported method" exception.
void setTime (int, Time, java.util.Calendar)	2.0 Core	Yes	SQLServer throws "optional feature not implemented" exception.
void setTime (String, Time, java.util.Calendar)	3.0	No	Throws "unsupported method" exception.
void setTimestamp (int, Timestamp)	1.0	Yes	
void setTimestamp (String, Timestamp)	3.0	No	Throws "unsupported method" exception.

<b>CallableStatement Object</b> <i>(cont.)</i>	<b>Version Introduced</b>	<b>Supported</b>	<b>Comments</b>
<b>Methods</b>			
void setTimestamp (int, Timestamp, java.util.Calendar)	2.0 Core	Yes	
void setTimestamp (String, Timestamp, java.util.Calendar)	3.0	No	Throws “unsupported method” exception.
void setUnicodeStream (int, java.io.InputStream, int)	1.0	Yes	Supported by SequeLink. This method has been deprecated in the JDBC 3.0 specification.
void setURL (int, java.net.URL)	3.0	No	Throws “unsupported method” exception.
void setURL (String, URL)	3.0	No	Throws “unsupported method” exception.
boolean wasNull ()	1.0	Yes	

<b>Clob Object Methods</b>	<b>Version Introduced</b>	<b>Supported</b>	<b>Comments</b>
java.io.InputStream getAsciiStream ()	2.0 Core	Yes	Not supported in DB2 OS/390 V5, SQL Server 2000, Sybase, Informix, and Legacy drivers.
java.io.Reader getCharacterStream ()	2.0 Core	Yes	Not supported in DB2 OS/390 V5, SQL Server 2000, Sybase, Informix, and Legacy drivers.
String getSubString (long, int)	2.0 Core	Yes	Not supported in DB2 OS/390 V5, SQL Server 2000, Sybase, Informix, and Legacy drivers.
long length ()	2.0 Core	Yes	Not supported in DB2 OS/390 V5, SQL Server 2000, Sybase, Informix, and Legacy drivers.
long position (Clob, long)	2.0 Core	Yes	Not supported in DB2 OS/390 V5, SQL Server 2000, Sybase, Informix, and Legacy drivers.

<b>Clob Object</b> <i>(cont.)</i> <b>Methods</b>	<b>Version Introduced</b>	<b>Supported</b>	<b>Comments</b>
long position (String, long)	2.0 Core	Yes	Not supported in DB2 OS/390 V5, SQL Server 2000, Sybase, Informix, and Legacy drivers.
java.io.OutputStream setAsciiStream (long)	3.0 Core	No	Throws "unsupported method" exception.
java.io.Writer setCharacterStream (long)	3.0 Core	No	Throws "unsupported method" exception.
int setString (long, String)	3.0 Core	No	Throws "unsupported method" exception.
int setString (long, String, int, int)	3.0 Core	No	Throws "unsupported method" exception.
void truncate (long)	3.0 Core	No	Throws "unsupported method" exception.

<b>Connection Object Methods</b>	<b>Version Introduced</b>	<b>Supported</b>	<b>Comments</b>
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	
Statement createStatement ()	1.0	Yes	
Statement createStatement (int, int)	2.0 Core	Yes	The supported values for resultSetConcurrency depend on the data store you are connected to.
Statement createStatement (int, int, int)	3.0	No	SequeLink only supports fixed Holdability values per RDBMS
boolean getAutoCommit ()	1.0	Yes	



<b>Connection Object</b> <i>(cont.)</i> <b>Methods</b>	<b>Version Introduced</b>	<b>Supported</b>	<b>Comments</b>
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	
java.util.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
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.

<b>Connection Object</b> <i>(cont.)</i>	<b>Version</b>	<b>Supported</b>	<b>Comments</b>
<b>Methods</b>	<b>Introduced</b>		
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	
void setSavepoint ()	3.0	Yes	Not supported with DB2 V5 on OS/390, DB2 V5.2 and DB2 V6.1, Informix, and Legacy drivers.
void setSavepoint (String)	3.0	Yes	Not supported with DB2 V5 on OS/390, DB2 V5.2 and DB2 V6.1, Informix, and Legacy drivers.

<b>Connection Object</b> <i>(cont.)</i>	<b>Version</b>		
<b>Methods</b>	<b>Introduced</b>	<b>Supported</b>	<b>Comments</b>
void setTransactionIsolation (int)	1.0	Yes	
void setTypeMap (java.util.Map)	2.0 Core	Yes	Ignored.

<b>ConnectionPoolData</b>			
<b>Source Object</b>	<b>Version</b>		
<b>Methods</b>	<b>Introduced</b>	<b>Supported</b>	<b>Comments</b>
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	

<b>DatabaseMetaData Object</b>	<b>Version</b>		
<b>Methods</b>	<b>Introduced</b>	<b>Supported</b>	<b>Comments</b>
boolean allProceduresAreCallable ()	1.0	Yes	
boolean allTablesAreSelectable ()	1.0	Yes	
boolean dataDefinitionCausesTransaction Commit ()	1.0	Yes	

<b>DatabaseMetaData Object</b> <i>(cont.)</i>	<b>Version</b>	<b>Supported</b>	<b>Comments</b>
<b>Methods</b>	<b>Introduced</b>		
boolean dataDefinitionIgnoredInTransactions ()	1.0	Yes	
boolean deletesAreDetected (int)	2.0 Core	Yes	
boolean doesMaxRowSizeIncludeBlobs ()	1.0	Yes	Not supported in DB2, SQL Server, and Sybase drivers.
ResultSet getAttributes (String, String)	3.0	No	Always returns empty result set.
ResultSet getBestRowIdentifier (String, String, String, int, boolean)	1.0	Yes	The SequeLink 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 SequeLink JDBC driver supports this method if the data store has support for it.
ResultSet getColumns (String, String, String, String)	1.0	Yes	
ResultSet getColumns (String, String, String, String)	3.0 Extension	Yes	The extended resultset columns will all contain null as values.

<b>DatabaseMetaData Object</b> <i>(cont.)</i>	<b>Version</b>	<b>Supported</b>	<b>Comments</b>
<b>Methods</b>	<b>Introduced</b>		
Connection getConnection ()	2.0 Core	Yes	
ResultSet getCrossReference (String, String, String, String, String, String)	1.0	Yes	The SequeLink 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 SequeLink JDBC driver supports this method if the data store has support for it.
String getExtraNameCharacters ()	1.0	Yes	
String getIdentifierQuoteString ()	1.0	Yes	
ResultSet getImportedKeys (String, String, String)	1.0	Yes	The SequeLink JDBC driver supports this method if the data store has support for it.

DatabaseMetaData Object (cont.) Methods	Version Introduced	Supported	Comments
ResultSet getIndexInfo (String, String, String, boolean, boolean)	1.0	Yes	The SequeLink 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	
int getMaxStatementLength ()	1.0	Yes	
int getMaxStatements ()	1.0	Yes	
int getMaxTableNameLength ()	1.0	Yes	

<b>DatabaseMetaData Object</b> <i>(cont.)</i>	<b>Version</b>	<b>Supported</b>	<b>Comments</b>
<b>Methods</b>	<b>Introduced</b>		
int getMaxTablesInSelect ()	1.0	Yes	
int getMaxUserNameLength ()	1.0	Yes	
String getNumericFunctions ()	1.0	Yes	
ResultSet getPrimaryKeys (String, String, String)	1.0	Yes	The SequeLink JDBC driver supports this method if the data store has support for it.
ResultSet getProcedureColumns (String, String, String, String)	1.0	Yes	The SequeLink 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 SequeLink JDBC driver supports this method if the data store has support for it.
int getResultSetHoldability ()	3.0	Yes	
ResultSet getSchemas ()	1.0	Yes	The SequeLink JDBC driver supports this method if the data store has support for it.

