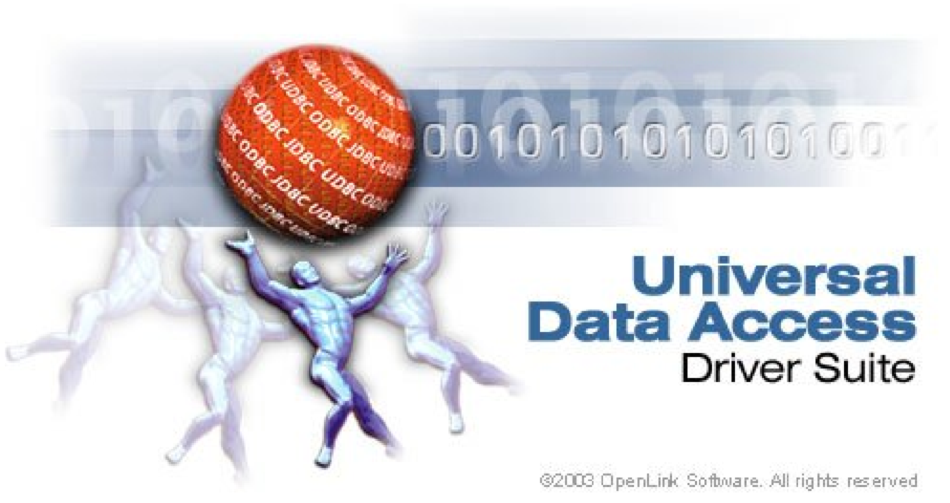


OpenLink ODBC Driver (Multi-Tier Edition) Documentation

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

1 OpenLink ODBC Driver (Multi-Tier Edition) Documentation

1.1 OpenLink Software Documentation Team `<docs@openlinksw.com>`

`<docs@openlinksw.com>`

Table of Contents

List of Figures

List of Tables

2 Preface

Table of Contents

2.1 Conventions

A note on the typographical conventions that have been used in this text:

This is the normal font type used for ordinary text.

```
This monospaced font is used to describing program code.
```

```
This monospaced font is used to describe screen output, differentiated from code if required.
```

2.1.1 This See Also:

is a tip used for cross-referencing meterial either within the documentation, or externally.

2.1.2 This Note:

is a note used for revealing a point of interest or special feature

Special formatting is used to highlight `functions()` and their *`parameters`* above the rest of the text.

Important keywords are *emphasised* using bolding (or strong character typing) and text that is

2.1.3 Important

very important and must be noticed

is displayed in a traditional red warning color

2.2 Copyright

3 Chapter 1. Overview

Abstract
ODBC, and further developments such as JDBC, OLEDB, define a low-level set of calls which allow client applications and servers applications to exchange instructions and share data without needing to know anything about each other. It applies to any Client-Server operation, whether or not the client and server applications are resident on the same machine, or on different PC's, or even if the server is on a remote machine running a different operating system.

This chapter introduces OpenLink's offerings to provide these technologies to you.

OpenLink Universal Data Access (UDA) Multi Tier consists of three main components, of varying types:

- Generic Client. One client driver connecting to a variety of data sources.
    - ODBC
    - UDBC
    - JDBC
    - OLE-DB
- Request Broker. Responsible for managing connections between Generic Clients and Server Agents. The Request Broker is completely configurable and its responses to incoming requests from Clients controlled by the *Sessions Rules Book* (oplrqb.ini).
- Server Agents. Task specific applications exclusively launched by the Request Broker for Generic Clients to communicate with, and facilitate their needs.
    - Database Agents. one specific for each database and major version as required: Oracle, Progress, Ingres, Postgres, SQL Server, etc...
    - Proxy Agent. special agent for forwarding requests to another Request Broker, particularly useful on firewalls.
    - ODBC Agent. Special case of Database Agent where another ODBC datasource is the database.

The Request Broker and Server Agents are typical addressed as the server components. This is a logical distinction because the client and server can be the same machine. In testing environments this is often the case.

The Request Broker is responsible for brokering the services of OpenLink Data Access and Service Providing Agents. It is also the component responsible for coordinating and controlling your entire OpenLink Data Access session irrespective of Data Access mechanism being used. This is the heart and soul of the OpenLink Database Independent Communications Layer. It is the technology that enables the OpenLink Data Access drivers to communicate with your backend database engines alleviating the need to acquire additional database specific networking software from your backend database vendor(s). More than one Broker may exist on a system if each instance is defined separately.

One or more Database Agents (one for each supported database engine) are database specific components that provide the actual backend database connectivity and data access services to your OpenLink Data Access Clients (ODBC, JDBC, UDBC, OLE-DB etc.). These components are actually clients from the perspective of each supported backend database, this is because they are built using the Call Level Interfaces (CLIs) or Embedded SQL interfaces of these backend databases.

Service Provider Agents (ODBC, JDBC, PROXY) are generic agents that provide Distributed Data Access Protocol handling services to OpenLink Data Access Drivers. The data access protocols supported are as follows:

- *ODBC Agent* - enabling OpenLink Multi-Tier ODBC Drivers to connect to local or remote non OpenLink ODBC Drivers
- *JDBC Agent* - enabling OpenLink JDBC Drivers to connect local or remote OpenLink or non OpenLink ODBC or UDBC Drivers
- *PROXY Agent* - enabling all OpenLink Data Access Clients to connect to OpenLink Database Agents that do not reside on the same server machine as the prime OpenLink Request Broker. The prime Broker being the Request Broker that all your OpenLink Clients are configured to request data access services from. This enables you to configure your OpenLink Database Agents for use in N-Tier distributed computing environments.

An architectural overview of the OpenLink Multi-tier Data Access infrastructure is available at: www.openlinksw.com/info/mtproduct.htm. Note that the OpenLink Request Broker is the core component that makes up the OpenLink Database Independent Communications Layer in this illustration.

The components listed above are presented to you for download at the end of your interaction with the OpenLink Software Download Wizard.

4 Chapter 2. New Features in OpenLink UDA

Table of Contents

4.1 New Features in OpenLink UDA v6.0

The following are new features in the Universal Data Access Suite version 6.0:

1. Oracle:

   Real Application Cluster (RAC) Support

   Connection-Pooling

   Direct Connectivity to Oracle without Net8|9|10 or SQL*Net (Single Tier Drivers)
2. Ingres / OpenIngres:

   Ingres Native ADO.Net Provider
3. Generic, cross-database features:

   Distributed Transaction Support

   PAM, LDAP & Active Directory support

   Support for XML Types

4.2 New Features in OpenLink UDA v5.2

The following are new features of the Universal Data Access Suite version 5.2:

64-bit ODBC Drivers for Windows, Linux, FreeBSD, Solaris, AIX, HP-UX, Tru64 Linux, IRIX across Itanium, Opteron, PA-RISC, IBM Risc, and PowerPC processors

Addition of SSL support to the existing data encryption functionality in our Multi-Tier Drivers

New ADO.NET managed providers for all supported databases

New ODBC.NET provider that facilitates compatability between ADO.NET clients and third-party ODBC Drivers

Enhanced ODBC-JDBC Bridge Driver that provides JDBC connectivity to any ODBC or ADO.NET client application

JDBC-to-ODBC Bridge Driver that provides ODBC connectivity to Java Application

A compatibility layer for Microsoft InfoPath that provides connectivity to non SQL Server and ACCESS databases via ODBC

JDBC 3.0-compliant Drivers for version 1.5 of the Java Virtual Machine and SDK

5 Chapter 3. OpenLink ODBC Driver (Multi-Tier Edition) Client Component

Abstract
This chapter describes the ODBC client component of OpenLink Universal Data Access (UDA).

ODBC stands for Open Database Connectivity, a data-access API implementation from Microsoft that conforms to the X/Open SQL CLI specification. ODBC links ODBC compliant applications to ODBC Drivers via the ODBC Driver Manager.

ODBC Driver Managers exist on other platforms such as Mac, Unix and Linux. These platforms usually make use of the "iODBC Driver Manager". Additional information regarding iODBC can be obtained from: www.iodbc.org.

Table of Contents

5.1 Generic Client for Windows

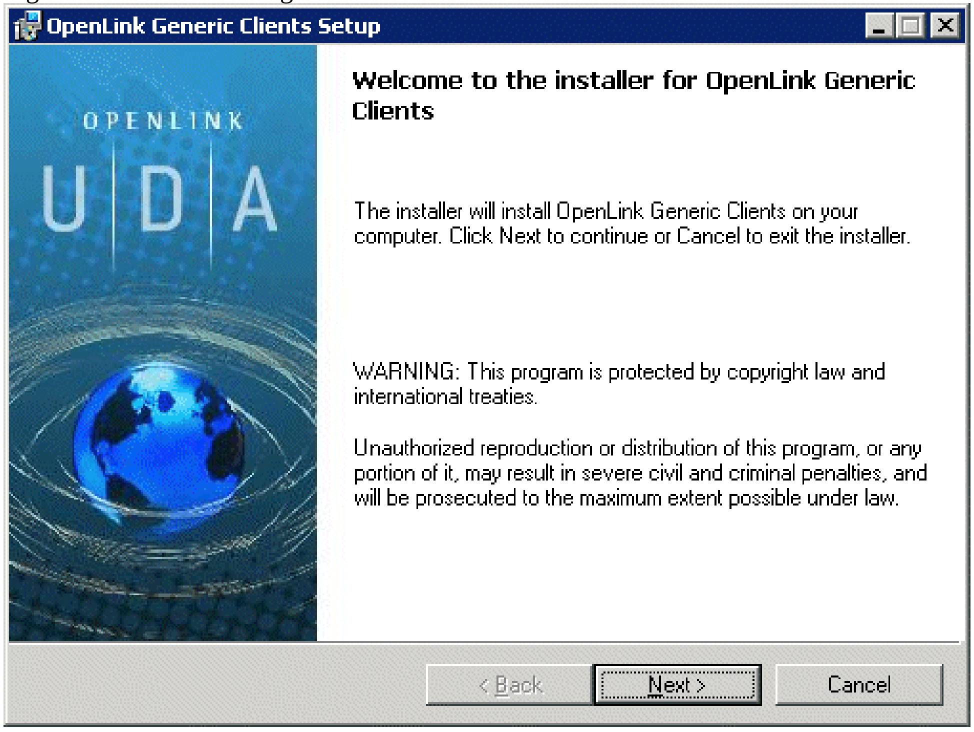
5.1.1 Installation (32-bit)

1. After downloading the OpenLink Data Access Driver Suite client components from our Website, Unzip the contents of the ZIP format archive into a temporary installation directory on your client machine.
2. Run the setup.exe installer program from the temporary installation directory and then follow the on screen prompts.
3. After installation you will be able to safely remove the files in the temporary directory.