<b>DatabaseMetaData Object</b> <i>(cont.)</i>	<b>Version Introduced</b>	<b>Supported</b>	<b>Comments</b>
<b>Methods</b>			
ResultSet getSchemas ()	3.0 Extension	Yes	The SequeLink 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 SequeLink 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	



<b>DatabaseMetaData Object</b> <i>(cont.)</i>	<b>Version</b>	<b>Supported</b>	<b>Comments</b>
<b>Methods</b>	<b>Introduced</b>		
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 SequeLink 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 nullPlusNonNullsIsNull ()	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	

<b>DatabaseMetaData Object</b> <i>(cont.)</i>	<b>Version</b>	<b>Supported</b>	<b>Comments</b>
<b>Methods</b>	<b>Introduced</b>		
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	

<b>DatabaseMetaData Object</b> <i>(cont.)</i>	<b>Version</b>	<b>Supported</b>	<b>Comments</b>
<b>Methods</b>	<b>Introduced</b>		
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	
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	
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	

<b>DatabaseMetaData Object</b> <i>(cont.)</i>	<b>Version</b>	<b>Supported</b>	<b>Comments</b>
<b>Methods</b>	<b>Introduced</b>		
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	The driver returns hard coded false.  NOTE: This is not completely correct as SequeLink does support SELECT FOR UPDATE statements against certain DBMSs. SequeLink does return false, as it does not support UPDATE WHERE CURRENT OF statements.
boolean supportsStoredProcedures ()	1.0	Yes	
boolean supportsSubqueriesIn Comparisons ()	1.0	Yes	
boolean supportsSubqueriesInExists ()	1.0	Yes	

DatabaseMetaData Object Methods	Version Introduced	Supported	Comments
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 (java.io.PrintWriter)	2.0 Optional	Yes	Enable Spy, which traces all of the information into the specified PrintWriter.
void setLoginTimeout (int)	2.0 Optional	Yes	

<b>Driver Object Methods</b>	<b>Version Introduced</b>	<b>Supported</b>	<b>Comments</b>
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	

<b>ParameterMetaData Object Methods</b>	<b>Version Introduced</b>	<b>Supported</b>	<b>Comments</b>
String getParameterClassName (int)	3.0	Yes	Not supported on Oracle and Legacy servers.
int getParameterCount ()	3.0	Yes	Not supported on Oracle and Legacy servers.
int getParameterMode (int)	3.0	Yes	Not supported on Oracle and Legacy servers.
int getParameterType (int)	3.0	Yes	Not supported on Oracle and Legacy servers.
String getParameterTypeName (int)	3.0	Yes	Not supported on Oracle and Legacy servers.
int getPrecision (int)	3.0	Yes	Not supported on Oracle and Legacy servers.
int getScale (int)	3.0	Yes	Not supported on Oracle and Legacy servers.
int isNullable (int)	3.0	Yes	Not supported on Oracle and Legacy servers.

ParameterMetaData Object <i>(cont.)</i> Methods	Version Introduced	Supported	Comments
boolean isSigned (int)	3.0	Yes	Not supported on Oracle and Legacy servers.

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 particular PooledConnection object can have only one Connection object open, and that is 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 someone else. This is rare, but the capability is there. The drivers do not support this "reclaiming" of connections and will throw a SQLException "Reclaim of open connection is not supported."



<b>PooledConnection Object</b> (cont.)			
Methods	Version Introduced	Supported	Comments
void removeConnectionEventListener (ConnectionEventListener)	2.0 Optional	Yes	

<b>PreparedStatement Object</b>			
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.

<b>PreparedStatement Object</b> <i>(cont.)</i>	<b>Version Introduced</b>	<b>Supported</b>	<b>Comments</b>
<b>Methods</b>			
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	
ResultSet getGeneratedKeys ()	3.0	No	Throws “unsupported method” exception.
int getMaxFieldSize ()	1.0	Yes	
int getMaxRows ()	1.0	Yes	
ResultSetMetaData getMetaData ()	2.0 Core	Yes	
boolean getMoreResults ()	1.0	Yes	
boolean getMoreResults (int)	3.0	Yes	
int getQueryTimeout ()	1.0	Yes	

<b>PreparedStatement Object</b> <i>(cont.)</i>			
<b>Methods</b>	<b>Version Introduced</b>	<b>Supported</b>	<b>Comments</b>
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, java.io.InputStream, int)	1.0	Yes	
void setBigDecimal (int, java.math.BigDecimal)	1.0	Yes	
void setBinaryStream (int, java.io.InputStream, int)	1.0	Yes	
void setBlob (int, Blob)	2.0 Core	Yes	Not supported in DB2 OS/390 V5, SQL Server 2000, Sybase, Informix, and Legacy drivers.
void setBoolean (int, boolean)	1.0	Yes	
void setByte (int, byte)	1.0	Yes	
void setBytes (int, byte [])	1.0	Yes	
void setCharacterStream (int, java.io.Reader, int)	2.0 Core	Yes	
void setClob (int, Clob)	2.0 Core	Yes	Not supported in DB2 OS/390 V5, SQL Server 2000, Sybase, Informix, and Legacy drivers.
void setCursorName (String)	1.0	No	

<b>PreparedStatement Object</b> <i>(cont.)</i>	<b>Version Introduced</b>	<b>Supported</b>	<b>Comments</b>
<b>Methods</b>			
void setDate (int, Date)	1.0	Yes	SQLServer throws "optional feature not implemented" exception.
void setDate (int, Date, java.util.Calendar)	2.0 Core	Yes	SQLServer 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 simply ignore this hint provided by the application.
void setFetchSize (int)	2.0 Core	Yes	SequeLink supports this method, but will simply 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	SQLServer 7 throws "optional feature not implemented" exception. However, there is a workaround available which maps 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	

<b>PreparedStatement Object</b> <i>(cont.)</i>			
<b>Methods</b>	<b>Version Introduced</b>	<b>Supported</b>	<b>Comments</b>
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 OS/390 and Legacy 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	SQLServer throws "optional feature not implemented" exception.
void setTime (int, Time, java.util.Calendar)	2.0 Core	Yes	SQLServer throws "optional feature not implemented" exception.
void setTimestamp (int, Timestamp)	1.0	Yes	
void setTimestamp (int, Timestamp, java.util.Calendar)	2.0 Core	Yes	

PreparedStatement Object <i>(cont.)</i>	Version Introduced	Supported	Comments
Methods			
void setUnicodeStream (int, java.io.InputStream, int)	1.0	No	Throws “unsupported method” exception. This method has been deprecated in the JDBC 3.0 specification.
void setURL (int, java.net.URL)	3.0	No	Throws “unsupported method” exception.
boolean wasNull ()	1.0	Yes	

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

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

ResultSet Object	Version Introduced	Supported	Comments
Methods			
boolean absolute (int)	2.0 Core	Yes	
void afterLast ()	2.0 Core	Yes	
void beforeFirst ()	2.0 Core	Yes	
void cancelRowUpdates ()	2.0 Core	Yes	
void clearWarnings ()	1.0	Yes	

<b>ResultSet Object</b> <i>(cont.)</i>	<b>Version Introduced</b>	<b>Supported</b>	<b>Comments</b>
<b>Methods</b>			
void close ()	1.0	Yes	
void deleteRow ()	2.0 Core	Yes	
int findColumn (String)	1.0	Yes	
boolean first ()	2.0 Core	Yes	
Array getArray (int)	2.0 Core	No	Throws "unsupported method" exception.
Array getArray (String)	2.0 Core	No	Throws "unsupported method" exception.
java.io.InputStream getAsciiStream (int)	1.0	Yes	For the implementation of getAsciiStream (), the drivers assume that the underlying data in the database only contains 7-bit ASCII characters (represented in an ASCII or EBCDIC code-page). If the assumption is true, the drivers return bytes corresponding to ASCII characters, otherwise the results are unpredictable. In the case that getAsciiStream () is called on a binary column, the driver converts the binary data to the hexadecimal representation and returns the ASCII representation. For example, the binary value 20 is represented in hexadecimal as 14 or "1" "4". In ASCII, 1 is represented by "49" and 4 is represented by "52". This implies that the ASCII stream "4952" is returned.

<b>ResultSet Object</b> <i>(cont.)</i>	<b>Version</b>		
<b>Methods</b>	<b>Introduced</b>	<b>Supported</b>	<b>Comments</b>
InputStream getAsciiStream (String)	1.0	Yes	For the implementation of getAsciiStream (), the drivers assume that the underlying data in the database only contains 7-bit ASCII characters (represented in an ASCII or EBCDIC code-page). If the assumption is true, the drivers return bytes corresponding to ASCII characters, otherwise the results are unpredictable. In the case that getAsciiStream () is called on a binary column, the driver converts the binary data to the hexadecimal representation and returns the ASCII representation. For example, the binary value 20 is represented in hexadecimal as 14 or "1" "4". In ASCII, 1 is represented by "49" and 4 is represented by "52". This implies that the ASCII stream "4952" is returned.
java.math.BigDecimal getBigDecimal (int)	2.0 Core	Yes	
java.math.BigDecimal getBigDecimal (String)	2.0 Core	Yes	
java.math.BigDecimal getBigDecimal (int, int)	1.0	Yes	
java.math.BigDecimal getBigDecimal (String, int)	1.0	Yes	
java.io.InputStream getBinaryStream (int)	1.0	Yes	



<b>ResultSet Object</b> <i>(cont.)</i>	<b>Version</b>	<b>Supported</b>	<b>Comments</b>
<b>Methods</b>	<b>Introduced</b>		
java.io.InputStream getBinaryStream (String)	1.0	Yes	
Blob getBlob (int)	2.0 Core	Yes	Not supported in DB2 OS/390 V5, SQL Server 2000, Sybase, Informix, and Legacy drivers.
Blob getBlob (String)	2.0 Core	Yes	Not supported in DB2 OS/390 V5, SQL Server 2000, Sybase, Informix, and Legacy drivers.
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	
java.io.Reader getCharacterStream (int)	2.0 Core	Yes	
java.io.Reader getCharacterStream (String)	2.0 Core	Yes	
Clob getClob (int)	2.0 Core	Yes	Not supported in DB2 OS/390 V5, SQL Server 2000, Sybase, Informix, and Legacy drivers.
Clob getClob (String)	2.0 Core	Yes	Not supported in DB2 OS/390 V5, SQL Server 2000, Sybase, Informix, and Legacy drivers.
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	

<b>ResultSet Object</b> <i>(cont.)</i>	<b>Version</b>	<b>Supported</b>	<b>Comments</b>
<b>Methods</b>	<b>Introduced</b>		
Date getDate (int, java.util.Calendar)	2.0 Core	Yes	
Date getDate (String, java.util.Calendar)	2.0 Core	Yes	
double getDouble (int)	1.0	Yes	
double getDouble (String)	1.0	Yes	
int getFetchDirection ()	2.0 Core	Yes	SequeLink supports this method, but will simply ignore this hint provided by the application.
int getFetchSize ()	2.0 Core	Yes	SequeLink supports this method, but will simply 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, java.util.Map)	2.0 Core	Yes	Map ignored.
Object getObject (String, java.util.Map)	2.0 Core	Yes	Map ignored.

<b>ResultSet Object</b> <i>(cont.)</i>	<b>Version Introduced</b>	<b>Supported</b>	<b>Comments</b>
<b>Methods</b>			
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	
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, java.util.Calendar)	2.0 Core	Yes	
Time getTime (String, java.util.Calendar)	2.0 Core	Yes	
Timestamp getTimestamp (int)	1.0	Yes	
Timestamp getTimestamp (String)	1.0	Yes	
Timestamp getTimestamp (int, java.util.Calendar)	2.0 Core	Yes	
Timestamp getTimestamp (String, java.util.Calendar)	2.0 Core	Yes	
int getType ()	2.0 Core	Yes	
java.io.InputStream getUnicodeStream (int)	1.0	Yes	Throws “unsupported method” exception.

<b>ResultSet Object</b> <i>(cont.)</i> <b>Methods</b>	<b>Version Introduced</b>	<b>Supported</b>	<b>Comments</b>
java.io.InputStream getUnicodeStream (String)	1.0	Yes	Throws “unsupported method” exception.
java.net.URL getURL (int)	3.0	No	Throws “unsupported method” exception.
java.net.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	
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.

<b>ResultSet Object</b> <i>(cont.)</i>	<b>Version</b>	<b>Supported</b>	<b>Comments</b>
<b>Methods</b>	<b>Introduced</b>		
void updateArray (String, Array)	3.0	No	Throws “unsupported method” exception.
void updateAsciiStream (int, java.io.InputStream, int)	2.0 Core	Yes	
void updateAsciiStream (String, java.io.InputStream, int)	2.0 Core	Yes	
void updateBigDecimal (int, BigDecimal)	2.0 Core	Yes	
void updateBigDecimal (String, BigDecimal)	2.0 Core	Yes	
void updateBinaryStream (int, java.io.InputStream, int)	2.0 Core	Yes	
void updateBinaryStream (String, java.io.InputStream, int)	2.0 Core	Yes	
void updateBlob (int, Blob)	3.0	No	Throws “unsupported method” exception.
void updateBlob (String, Blob)	3.0	No	Throws “unsupported method” exception.
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	

<b>ResultSet Object</b> <i>(cont.)</i>	<b>Version</b>	<b>Supported</b>	<b>Comments</b>
<b>Methods</b>	<b>Introduced</b>		
void updateCharacterStream (int, java.io.Reader, int)	2.0 Core	Yes	
void updateCharacterStream (String, java.io.Reader, int)	2.0 Core	Yes	
void updateClob (int, Clob)	3.0	No	Throws “unsupported method” exception.
void updateClob (String, Clob)	3.0	No	Throws “unsupported method” exception.
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	
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	