5.1.2 Installation (64-bit)

The OpenLink Multi-Tier Generic ODBC client is distributed in a single windows msi file. Simply run the required installer wao3zzzz.msi or wio3zzzz.msi for Opteron (AMD64) and Itanium (IA64) respectively, which will display the following Welcome message:

Figure 3.1. win64instmt1.gif



Welcome to the installer for OpenLink Generic Clients

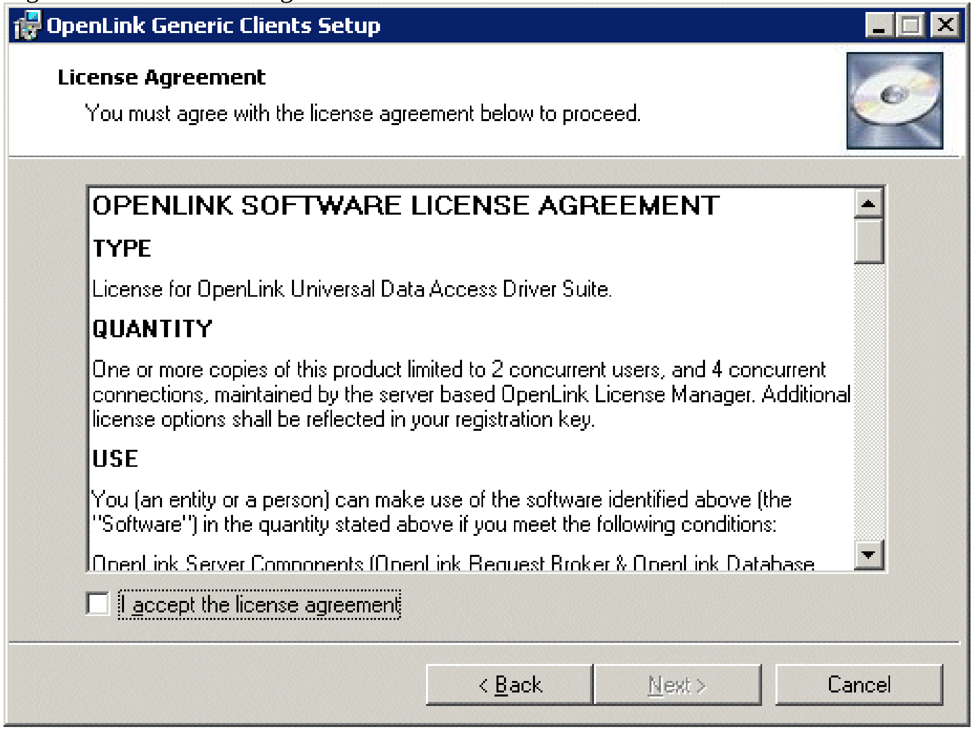
The installer will install OpenLink Generic Clients on your computer. Click Next to continue or Cancel to exit the installer.

WARNING: This program is protected by copyright law and international treaties.

Unauthorized reproduction or distribution of this program, or any portion of it, may result in severe civil and criminal penalties, and will be prosecuted to the maximum extent possible under law.

The next screen will display the License Agreement for use with the OpenLink Multi-Tier Driver product. Please read this. If you agree, then select the accept checkbox and continue:

Figure 3.2. win64instmt2.gif



License Agreement

You must agree with the license agreement below to proceed.

OPENLINK SOFTWARE LICENSE AGREEMENT

TYPE

License for OpenLink Universal Data Access Driver Suite.

QUANTITY

One or more copies of this product limited to 2 concurrent users, and 4 concurrent connections, maintained by the server based OpenLink License Manager. Additional license options shall be reflected in your registration key.

USE

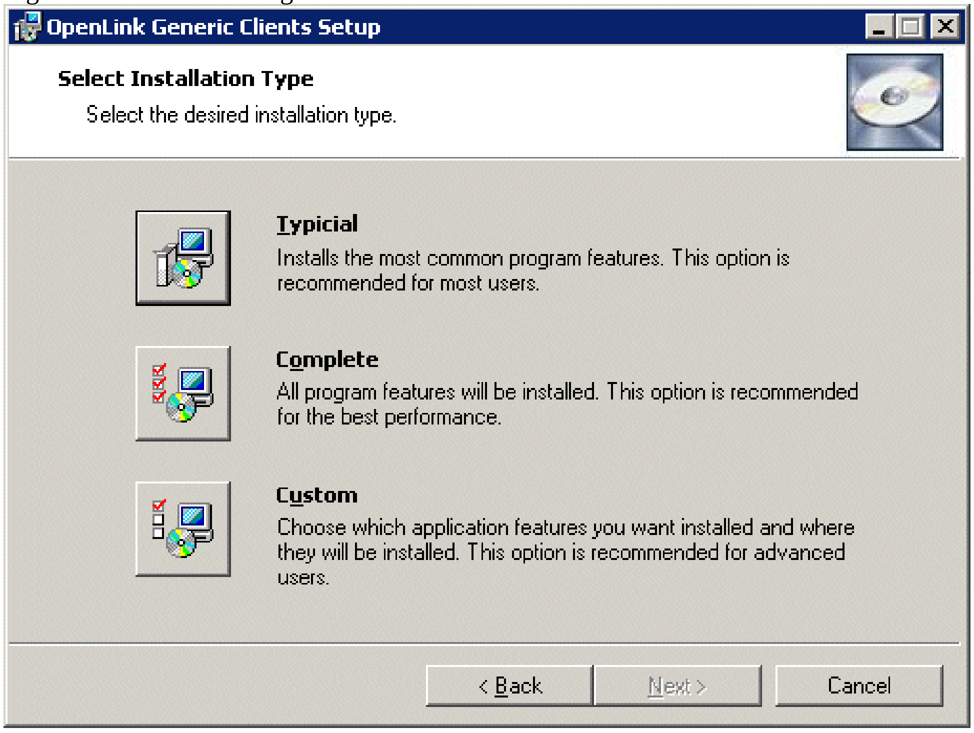
You (an entity or a person) can make use of the software identified above (the "Software") in the quantity stated above if you meet the following conditions:

OpenLink Server Components (OpenLink Request Broker & OpenLink Database

☐ I accept the license agreement

Chooses the type of installation to perform, noting that typical and complete will immediately install the default or complete components respectively, otherwise select the custom option which also the compenents to be installed to be chosen:

Figure 3.3. win64instmt3.gif



OpenLink Generic Clients Setup

Select Installation Type

Select the desired installation type.

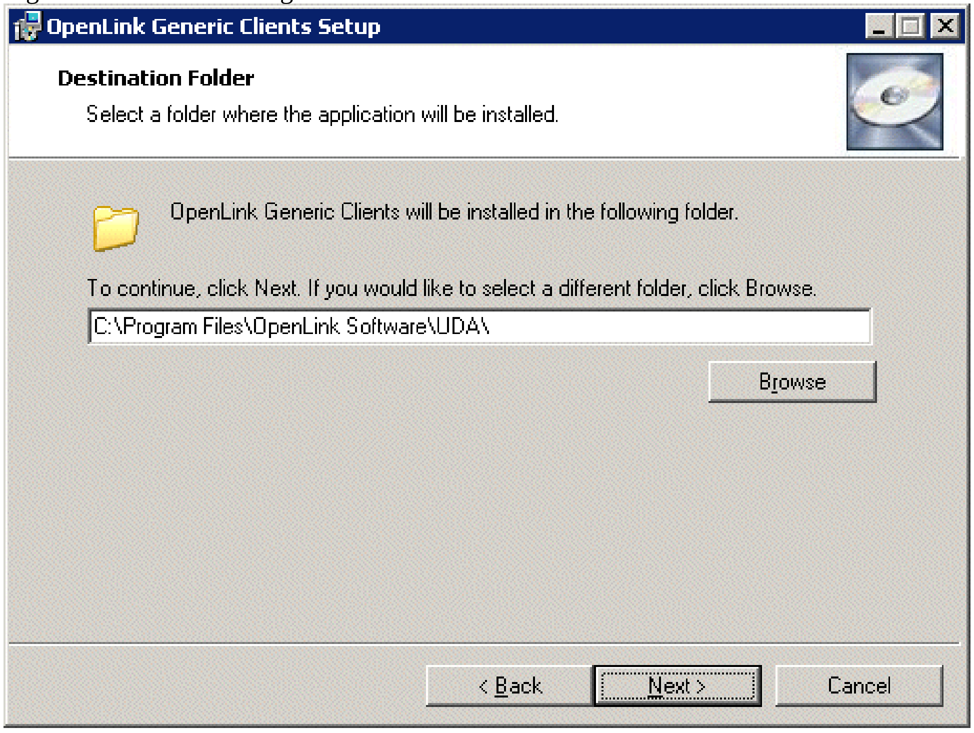
Typicial
Installs the most common program features. This option is recommended for most users.

Complete
All program features will be installed. This option is recommended for the best performance.

Custom
Choose which application features you want installed and where they will be installed. This option is recommended for advanced users.

Assuming you chosen the custom installation option you will then be promptied with the option to change the installation directory if required:

Figure 3.4. win64instmt4.gif



OpenLink Generic Clients Setup

Destination Folder

Select a folder where the application will be installed.

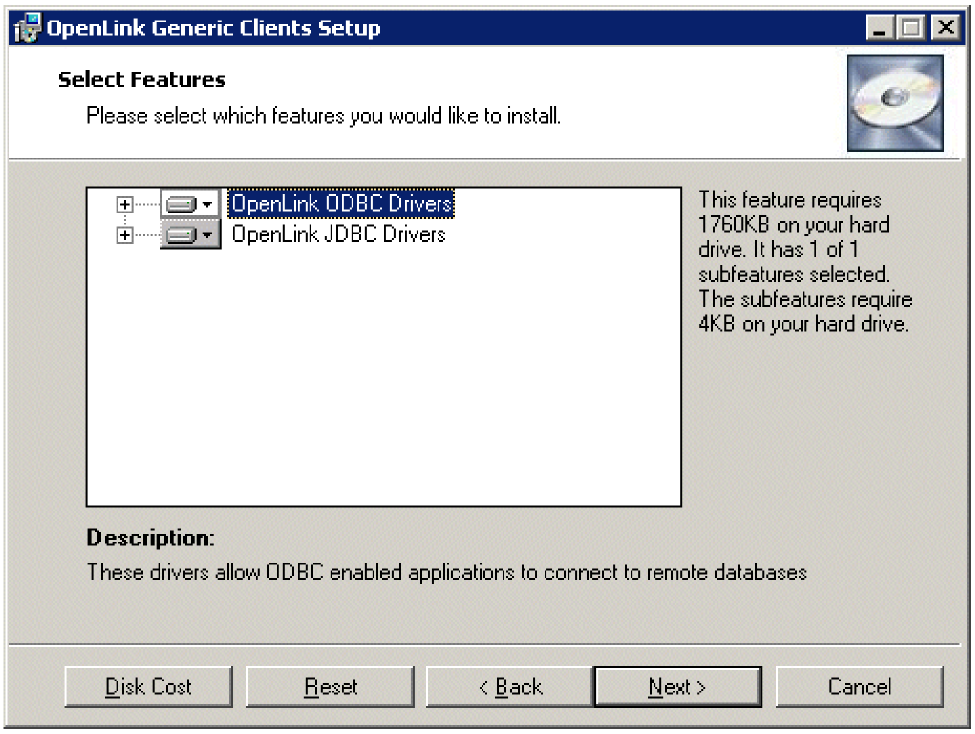
OpenLink Generic Clients will be installed in the following folder.

To continue, click Next. If you would like to select a different folder, click Browse.

C:\Program Files\OpenLink Software\UDA\

Next you have presented with the option to choose which components are to be installed:

Figure 3.5. win64instmt5.gif


OpenLink Generic Clients Setup

Select Features

Please select which features you would like to install.

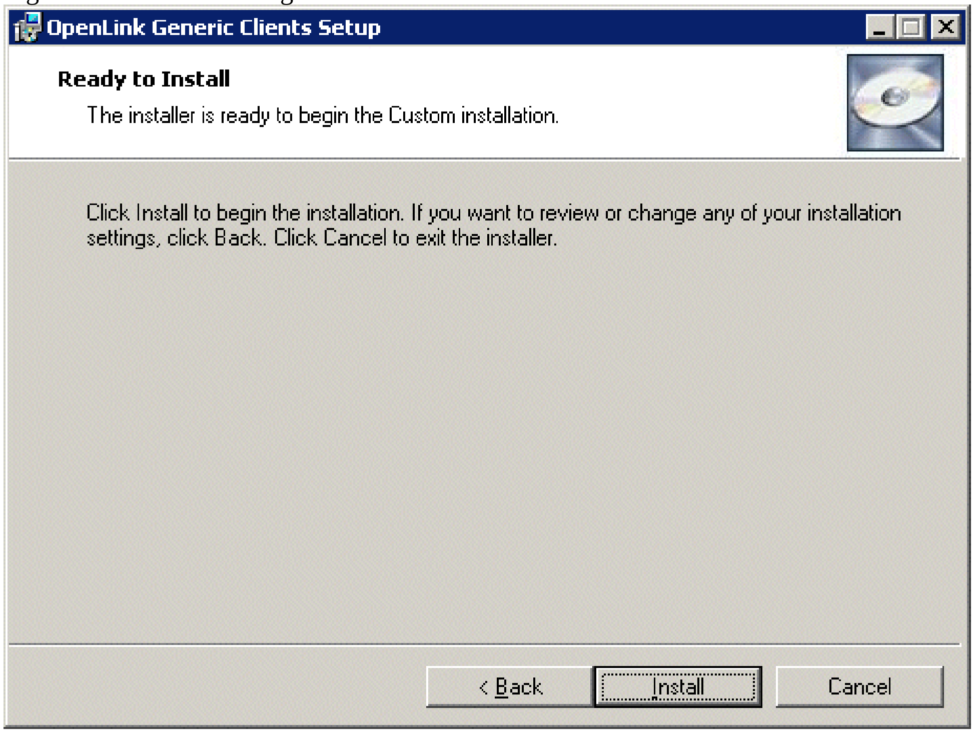
OpenLink ODBC Drivers
OpenLink JDBC Drivers

This feature requires 1760KB on your hard drive. It has 1 of 1 subfeatures selected. The subfeatures require 4KB on your hard drive.

Description:
These drivers allow ODBC enabled applications to connect to remote databases

A final confirmation screen is displayed for review prior to performing the installation:

Figure 3.6. win64instmt6.gif


OpenLink Generic Clients Setup

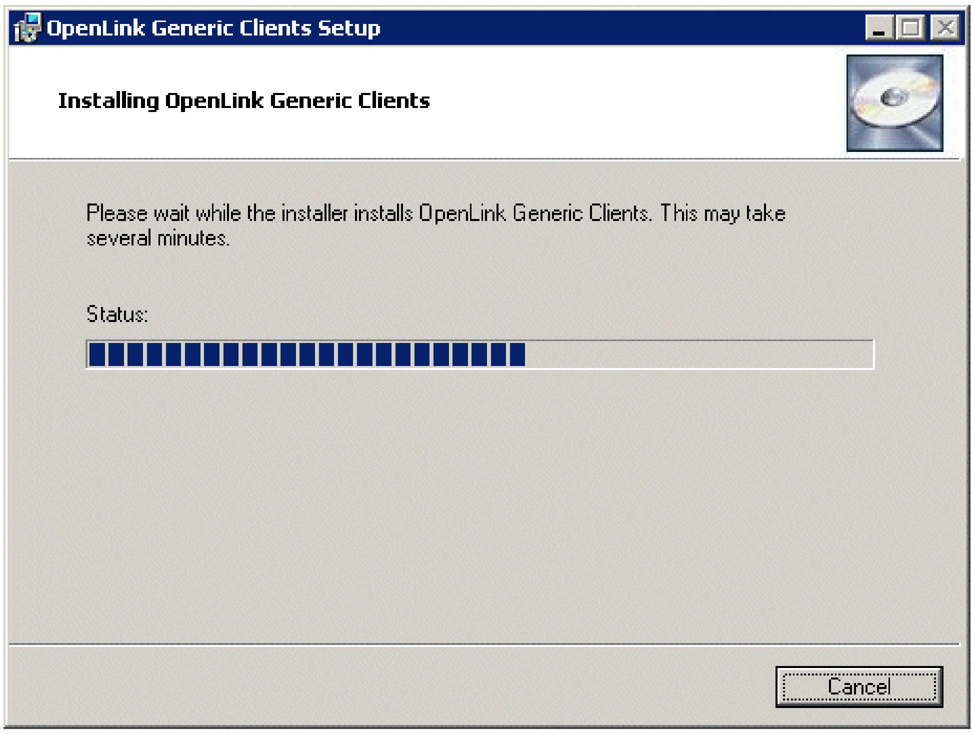
Ready to Install

The installer is ready to begin the Custom installation.

Click Install to begin the installation. If you want to review or change any of your installation settings, click Back. Click Cancel to exit the installer.

The installer status bar is then displayed indicating the progress and status of the the installation:

Figure 3.7. win64instmt7.gif



OpenLink Generic Clients Setup

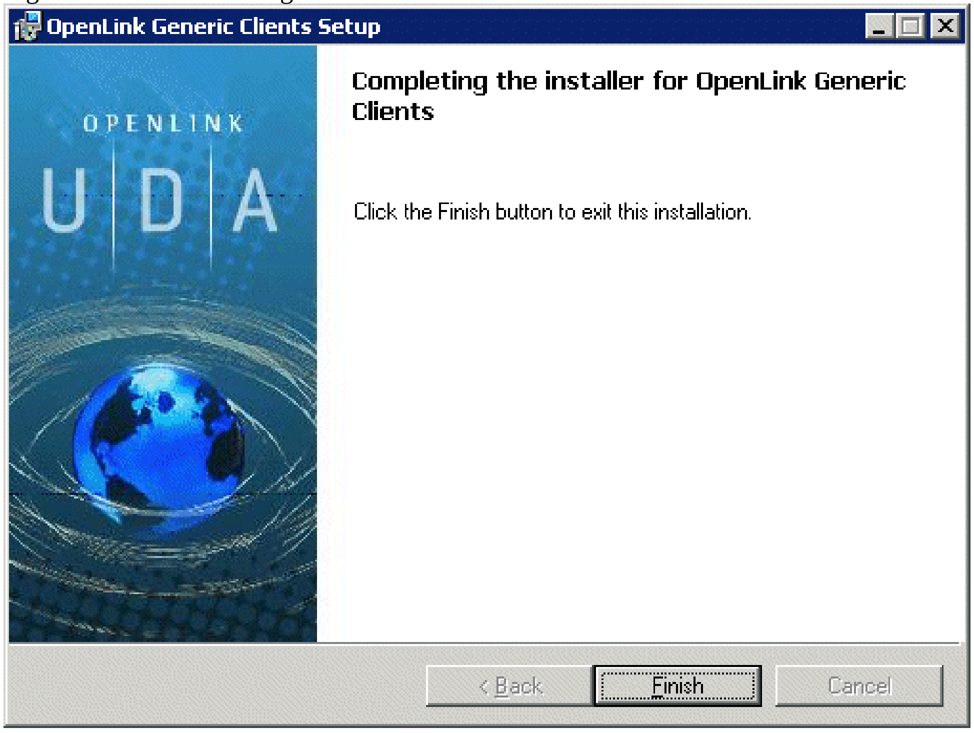
Installing OpenLink Generic Clients

Please wait while the installer installs OpenLink Generic Clients. This may take several minutes.