<b>ResultSet Object</b> <i>(cont.)</i>	<b>Version</b>	<b>Supported</b>	<b>Comments</b>
<b>Methods</b>	<b>Introduced</b>		
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	
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	

<b>ResultSetMetaData Object Methods</b>	<b>Version Introduced</b>	<b>Supported</b>	<b>Comments</b>
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	
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	



<b>ResultSetMetaData Object</b> <i>(cont.)</i>	<b>Version</b>		
<b>Methods</b>	<b>Introduced</b>	<b>Supported</b>	<b>Comments</b>
boolean isWritable (int)	1.0	Yes	

<b>RowSet Object</b>	<b>Version</b>		
<b>Methods</b>	<b>Introduced</b>	<b>Supported</b>	<b>Comments</b>
(all)	2.0 Optional	No	

<b>SavePoint Object</b>	<b>Version</b>		
<b>Methods</b>	<b>Introduced</b>	<b>Supported</b>	<b>Comments</b>
int getSavepointId ()	3.0	Yes	Savepoint not supported on Informix, ODBC Socket Server, and Legacy servers.
String getSavepointName ()	3.0	Yes	Savepoint not supported on Informix, ODBC Socket Server, and Legacy servers.

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

<b>Statement Object</b>	<b>Version</b>		
<b>Methods</b>	<b>Introduced</b>	<b>Supported</b>	<b>Comments</b>
void addBatch (String)	2.0 Core	Yes	

<b>Statement Object</b> <i>(cont.)</i> <b>Methods</b>	<b>Version Introduced</b>	<b>Supported</b>	<b>Comments</b>
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.
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	

<b>Statement Object</b> <i>(cont.)</i> <b>Methods</b>	<b>Version Introduced</b>	<b>Supported</b>	<b>Comments</b>
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	
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	

Statement Object <i>(cont.)</i>	Version	Supported	Comments
Methods	Introduced		
void setMaxRows (int)	1.0	Yes	
void setQueryTimeout (int)	1.0	Yes	Not supported on DB2 OS/390 and Legacy servers. Support for ODBC Socket Server is dependent on the ODBC driver it loads.

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

XAConnection Object	Version	Supported	Comments
Methods	Introduced		
javax.transaction.xa.XAResource getXAResource()	2.0 Optional	Yes	

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

<b>XADatasource Object</b> (cont.)			
<b>Methods</b>	<b>Version Introduced</b>	<b>Supported</b>	<b>Comments</b>
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 SequeLink 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 SequeLink JDBC driver will throw an "XA-Open failed with return code -3" exception.
void setLoginTimeout (int)	2.0 Optional	Yes	
void setLogWriter (java.io.PrintWriter)	2.0 Optional	Yes	



# D 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 JDBC DataSource object and register it to a JNDI naming service. The DataSource class is provided by your SequeLink 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”](#) 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 389](#)), 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 386](#). 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 JDBC data source and registers it to a JNDI
// naming service. This SequeLink JDBC data source uses the DataSource
// implementation provided by the SequeLink 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 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 SequeLink 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 SequeLink 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 SequeLink JDBC `DataSource` object created in section [“Creating a DataDirect SequeLink Data Source Object” on page 384](#)).

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 386](#)), 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 386](#)).

```
//*****
//
// 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 SequeLink 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 384](#) 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();  
}
```



# Index

## A

- ABS function 308
- Access Order 150, 163
- accessor support 137, 158
- ACOS function 308
- Active Sessions 154
- ADO
  - Command object 150
  - Connection object 153
  - mapping OLE DB methods 150
  - OLE DB interfaces supported 148
  - Recordset object 161
- ADO Client
  - copying data sources 112
  - creating data sources 108
  - deleting data sources 112
  - modifying data sources 111
  - renaming data sources 111
- ADO client data sources. *See* client data sources
- API functions 62
- application developers
  - ADO 131
  - JDBC 267
  - ODBC 60
- application ID
  - generating automatically 75, 169
  - specifying explicitly 74, 126, 169
  - specifying for ADO provider 169
  - specifying for JDBC driver 275
- ApplicationID ODBC connection attribute 49
- ApplicationName
  - ADO Client 126
  - Java Client 249
  - SequeLink ODBC Client 50
- arguments, null 81

- Array object 332
- ASCII
  - converting to character 305
  - function 305
- ASIN function 308
- Asynchable
  - Abort 154
  - Commit 154
- ATAN function 308
- ATAN2 function 308
- attribute=value pairs 125
- attributes
  - ADO 126
  - DistinguishedName 129
  - ODBC 49
- Autocommit Isolation Levels 154
- AutomaticApplicationID ODBC connection
  - attribute 50

## B

- BigDecimal objects 279
- blanks, generating 307
- Blob object 332
- BlockFetchForUpdate 50
- Blocking Storage Objects 150, 163
- Bookmark Information 163
- Bookmark Type 163
- bookmarks 144
- books
  - online 19
  - order form 23
  - ordering printed 22
- bound columns 88

## C

- CallableStatement object 333
- catalog functions
  - defined 80
  - effect on performance 81
- Catalog Location 154
- Catalog Term 154
- Catalog Usage 155
- CEILING function 308
- centralized odbcc.ini files 47
- Change Inserted Rows 150, 163
- changing directories for ADO client data sources 113
- CHAR function 305
- cipher suites 251
- Client
  - SequeLink ADO 101
  - SequeLink Java 227
  - SequeLink ODBC 29
- client data sources
  - configuring
    - ODBC file 34
    - ODBC User and System 31
    - on UNIX 46
  - creating
    - ADO 108
    - ODBC 30
  - deleting ADO 112
  - modifying ADO 111
- Clob object 343
- code page 33
- Column Definition 155
- Column Privileges 151, 163
- COM Object Support 155
- Command object
  - dynamic properties 150
  - methods 150
- committing data 94
- compatibility of JDBC versions 331
- compiler requirements, UNIX 61
- CONCAT function 306
- concurrency types for result sets 273
- Configuration Manager
  - menu bar 104
  - overview 102
- configuring
  - ADO client data sources 107
  - file client data sources 34
  - JDBC data sources 238
  - ODBC client data source for UNIX 46
- connecting
  - to data source, logon dialog box 117
  - using JDBCtest 179
  - using URLs (JDBC) 236
  - with a provider string 125
  - with JDBC data sources 248
- connection
  - attributes in ADO 126
  - attributes in ODBC 49
  - defining using master data source file 116
  - handles 94
  - JDBC options 227
  - managing 93
  - non-pooled 391
  - ODBC options 29
  - precedence for JDBC driver 248
  - properties for JDBC driver 248
  - testing ADO 117
  - testing ODBC on Windows 45
- Connection dialog box 117
- Connection object (ADO)
  - dynamic properties 154
  - methods supported 153
- Connection object (JDBC) 344
- connection pool manager, about 230
- connection pooling
  - connecting to a data source 389
  - Connection Pool Manager 383
  - creating a data source 386
  - creating a pooled data source object 384
  - JDBC driver 240
  - ODBC driver 94
  - terminating the pool manager 392
- Connection Status 155
- contacting Technical Support 24
- conventions used in this book 20