Status:

If the installation is successful a final dialog with be displayed indicating this:

Figure 3.8. win64instmt8.gif



OpenLink Generic Clients Setup

Completing the installer for OpenLink Generic Clients

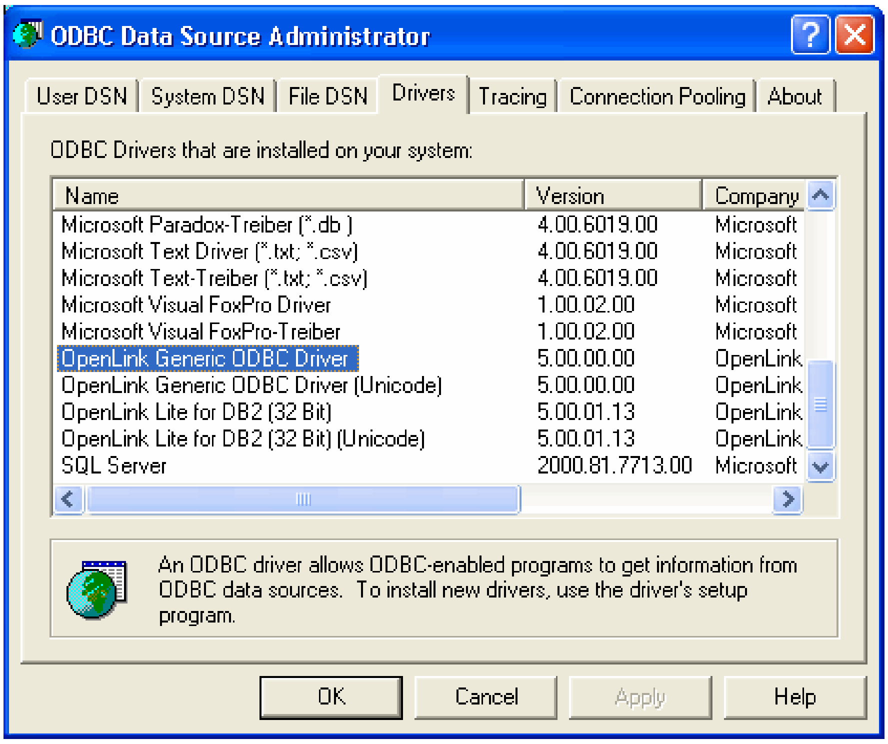
Click the Finish button to exit this installation.

5.1.3 Data Source Configuration

Once you have completed the installation process, proceed to create a new ODBC Data Source Name (DSN) using the ODBC Driver Manager.

The ODBC Driver Manager is in the Windows Control panel. Windows 2000 and Windows XP have it within Administrative tools. The figure below depicts ODBC v3.5 Driver Manager displaying a list of installed ODBC Drivers.

Figure 3.9. Data Source Configuration


ODBC Data Source Administrator

User DSN | System DSN | File DSN | Drivers | Tracing | Connection Pooling | About

ODBC Drivers that are installed on your system:

Name	Version	Company
Microsoft Paradox-Treiber (*.db)	4.00.6019.00	Microsoft
Microsoft Text Driver (*.txt; *.csv)	4.00.6019.00	Microsoft
Microsoft Text-Treiber (*.txt; *.csv)	4.00.6019.00	Microsoft
Microsoft Visual FoxPro Driver	1.00.02.00	Microsoft
Microsoft Visual FoxPro-Treiber	1.00.02.00	Microsoft
OpenLink Generic ODBC Driver	5.00.00.00	OpenLink
OpenLink Generic ODBC Driver (Unicode)	5.00.00.00	OpenLink
OpenLink Lite for DB2 (32 Bit)	5.00.01.13	OpenLink
OpenLink Lite for DB2 (32 Bit) (Unicode)	5.00.01.13	OpenLink
SQL Server	2000.81.7713.00	Microsoft

An ODBC driver allows ODBC-enabled programs to get information from ODBC data sources. To install new drivers, use the driver's setup program.

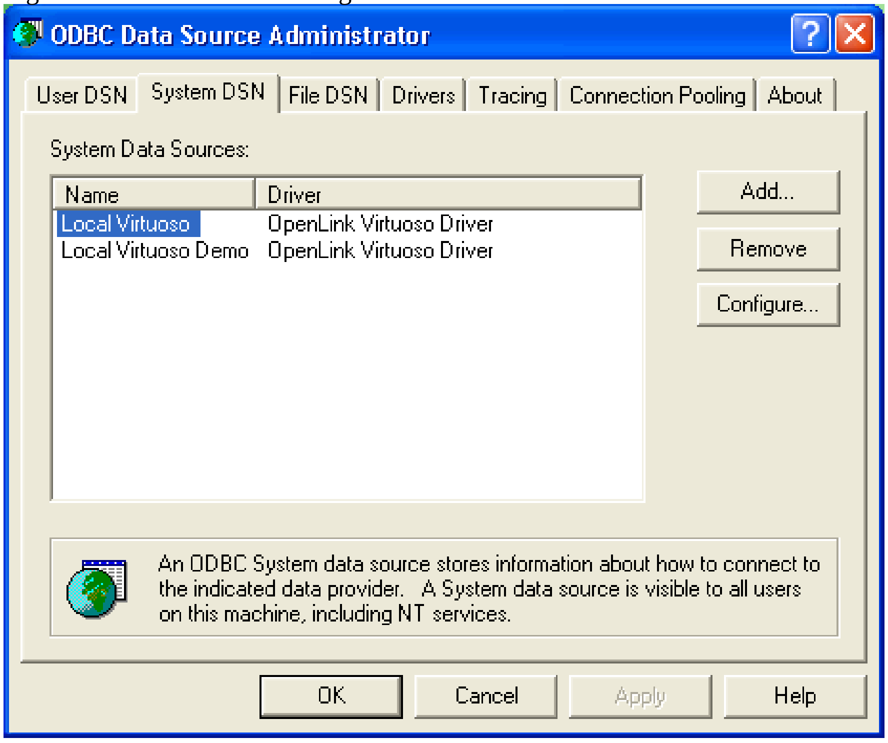
5.1.4 ODBC System & User Data Sources

ODBC Driver Manager can create User, System, and File DSNs. Here are the steps for creating a User or System DSN:

1. Decide whether the DSN should be a USER DSN so that it is available for all users, or a USER DSN is required for access by the current user only.

   In the ODBC Driver Manager pick the SYSTEM or USER tab. In this example the SYSTEM tab is chosen, and the current System DSNs are shown:

   Figure 3.10. Data Source Configuration


ODBC Data Source Administrator

User DSN | System DSN | File DSN | Drivers | Tracing | Connection Pooling | About

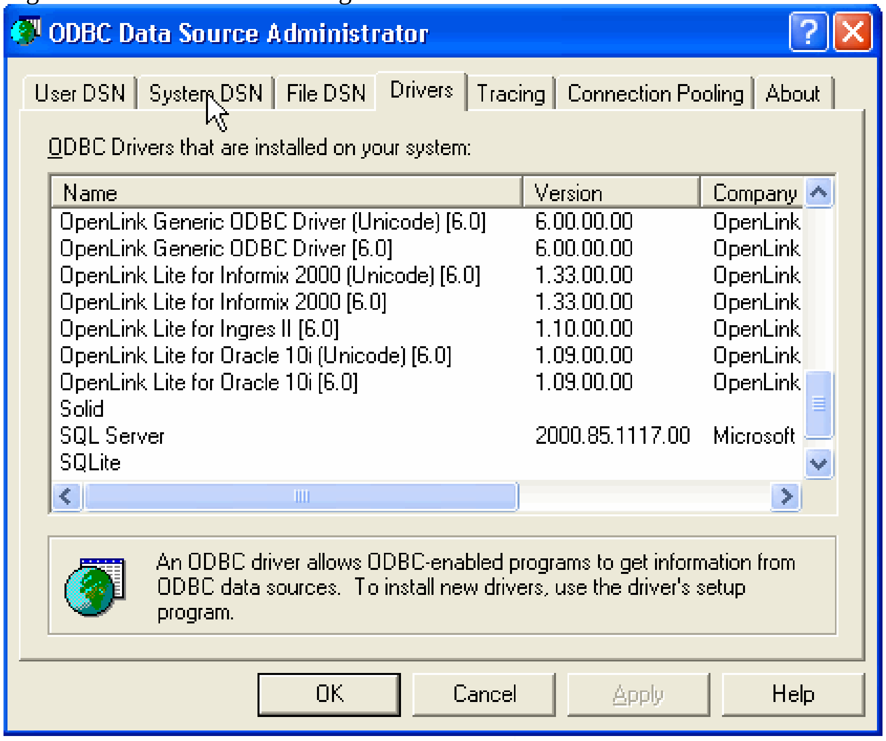
System Data Sources:

Name	Driver
Local Virtuoso	OpenLink Virtuoso Driver
Local Virtuoso Demo	OpenLink Virtuoso Driver

An ODBC System data source stores information about how to connect to the indicated data provider. A System data source is visible to all users on this machine, including NT services.

2. Pick the ODBC Driver to be used to create your ODBC DSN, in the case of the OpenLink Universal Data Access Driver Suite 5.0, this would be the Driver identified below as "OpenLink Generic ODBC Driver".

Figure 3.11. Data Source Configuration



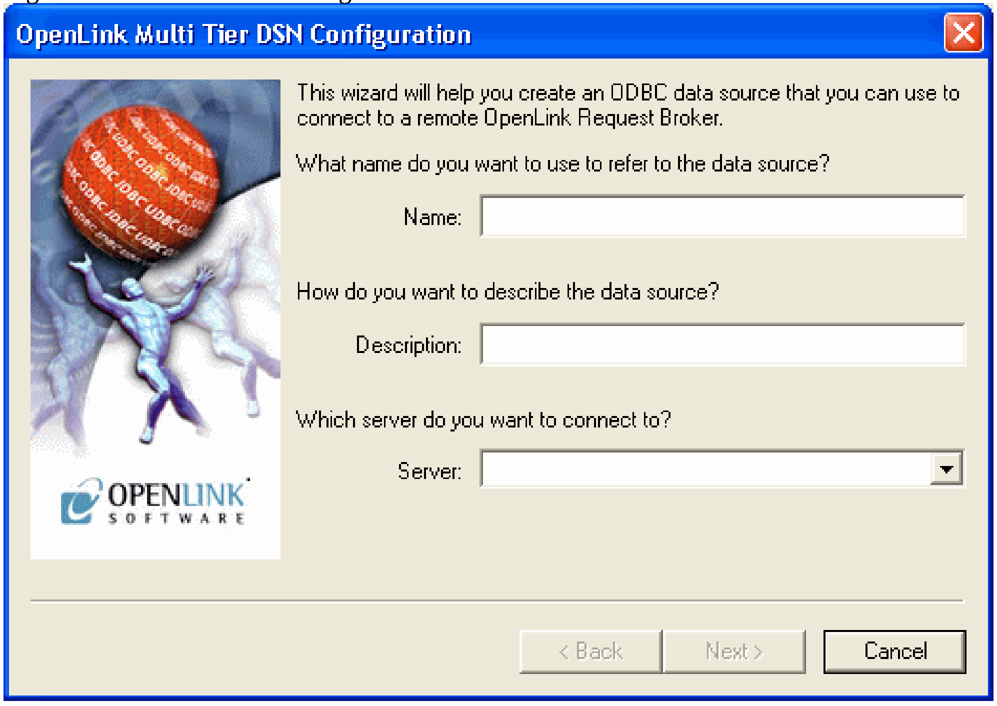
ODBC Drivers that are installed on your system:

Name	Version	Company
OpenLink Generic ODBC Driver (Unicode) [6.0]	6.00.00.00	OpenLink
OpenLink Generic ODBC Driver [6.0]	6.00.00.00	OpenLink
OpenLink Lite for Informix 2000 (Unicode) [6.0]	1.33.00.00	OpenLink
OpenLink Lite for Informix 2000 [6.0]	1.33.00.00	OpenLink
OpenLink Lite for Ingres II [6.0]	1.10.00.00	OpenLink
OpenLink Lite for Oracle 10i (Unicode) [6.0]	1.09.00.00	OpenLink
OpenLink Lite for Oracle 10i [6.0]	1.09.00.00	OpenLink
Solid		
SQL Server	2000.85.1117.00	Microsoft
SQLite		

An ODBC driver allows ODBC-enabled programs to get information from ODBC data sources. To install new drivers, use the driver's setup program.

3. The ODBC Driver manager now launches the selected driver code so that the DSN may be configured. In this example it will present the OpenLink ODBC Data Source Configuration dialog (depicted below).

Figure 3.12. Data Source Configuration



This wizard will help you create an ODBC data source that you can use to connect to a remote OpenLink Request Broker.

What name do you want to use to refer to the data source?

Name:

How do you want to describe the data source?

Description:

Which server do you want to connect to?

Server:

Configure the fields of the ODBC "System" or "User" DSN Configuration dialog in the manner explained below:

Name. (Datasource) The name of the ODBC DSN, this is how you will interact with the OpenLink ODBC Driver from within ODBC compliant applications once your ODBC DSN has been created.

Description. Additional information that further describes the ODBC DSN that you are creating.

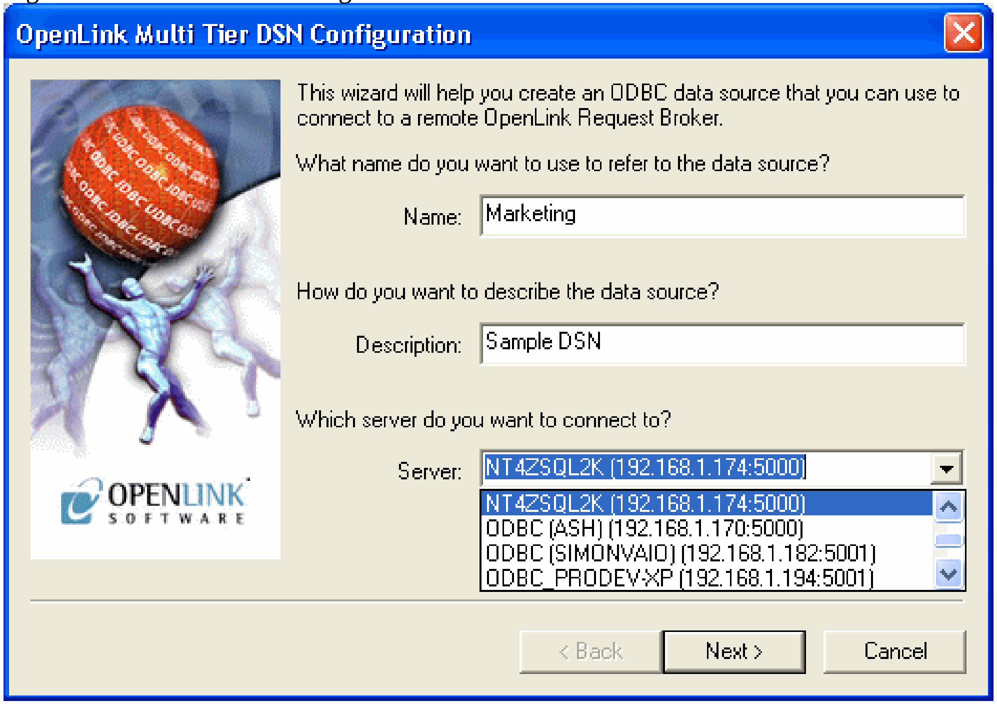
Server. Selected Zero Configuration service name or `<hostname or IP address>:<port>` of running Broker.

The easiest way to complete this field is to select a machine from the drop-down list-box. This list of servers is automatically generated by the Zero Configuration service, so you only need to chose which machine you wish to reach.

Alternatively, enter the hostname or IP address that identifies a Server Machine running OpenLink Server Components, that speak the OpenLink Data Access Protocol. Follow this with a colon and the TCP/IP Port number of the Broker to contact. This corresponds to the Listen parameter on the target Broker.

This example shall now assume the selection of a server from a picklist.

Figure 3.13. Data Source Configuration


OpenLink Multi Tier DSN Configuration

This wizard will help you create an ODBC data source that you can use to connect to a remote OpenLink Request Broker.

What name do you want to use to refer to the data source?

Name: Marketing

How do you want to describe the data source?

Description: Sample DSN

Which server do you want to connect to?

Server: NT4ZSQL2K (192.168.1.174:5000)

NT4ZSQL2K (192.168.1.174:5000)
ODBC (ASH) (192.168.1.170:5000)
ODBC (SIMONVAIO) (192.168.1.182:5001)
ODBC_PRODEV-XP (192.168.1.194:5001)

4. Now select the database and configuration details:

5.1.4.1 Domain

This is how you pick the Database Engine Type that your ODBC DSN is to be associated with e.g. Informix 7, Oracle 7, Progress 7 etc.

The default offered from the Zero Configuration is typically accepted. An alternative compatable Domain may be chosen for a specific customisation.

Database. This is how an actual database name within the Provider Type Domain is identified, for instance "stores7" indicates an "Informix 7" database called "stores7". This option corresponds to the Database parameter in the preference files.

Connection Options. This is where you place any database specific database connection options. This field in a majority of cases should be left blank by default.

5.1.4.2 Connect now..

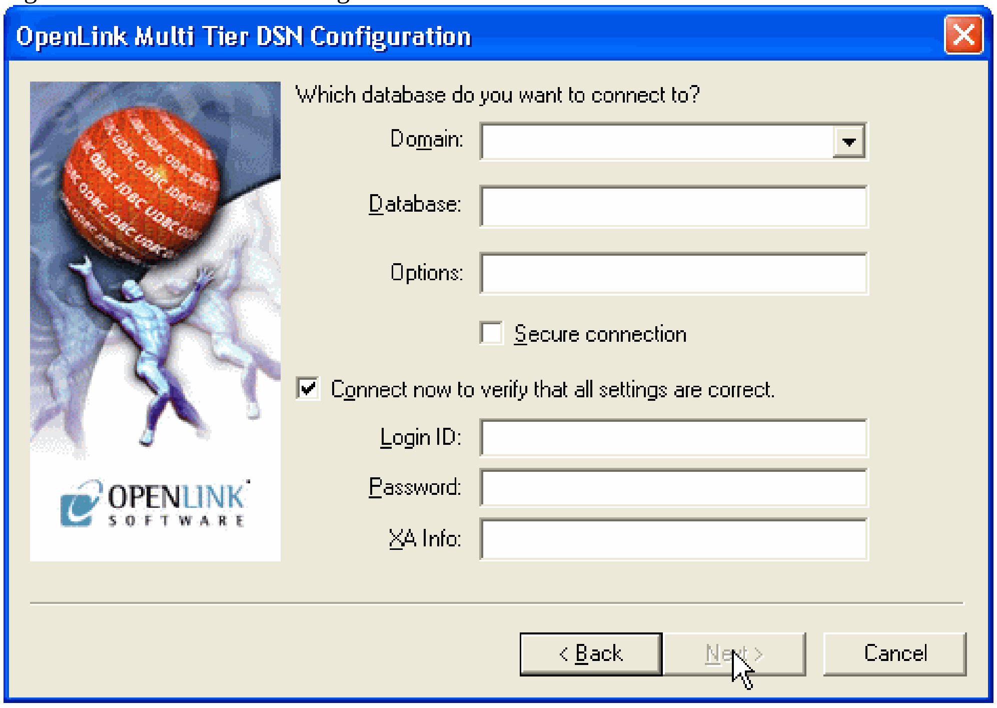
When this tick box is checked, a test connection is made to verify the Data Source connection.