- converting variant types to date/time types 168
- copying, ADO client data sources 112
- COS function 308
- COT function 308
- creating
  - ADO client data sources 108
  - JDBC data sources 239
  - master data source file 116
  - ODBC Client data sources 30
  - template data source file 115
- CURDATE function 310
- Current Catalog 155
- cursors
  - forward-only result sets (JDBC) 272
  - keyset-driven scrollable (ODBC) 68
  - scroll-insensitive (Java 2 platform) 257
  - scroll-insensitive result sets (JDBC) 272
  - scroll-sensitive result sets (JDBC) 272
  - static 68
  - static scrollable (ODBC) 68
  - support for scrollable (JDBC) 273
  - support for scrollable (ODBC) 68
  - using scrollable instead of cursor library (ODBC) 92
- CURTIME function 310

## D

- data shaping 166
- data source
  - configuring
    - JDBC 238
    - on UNIX (ODBC) 46
    - User and System (ODBC) 31
    - values overridden 167
  - connecting with JDBC 248
  - creating
    - ADO client 108
    - JDBC 239
    - non-pooled 384
    - ODBC file client 34

- ODBC system 31
- ODBC user 31
  - pooled 384
- default 114
- displaying properties 105
- examples of JDBC 239
- file
  - creating a master 116
  - creating a template 115
- non-pooled 384
- system 31
- user 31
- data source information property group 136
- Data Source Name 155
- Data Source Object Threading Model 155
- Data Source property 155
- data source property group 136
- data types
  - DB2 V5 318
  - DB2 V6, V7 319
  - Informix 9 320
  - Legacy Server 328
  - mapping year format 168
  - Microsoft SQL Server 321
  - Microsoft SQL Server2000 323
  - Oracle8, Oracle9i 324, 325
  - Sybase 327
- database
  - data dictionary 84
  - meta-information, retrieving 84
  - naming using JNDI 239
- Database connection attribute 50, 126
- Database Data Dictionary
  - filters 85
  - views (DB2 for OS/390) 85
- DATABASE function 312
- DatabaseMetaData object 347
- database-specific information 317
- DataDirect Configuration Manager
  - menu bar 104
- DataDirect Connection Pool Manager 233
- DataSource object 358

- date
  - returning current 310
  - returning month 311
  - returning year 311
  - scalar functions 309
- DAYNAME function 310
- DAYOFMONTH function 310
- DAYOFWEEK function 310
- DAYOFYEAR function 310
- DB2 data types 318, 319, 321
- DB2 for OS/390
  - Database Data Dictionary views 85
  - Distributed Transaction Management support 243
  - support for scrollable cursors (JDBC) 273
  - support for scrollable cursors (ODBC) 68
- DB2 on Windows, support for scrollable cursors (ODBC) 68
- DBMS Name property 155
- DBMS Version property 155
- DBPassword 50
- DBPROP\_
  - ROWSETCONVERSIONSONCOMMAND 140
- DBPROP\_ SUBQUERIES 141
- DBPROP\_ SUPPORTEDTXNDDL 141
- DBPROP\_ SUPPORTEDTXNISOLEVELS 141
- DBPROP\_ SUPPORTEDTXNISORETAIN 141
- DBPROP\_ ABORTPRESERVE 144
- DBPROP\_ ACCESSORDER 144
- DBPROP\_ ASYNCTXNABORT 137
- DBPROP\_ ASYNCTXNCOMMIT 137
- DBPROP\_ AUTH\_ PASSWORD 142
- DBPROP\_ AUTH\_ USERID 142
- DBPROP\_ AUTH\_ PASSWORD 142
- DBPROP\_ AUTH\_ PERSIST\_SENSITIVE\_AUTHINFO 142
- DBPROP\_ AUTH\_ USERID 142
- DBPROP\_ BLOCKINGSTORAGEOBJECTS 144
- DBPROP\_ BOOKMARKINFO 144
- DBPROP\_ BOOKMARKS 144
- DBPROP\_ BOOKMARKSKIPPED 144
- DBPROP\_ BOOKMARKTYPE 144
- DBPROP\_ CANFETCHBACKWARDS 144
- DBPROP\_ CANSROLLBACKWARDS 144
- DBPROP\_ CATALOGLOCATION 137
- DBPROP\_ CATALOGTERM 137
- DBPROP\_ CATALOGUSAGE 137
- DBPROP\_ COLUMNDEFINITION 137
- DBPROP\_ COMMITPRESERVE 145
- DBPROP\_ CONCATNULLBEHAVIOR 138
- DBPROP\_ CONNECTIONSTATUS 138
- DBPROP\_ CURRENTCATALOG 136
- DBPROP\_ DATASOURCENAME 138
- DBPROP\_ DATASOURCEREADONLY 138
- DBPROP\_ DBMSNAME 138
- DBPROP\_ DBMSVER 138
- DBPROP\_ DELAYSTORAGEOBJECTS 145
- DBPROP\_ DSOTHRMODEL 138
- DBPROP\_ HETEROGENEOUSTABLES 138
- DBPROP\_ IDENTIFIERCASE 138
- DBPROP\_ IMMOBILEROWS 145
- DBPROP\_ INIT\_ DATASOURCE 142
- DBPROP\_ INIT\_ HWND 142
- DBPROP\_ INIT\_ MODE 142
- DBPROP\_ INIT\_ PROMPT 142
- DBPROP\_ INIT\_ PROVIDERSTRING 143
- DBPROP\_ INIT\_ CATALOG 142
- DBPROP\_ INIT\_ DATASOURCE 142
- DBPROP\_ INIT\_ HWND 142
- DBPROP\_ INIT\_ LCID 142
- DBPROP\_ INIT\_ MODE 142
- DBPROP\_ INIT\_ OLEDBSERVICES 142
- DBPROP\_ INIT\_ PROMPT 142
- DBPROP\_ INIT\_ PROVIDERSTRING 142
- DBPROP\_ LITERALBOOKMARKS 145
- DBPROP\_ LITERALIDENTITY 145
- DBPROP\_ LOCKMODE 146
- DBPROP\_ MAXINDEXSIZE 138
- DBPROP\_ MAXOPENROWS 146
- DBPROP\_ MAXPENDINGROWS 146
- DBPROP\_ MAXROWS 146
- DBPROP\_ MAXROWSIZE 138
- DBPROP\_ MAXROWSIZEINCLUDESBLOB 139
- DBPROP\_ MAXTABLEINSELECT 139
- DBPROP\_ MEMORYUSAGE 146
- DBPROP\_ MULTIPLEPARAMSETS 139
- DBPROP\_ MULTIPLERESULTS 139
- DBPROP\_ MULTIPLESTORAGEOBJECTS 139