If there is no check then the Login ID and Password fields are ignored, and no test is performed.

Login ID. The default database UserID to use when logging on to a remote database engine (identified by the Domain above).

Password. The Password for the login of the above UserID.

Figure 3.14. Data Source Configuration



OpenLink Multi Tier DSN Configuration

Which database do you want to connect to?

Domain:

Database:

Options:

☐ Secure connection

☑ Connect now to verify that all settings are correct.

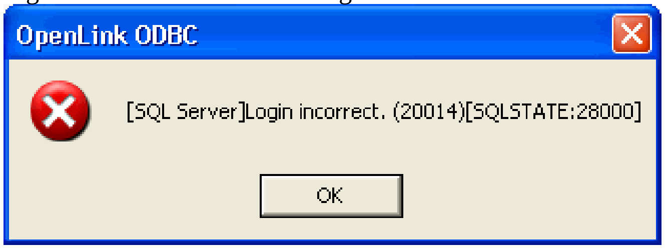
Login ID:

Password:

XA Info:

5. When a test connection fails, the error message is now displayed:

Figure 3.15. Data Source Configuration



OpenLink ODBC

[SQL Server]Login incorrect. (20014)[SQLSTATE:28000]

6. Now define additonal connection parameters:

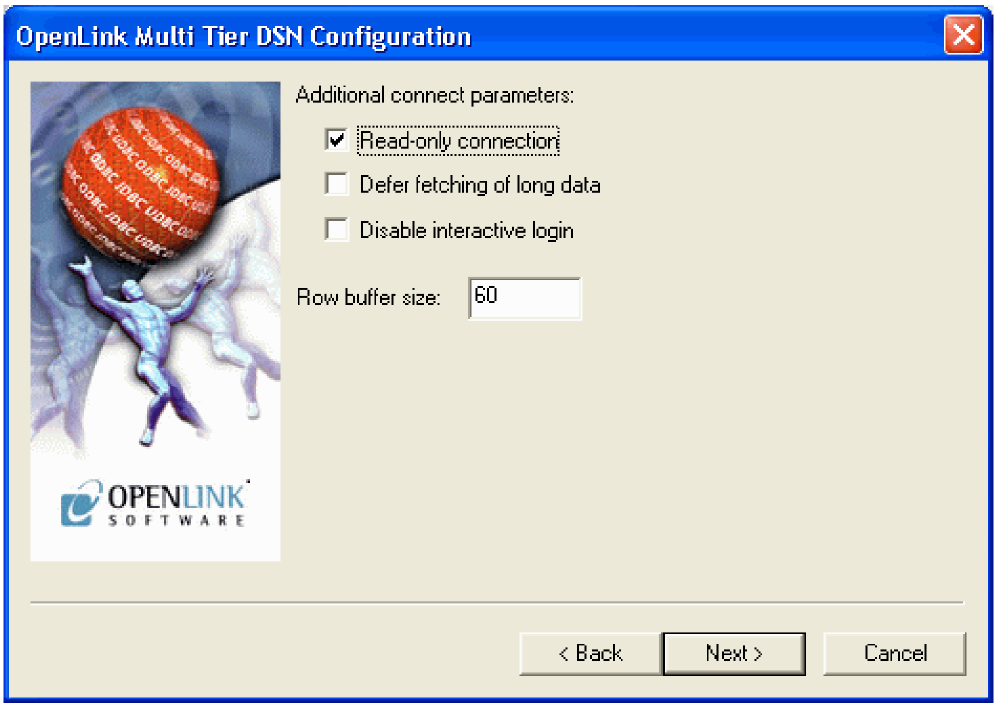
Read-only connection. Specify whether the connection is to be "Read-only". Make sure the checkbox is unchecked to request a "Read/Write" connection.

Defer fetching of long data. Check this box to defer the fetching of long data.

Disable interactive login. Suppress the ODBC "Username" and "Password" login dialog box when interacting with your ODBC DSN from within an ODBC compliant application.

Row Buffer Size. This attribute specifies the number of records to be transported over the network in a single network hop. Values can range from 1 to 99.

Figure 3.16. Data Source Configuration



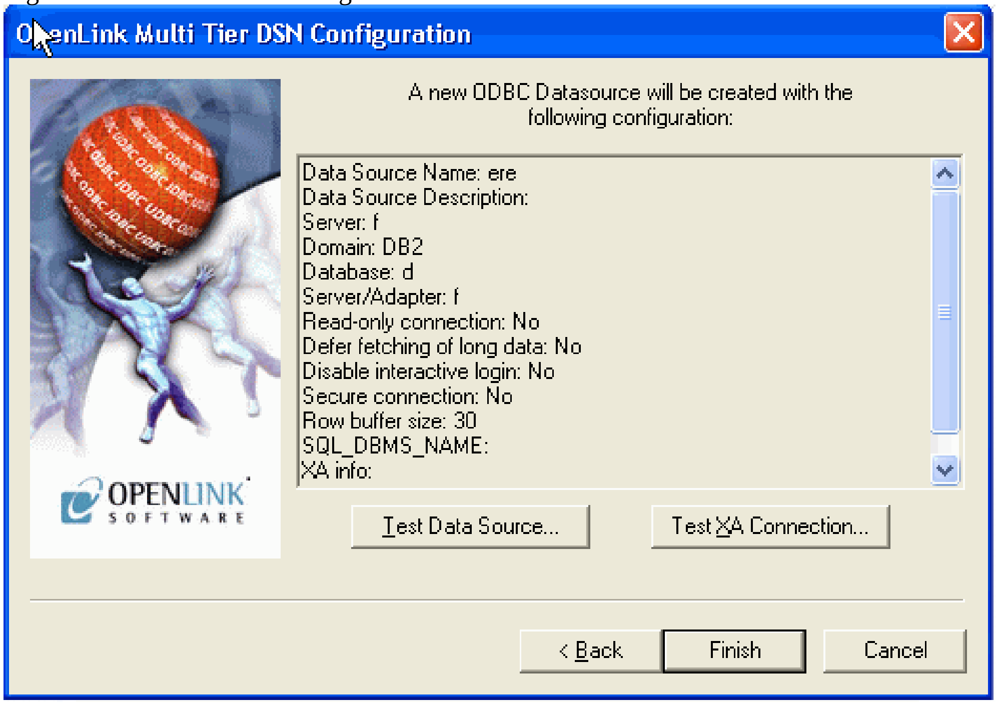
Additional connect parameters:

[✓] Read-only connection
[ ] Defer fetching of long data
[ ] Disable interactive login

Row buffer size: 60

7. The list of parameters for the DSN are now shown. The Test Data Source button will trigger a test of the DSN using the existing login parameters.

Figure 3.17. Data Source Configuration

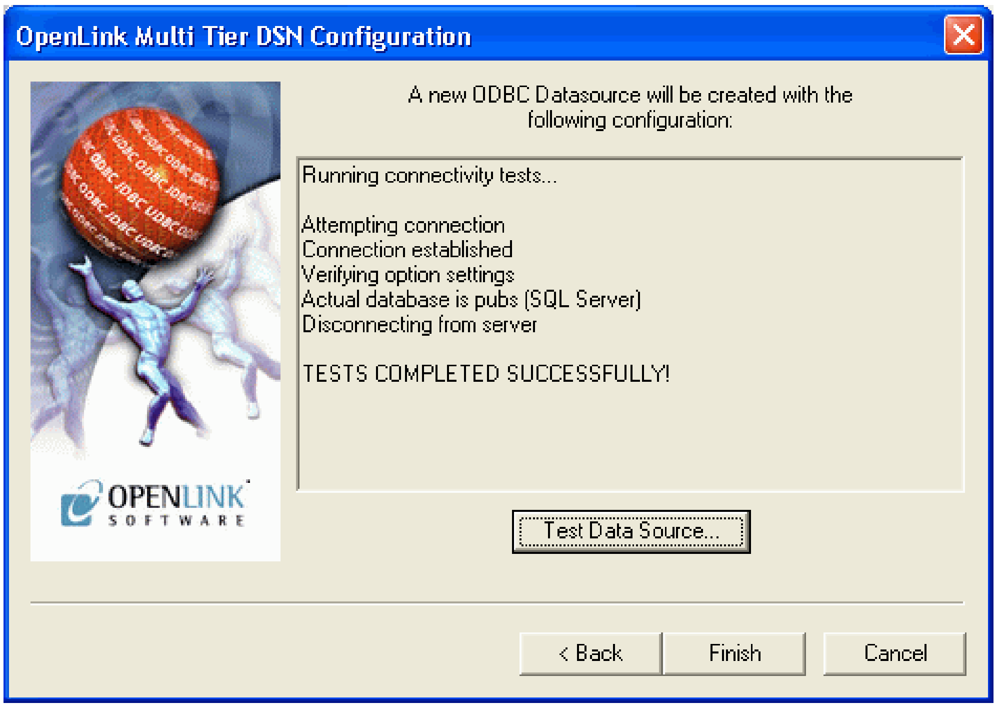


A new ODBC Datasource will be created with the following configuration:

```
Data Source Name: ere
Data Source Description:
Server: f
Domain: DB2
Database: d
Server/Adapter: f
Read-only connection: No
Defer fetching of long data: No
Disable interactive login: No
Secure connection: No
Row buffer size: 30
SQL_DBMS_NAME:
XA info:
```

8. If a test is requested, the results are shown:

Figure 3.18. Data Source Configuration



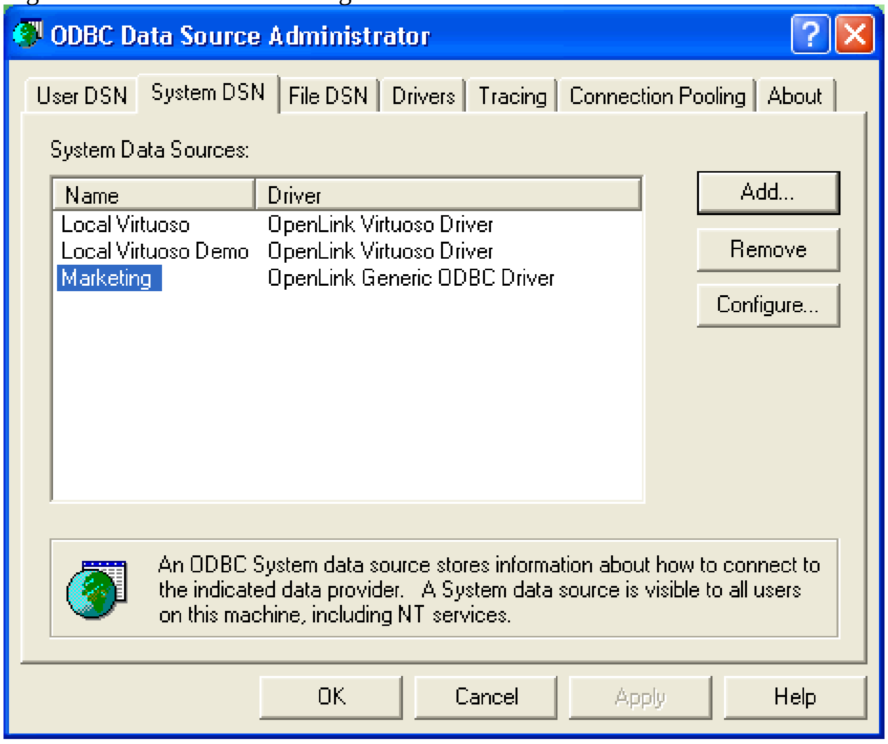
A new ODBC Datasource will be created with the following configuration:

```
Running connectivity tests...

Attempting connection
Connection established
Verifying option settings
Actual database is pubs (SQL Server)
Disconnecting from server

TESTS COMPLETED SUCCESSFULLY!
```

Press the Finish button to save the DSN.

9. This completes the DSN configuration. It will be listed in the ODBC Driver Manager.

Figure 3.19. Data Source Configuration



System Data Sources:

Name	Driver
Local Virtuoso	OpenLink Virtuoso Driver
Local Virtuoso Demo	OpenLink Virtuoso Driver
Marketing	OpenLink Generic ODBC Driver

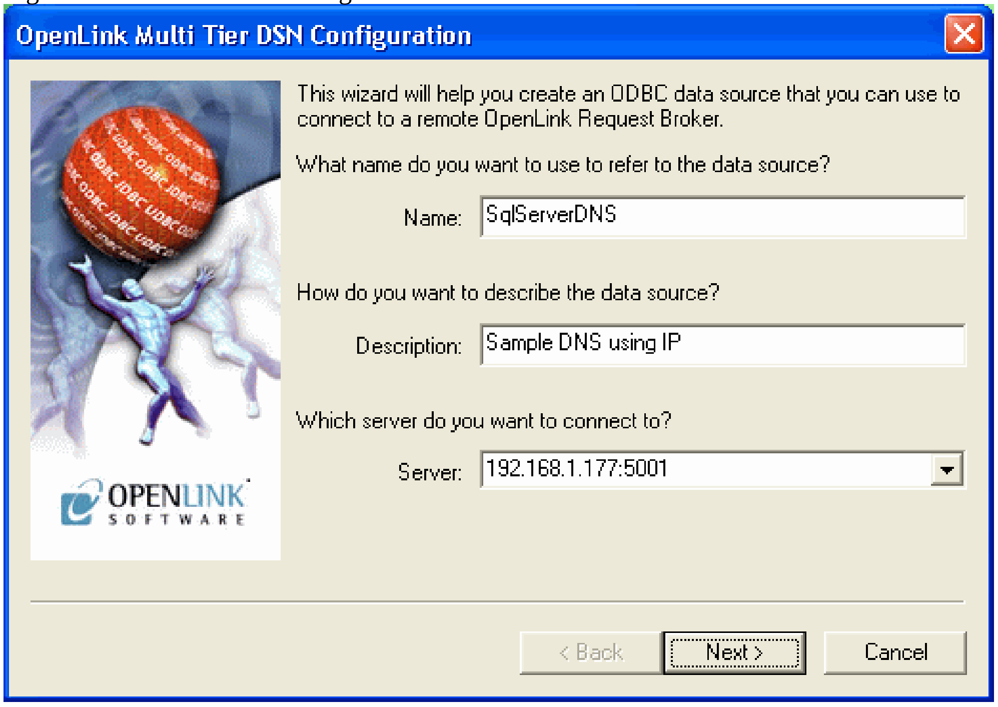
An ODBC System data source stores information about how to connect to the indicated data provider. A System data source is visible to all users on this machine, including NT services.

5.1.4.3 Manual Selection of Server

The previous example used a server that was picked from the list constructed by the zero configuration. The server you wish to use might not be in this list if it is not a local machine that can be discovered. In this case it is necessary to complete the server field manually with the host name or IP number, and the port number.

1. Complete the fields for Data Source Name and Description as before. In the server field, enter the TCPIP address and port of the server machine.

Figure 3.20. Data Source Configuration



OpenLink Multi Tier DSN Configuration

This wizard will help you create an ODBC data source that you can use to connect to a remote OpenLink Request Broker.

What name do you want to use to refer to the data source?

Name: SqlServerDNS

How do you want to describe the data source?

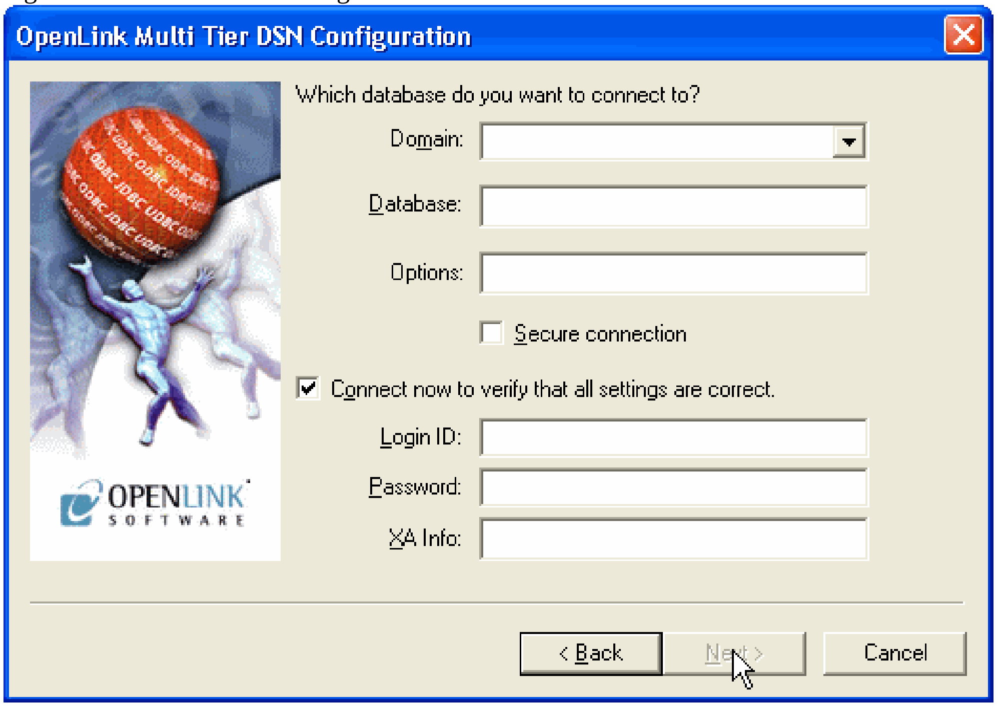
Description: Sample DNS using IP

Which server do you want to connect to?

Server: 192.168.1.177:5001

2. The connection parameter page will load without any fields completed.

Figure 3.21. Data Source Configuration



OpenLink Multi Tier DSN Configuration

Which database do you want to connect to?

Domain:

Database:

Options:

☐ Secure connection

☑ Connect now to verify that all settings are correct.