DBPROP\_MULTITABLEUPDATE 139  
 DBPROP\_NULLCOLLATION 139  
 DBPROP\_OLEOBJECTS 139  
 DBPROP\_OPENROWSETSUPPORT 139  
 DBPROP\_ORDERBYCOLUMNSINSELECT 139  
 DBPROP\_OTHERINSERT 146  
 DBPROP\_OTHERUPDELETEDELETE 146  
 DBPROP\_OUTPUTPARAMETERAVAILABILITY 139  
 DBPROP\_OWNINGINSERT 146  
 DBPROP\_OWNINGUPDELETEDELETE 146  
 DBPROP\_PERSISTENTIDTYPE 139  
 DBPROP\_PREPAREABORTBEHAVIOR 140  
 DBPROP\_PREPARECOMMITBEHAVIOR 140  
 DBPROP\_PROCEDURETERM 140  
 DBPROP\_PROVIDERFRIENDLYNAME 140  
 DBPROP\_PROVIDERNAME 140  
 DBPROP\_PROVIDEROLEDBVER 140  
 DBPROP\_PROVIDERVER 140  
 DBPROP\_QUOTEDIDENTIFIERCASE 140  
 DBPROP\_REENTRANTEVENTS 146  
 DBPROP\_REMOVEDELETED 146  
 DBPROP\_REPORTMULTIPLECHANGES 146  
 DBPROP\_RETURNPENDINGINSERTS 146  
 DBPROP\_ROWRESTRICT 147  
 DBPROP\_ROWTHREADMODEL 147  
 DBPROP\_SCHEMATERM 140  
 DBPROP\_SCHEMAUSAGE 140  
 DBPROP\_SERVERCURSOR 147  
 DBPROP\_SERVERNAME 140  
 DBPROP\_SESS\_ AUTOCOMMITISOLEVELS 148  
 DBPROP\_SQLSUPPORT 140  
 DBPROP\_STRONGIDENTITY 147  
 DBPROP\_STRUCTUREDSTORAGE 140  
 DBPROP\_TABLETERM 141  
 DBPROP\_TRANSACTEDOBJECT 147  
 DBPROP\_UNIQUEROWS 147  
 DBPROP\_UPDATABILITY 147  
 DBPROP\_USERNAME 141  
 default data source files  
     master 114  
     template 114  
 Default Length for Long Data 127

defining  
     data source using the ODBC Administrator 31  
     default setup options 114  
 DEGREES function 308  
 Delay Storage Object Updates 163  
 deletes, positional 95  
 deleting ADO client data sources 112  
 developing ODBC applications 60  
 DIFFERENCE function 306  
 directory structure for SequeLink Java Client 232  
 Distinguished Name identifier 127  
 DistinguishedName attribute  
     ADO 129  
     ODBC 51  
 Distributed Transaction Management 243  
 documentation  
     order form 23  
     ordering printed books 22  
 downloading applets, tips 278  
 DPROP\_GROUPBY 138  
 Driver object 359  
 DSN connection attribute 50  
 dynamic properties  
     Command object 150  
     Connection object 154  
     Recordset object 163

## E

EnableDescribeParam 51  
 environment variables (UNIX) 47  
 error handling  
     ADO provider 170  
     JDBC driver 276  
     ODBC driver 77  
 examples  
     connection pooling 240  
     creating and using JDBC data sources 239  
     odbc.ini file configured for Solaris 46

- Spy log 264
- URL (Spy) 263
- EXP function 308
- EXTRACT function 310

## F

- fetching
  - backward 144, 151, 163
  - BigDecimal objects 279
- FetchNextOnly 51
- file
  - creating a master data source 116
  - creating a template data source 115
- file client data sources, configuring 34
- File System JNDI Provider 235
- files in SequeLink Java Client directory 232
- filters, Database Data Dictionary 85
- FixCharTrim 52
- FLOOR function 308
- forward only cursors 89
- forward-only result sets (JDBC) 272
- functions
  - cancelling in multithreaded applications 67, 270
  - date and time 309
  - numeric 307
  - ODBC API 62
  - scalar 298
  - string 305

## G

- generating ODBC application IDs
  - automatically 75
- GetOutputParams 52
- GROUP BY Support 156

## H

- header files and libraries, platform-specific 60
- help for SequeLink Java Client 19
- Heterogeneous Table Support 156
- HLogonID 52
- Hold Rows 151, 163
- Host attribute 53, 127
- host password 52
- Host Password attribute 127
- HOURL function 310
- HPassword 52

## I

- Identifier Case Sensitivity 156
- IFNULL function 312
- Immobile Rows 164
- Informix 9
  - persisting result set as XML 76
  - support for scrollable cursors (ODBC) 68
- Informix 9 data types 320
- Initial Catalog 156
- initialization properties 141
- INSERT function 306
- interfaces supported by ADO provider 132
- IOpenRowset 167
- IPersistFile 167
- IRowset interface 132
- IRowsetIdentity 143
- isolation levels 157, 328
- Isolation Retention 157

## J

- JAR files 278
- Java 2 Platform 256
- Java Client
  - configuring JDBC data sources 239
  - directory structure 232
  - overview 227
  - using connection pooling 240
  - using on a Java 2 Platform 256
  - using the JTA 243
- Java Transaction API 243
- Java Virtual Machine versions 331
- JCA
  - resource adapter class 269
  - resource adapters
    - overview 244
    - using from an application 245
    - using with application server 245
- JDBC
  - compatibility 331
  - connection options 227
  - data types supported by driver
    - DB2 V5 on OS/390 318
    - DB2 V6, V7 319
    - Informix 320
    - Legacy Server 328
    - Microsoft SQL Server 7 321
    - Microsoft SQL Server2000 323
    - Oracle8 324
    - Oracle9i 325
    - Sybase 327
  - forward-only result sets 272
  - functionality supported 332
  - scroll-insensitive result sets 272
  - scroll-sensitive result sets 272
  - support 331
  - support for 2.0 functionality 268
- JDBC 2.0 Optional Package 232, 234
- JDBC data sources 232
- JDBC driver
  - connection properties 249
  - error handling 276
  - functionality for JDBC objects 332
  - registering with JDBC driver Manager 235
  - specifying application IDs 275
  - specifying connection URLs 236
  - threading for driver 270
  - tracking calls 229
  - using data sources 248
  - using Distributed Transaction Management 243
  - using Spy with data sources 262
- JDBC Spy
  - attributes 261
  - log example 264
  - overview 229
  - registering the JDBC driver 260
  - URL syntax 261
  - using with JDBC data sources 262
- JDBCTest 233
  - batch execution on a prepared statement 196
  - configuring 176
  - connecting 179
  - establishing savepoints 201
  - executing a prepared statement 186
  - executing a simple select statement 184
  - files 233
  - LOB support 220
  - retrieving database metadata 190
  - returning ParameterMetaData 200
  - scrolling through a result set 192
  - starting 177
  - tutorial 175
  - updateable result sets 208
  - using 175
- JNDI
  - registering a SequeLink JDBC DataSource object 384
  - using for naming databases 239
- JNDI 1.2 234
- JTA 1.0.1 234
- jXTransformer
  - about 230
  - installed files 233

## K

keyset-driven scrollable cursors (ODBC) 68, 70

## L

LCASE function 306

LDAP

- configuring the ADO Client 110
- configuring the ODBC Client 33
- creating a JDBC data source 239
- property in ADO 129
- setting parameters in JDBCTest 176
- specifying port for the listener 54, 110, 128
- TCP/IP address of the LDAP server 53
- UseLDAP attribute (ADO) 129
- UseLDAP attribute (ODBC) 55

LDAP JNDI Provider 235

LEFT function 306

Legacy Server

- data types 328
- support for scrollable cursors (ODBC) 68

LENGTH function 306

libraries and header files, platform-specific 60

Linux. *See* UNIX

Literal Bookmarks 164

Literal Row Identity 151, 164

loading the SequeLink JDBC driver 235

Locale Identifier 157

LOCATE function 306

Lock Mode 151, 164

log example, Spy 264

LOG function 308

LOG10 function 308

logging JDBC calls 259

LogonID 53

long data, performance issues with  
retrieving 86

LTRIM function 306

## M

ManagedConnectionFactory 269

managing

- connections to improve driver  
performance 93
- JDBC data sources 239
- retrieval of database meta-information 84

mapping data types

- ADO provider 168
- supported by DB2 V5 318
- supported by DB2 V6, V7 319
- supported by Informix 9 320
- supported by Microsoft SQL Server 7 321
- supported by Microsoft SQL Server2000 323
- supported by Oracle8 324
- supported by Oracle9i 325
- supported by Sybase 327

master data source file

- creating a 116
- overview 114

Maximum Index Size 157

Maximum Open Chapters 157

Maximum Open Rows 151, 164

Maximum Pending Rows 151, 164

Maximum Row Size 157

Maximum Row Size Includes BLOB 157

Maximum Rows 151, 164

Maximum Tables in SELECT 157

Memory Usage 151, 164

meta-information

- limiting amount to be retrieved 85
- managing retrieval of 84

methods

- ADO Command object 150
- ADO Recordset object 161
- supported by ADO Connection object 153

Microsoft SQL Server, support for scrollable  
cursors (ODBC) 68

Microsoft SQL Server2000 data types 323

MINUTE function 311

MOD function 308



- Mode property 157
- modifying, ADO client data sources 111
- MONTH function 311
- MONTHNAME function 311
- Multiple Connections property 157
- Multi-Table Update 157
- multithreaded applications
  - cancelling functions 67, 270
  - functionality of the SequeLink ODBC driver 66

## N

- naming databases with JNDI 239
- NewPassword 53
- non-pooled data source 384
- NOW function 311
- null arguments, impact on performance 81
- NULL Collation Order 158
- NULL Concatenation Behavior 158
- Number
  - raising to power 308
  - rounding 308
- numeric functions 307

## O

- objects supported
  - by the SequeLink ADO provider 132
  - by the SequeLink JDBC driver 332
- Objects Transacted 152, 164
- ODBC
  - functions that improve driver performance 90
  - keyset-driven scrollable cursors 70
  - Level 2.x API functions 62
  - Level 3.x API functions 64
  - optimizing performance 80
  - static scrollable cursors 68, 69
- ODBC Administrator
  - configuring file client data sources 34
  - configuring User and System client data sources 31
  - starting 30
- ODBC Client
  - connecting to a data source
    - using a connection string 48
  - connection string 48
- ODBC driver
  - application IDs 74
  - cancelling functions in multithreaded applications 67, 270
  - configuring file client data sources 34
  - connecting to a data source
    - using a logon dialog box 118
  - connecting to a data source using a logon dialog box 39
  - connection attributes 49
  - connection options 29
  - error handling 77
  - ODBC API functions supported 62
  - overview 29
  - required libraries and header files 60
  - specification supported 29
  - threading 65
- ODBC Socket, support for scrollable cursors (ODBC) 68
- ODBC translators 33
- odbc.ini (UNIX)
  - sample odbc.ini file 46
  - using a centralized odbc.ini file 47
- ODBCINI environment variable (UNIX) 47
- ODBCTest 73
- OLE DB
  - mapping methods to ADO 150
  - objects supported by the SequeLink ADO provider 132
  - property groups supported 135
  - schema rowsets supported 134
- OLE DB Services 158
- OLE DB Version 158

- online books
  - order form 23
  - ordering printed books 22
- Open Rowset Support 158
- Oracle8
  - data types 324
  - support for scrollable cursors (JDBC) 273
  - support for scrollable cursors (ODBC) 69
  - using stored procedures with 70
- Oracle9i
  - data types 325
  - support for scrollable cursors (JDBC) 273
  - support for scrollable cursors (ODBC) 69
  - using stored procedures with 70
- Order 23
- ORDER BY Columns In Select List 158
- ordering printed books 22
- Others' Changes Visible 152, 164
- Others' Inserts Visible 152
- Others' Inserts Visible 164
- Output Parameter Availability 158
- Own Changes Visible 152, 164
- Own Inserts Visible 152, 164

## P

- ParameterMetaData object 359
- Pass By Ref Accessors 158
- Password
  - ODBC connection attribute 53
  - property for Connection object 158
- performance hints
  - avoiding the cursor library 92
  - choosing the data type 317
  - committing data 94
  - JDBC
    - fetching BigDecimal objects 279
    - reducing download time 278
  - managing ODBC driver connections 93

- ODBC driver
  - catalog functions 80
  - locking a row when isolation level is
    - Read committed 50
  - managing retrieval of database
    - meta-information 84
  - null arguments 81
  - updating data in databases 95
  - using a dummy query 83
  - reducing the size of retrieved data 86
  - retrieving data with SQLExtendedFetch 89
  - using bound columns 88
- performance optimization
  - designing JDBC applications 288
  - retrieving data 283
  - retrieving long data 283
  - selecting JDBC objects and methods 285
  - updating data 290
- persisting information 167
- persisting result set as XML data file 76
- PI function 308
- platform-specific header files and libraries 60
- pool manager. *See* connection pooling
- pooled data source 384
- PooledConnection object 360
- Port (ODBC connection attribute) 54
- Port attribute 128
- positional updates and deletes 95
- POWER function 308
- precedence of JDBC connection properties 248
- Prepare Abort Behavior 158
- Prepare Commit Behavior 158
- PreparedStatement object 361
- Preserve on Abort 152, 165
- Preserve on Commit 152, 165
- printed books
  - order form 23
  - ordering 24
- Procedure Term 158
- Prompt property for Connection object 158

- properties
  - ADO Command object 150
  - ADO Connection object 154
  - ADO Recordset object 163
  - Data Source Information property group 136
  - Data Source property group 136
  - displaying ADO provider data source 105
  - Initialization property group 141
  - JDBC connection 248, 249
  - Rowset property group 143
  - Session property group 148
- property groups supported 135
- Provider Friendly Name 158
- Provider Name 159
- provider string 125
- Provider Version 159
- pseudo-columns 96

## Q

- QUARTER function 311
- Quick Restart 152, 165
- Quoted Identifier Sensitivity 159

## R

- RADIANS function 308
- RAND function 308
- Read Only Data Source 159
- read-only result set 273
- Recordset object
  - dynamic properties 163
  - methods 161
  - overview 161
- Ref object 366
- Referenceable object 366
- registering the JDBC driver 235
- Remove Deleted Rows 152, 165
- renaming an ADO Client data source 111

- REPEAT function 306
- REPLACE function 306
- Report Multiple Changes 152, 165
- resource adapters, JCA 231
  - overview 244
  - resource adapter class 269
- Resource Archive 233
- result sets (JDBC)
  - concurrency types 273
  - types 272
  - updatable supported 274
- ResultSet object 366
- ResultSetMetaData object 376
- retrieving data
  - avoiding long data 86
  - reducing the size 86
  - using bound columns 88
  - using SQLExtendedFetch 89
- retrieving database meta-information 84
- Return Pending Inserts 152, 165
- reusing connections with connection pooling 240
- RIGHT function 306
- ROUND function 308
- Row Privileges 152, 165
- Row Threading Model 152, 165
- rowset
  - characteristics 143
  - properties 143
  - using 167
- Rowset Conversions on Command 159
- RowSet object 377
- RTRIM function 307

## S

- SavePoint object 377
- saving current connection information 167
- scalar functions 298
- schema rowsets supported 134
- Schema Term 159
- Schema Usage 159

- Scroll Backward 152, 165
- scrollable cursors
  - concurrency types (JDBC) 273
  - insensitive (JDBC) 257
  - result set types (JDBC) 272
  - using keyset-driven (ODBC) 70
  - using static (ODBC) 69
  - using with ODBC driver 68
  - using with SequeLink JDBC driver 272
- scrolling backward 144
- scroll-insensitive result sets (JDBC) 272
- scroll-sensitive result sets (JDBC) 272
- SECOND function 311
- SequeLink ADO provider
  - ADO Command object 150
  - connection attributes 126
  - data shaping 166
  - data source information properties 136
  - data source property group 136
  - data types
    - DB2 V5 318
    - DB2 V6, V7 319
    - Informix 9 320
    - Legacy Server 328
    - Microsoft SQL Server 2000 323
    - Microsoft SQL Server 7 321
    - Oracle8 324
    - Oracle9i 325
    - Sybase 327
  - error handling 170
  - initialization properties 141
  - objects and interfaces 132
  - overview 101
  - rowset properties 143
  - rowset property group 143
  - session property group 148
  - SQL grammar supported 160
- SequeLink Java Client
  - directory structure 232
  - overview 227
  - using on a Java 2 Platform 256
- SequeLink JDBC driver
  - configuring data sources 238
  - connection properties 248
  - creating data sources 239
  - JDBC 3.0 functionality 268
  - loading 235
  - on Java 2 platform 256
  - overview 228
  - registering with JDBC driver Manager 235
  - specification supported 228
  - specifying application IDs 275
  - specifying connection URLs 236
  - supported JDBC connection properties 249
  - using connection pooling 240
  - using Distributed Transaction Management 243
- SequeLink ODBC driver
  - configuring file client data sources 34
  - connection dialogs 39
  - data types
    - DB2 V5 318
    - DB2 V6, V7 319
    - Informix 9 320
    - Legacy Server 328
    - Microsoft SQL Server 7 321
    - Microsoft SQL Server2000 323
    - Oracle8 324
    - Oracle9i 325
    - Sybase 327
  - functions that improve performance 90
  - isolation levels 328
  - overview 29
  - persisting result set as XML data file 76
  - required libraries and header files 60
  - threading 65
- SequeLink Proxy Server
  - files 233, 234
  - overview 228
  - using SSL encryption 228
- Serializable object 377
- Server Cursor 152, 165
- Server Name 160
- session property group 148
- sessions
  - maximum number supported 137
  - setting maximum number supported 154

- setting
    - maximum number of sessions 154
    - ODBCINI environment variable (UNIX) 47
  - shell script, using to set environmental variables 47
  - SIGN function 308
  - SIN function 309
  - Skip Deleted Bookmarks 165
  - SLKStaticCursorLongColBuffLen 54
  - SOUNDEX function 307
  - SPACE function 307
  - specification supported
    - JDBC 228, 268
    - ODBC driver 29
    - OLE DB 140
  - specifying provider-specific logon information 156
  - Spy
    - about 259
    - attributes 261
    - jar file 234
    - log example 264
    - options 261
    - registering the JDBC driver 260
    - URL examples 263
    - URL syntax 261
    - using with JDBC data sources 262
  - SQL Server, support for scrollable cursors (ODBC) 68
  - SQL Support 160
  - SQLCancel, effect of threading 67, 270
  - SQLColumns, performance implications 83
  - SQLExecDirect 90
  - SQLExtendedFetch 89
  - SQLParamOptions 64, 91
  - SQLPrepare 90
  - SQLSetConnectAttr 75
  - SQLSpecialColumns 96
  - SQRT function 309
  - SSL encryption
    - cancel functionality 271
    - connection URL format 236
    - JAR files 232, 233
    - permissions required 257
    - using with the proxy server 228
  - starting
    - Configuration Manager 105
    - ODBC Administrator 30
  - statement handles 94
  - Statement object 377
  - static scrollable cursors (ODBC) 68, 69
  - Status bar (Configuration Manager) 105
  - stored procedure
    - using with ODBC driver 90
    - using with Oracle 70
  - string
    - changing case of 306
    - functions 305
    - length of 306
    - removing blanks from 306, 307
    - returning substring of 307
  - Strong Row Identity 152, 165
  - Struct object 380
  - Structured Storage 160
  - Subquery Support 160
  - SUBSTRING function 307
  - SupportLink 24
  - Sybase
    - data types 327
    - support for scrollable cursors (ODBC) 69
  - syntax for URLs for Spy 261
- ## T
- table characteristics, determining 83
  - Table Term 160
  - TAN function 309
  - TCP port 237
  - TCP/IP port for SequeLink listener 54
  - Technical Support, contacting 24
  - template data source file
    - creating a 115
    - overview 114
  - terminating the pool manager 392

- testing
  - ADO connections 117
  - ODBC connections on Windows 45
- threading
  - ADO provider 138, 155
  - JDBC driver 270
  - ODBC driver use 65
- time functions 309
- TIMESTAMPADD function 311
- TIMESTAMPDIFF function 311
- tracking JDBC calls 229, 262
- Transaction DDL 161
- transaction isolation levels 157
- Translate button 33
- transliteration 33
- TRUNCATE function 309

## U

- UCASE function 307
- Unicode, support for 168
- Unique Rows 153, 165
- UNIX
  - compiler requirements 61
  - configuring an ODBC data source 46
  - multithreading functionality of the
    - SequeLink ODBC driver 66
  - setting environmental variables 47
  - using a centralized odbc.ini file 47
- Updatability 153, 165
- updatable result sets 273
- updates, positional 95
- updating opened rowset 143
- URL
  - examples (Spy) 263
  - format for connections (JDBC) 236
- Use Bookmarks 153, 165
- Use LDAP attribute (ADO) 129
- UseLDAP attribute (ODBC) 55
- USER function 312
- User Name property of Connection object 161

- using
  - centralized odbc.ini files (UNIX) 47
  - keyset-driven cursors 68
  - scrollable cursors
    - with the JDBC driver 274
    - with the ODBC driver 68
  - SequeLink Java Client on Java 2 Platform 256
  - static cursors 68
  - stored procedures with Oracle 70

## W

- WEEK function 311
- window handle 142
- Window Handle (ADO Connection object property) 161
- Windows
  - configuring the SequeLink ADO Client 101
  - multithreading functional of the
    - SequeLink ODBC driver 66
  - required ODBC libraries and header files 60
  - testing ODBC connections 45
- WorkArounds attribute 55

## X

- XAConnection object 380
- XADatasource object 380
- XML
  - jXTransformer
    - about 230
    - persistence, implementing 76
    - persisting result set as (Informix driver) 76
- XPath 230

## Y

year format for conversions 168  
YEAR function 311

## Z

z/OS. *See* OS/390