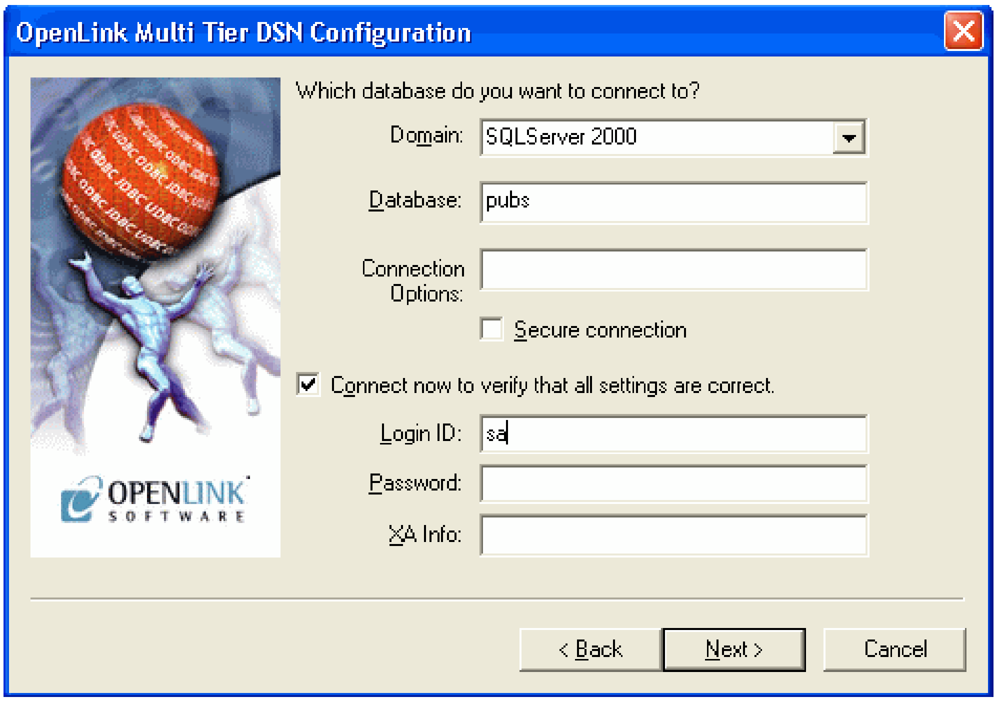
Login ID:

Password:

XA Info:

Complete the fields for database connection.

Figure 3.22. Data Source Configuration


OpenLink Multi Tier DSN Configuration

Which database do you want to connect to?

Domain: SQLServer 2000

Database: pubs

Connection Options:

☐ Secure connection

☑ Connect now to verify that all settings are correct.

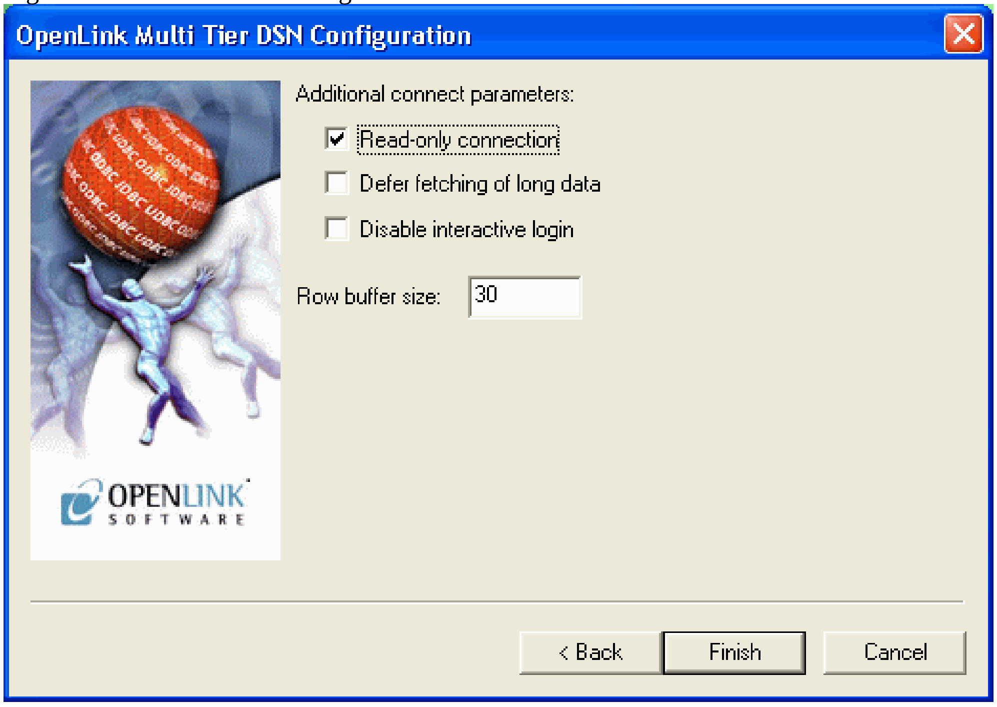
Login ID: sa

Password:

XA Info:

3. Complete the fields for additional connection parameters.

Figure 3.23. Data Source Configuration


OpenLink Multi Tier DSN Configuration

Additional connect parameters:

☑ Read-only connection

☐ Defer fetching of long data

☐ Disable interactive login

Row buffer size: 30

4. This completes the DSN configuration. It will be listed in the ODBC Driver Manager.

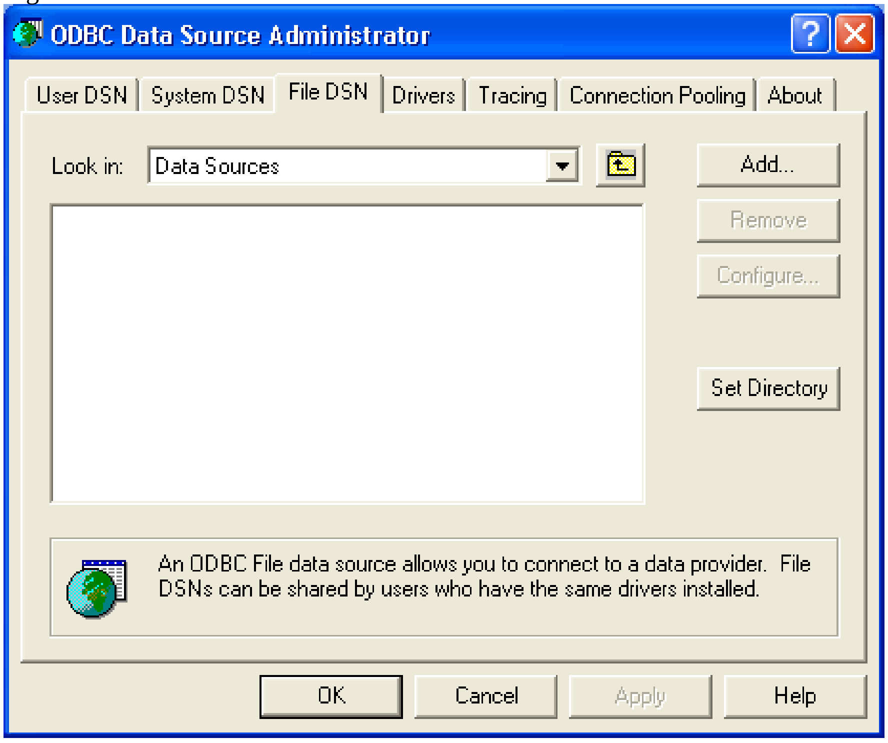
5.1.5 Creating a File Data Source

This format of ODBC DSN enables the creation of centralized ODBC DSNs on a designated NT or Windows 95 Server machine, thereby reducing the administrative overhead of ODBC DSN configuration for every single machine within your infrastructure.

The steps for creating an ODBC File DSN are as follows:

1. Select the File DSN Tab from the ODBC Driver Manger. Press the Add button.

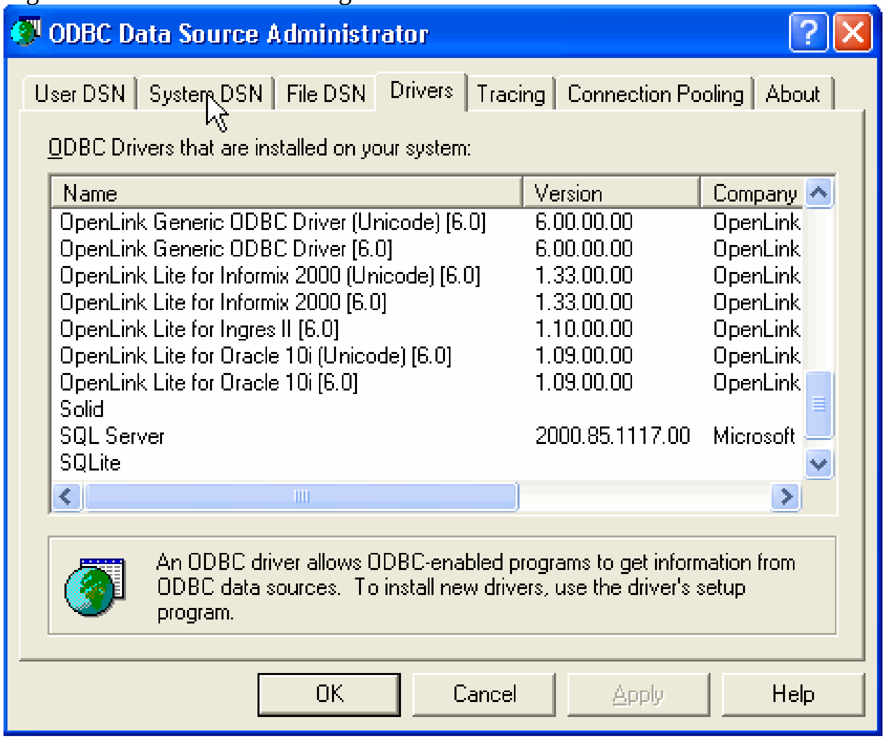
Figure 3.24. File Data Sources



An ODBC File data source allows you to connect to a data provider. File DSNs can be shared by users who have the same drivers installed.

2. Pick the ODBC Driver to be used to create your ODBC DSN, in the case of the OpenLink Universal Data Access Driver Suite 5.0, this would be the Driver identified below as "OpenLink Generic ODBC Driver".

Figure 3.25. Data Source Configuration



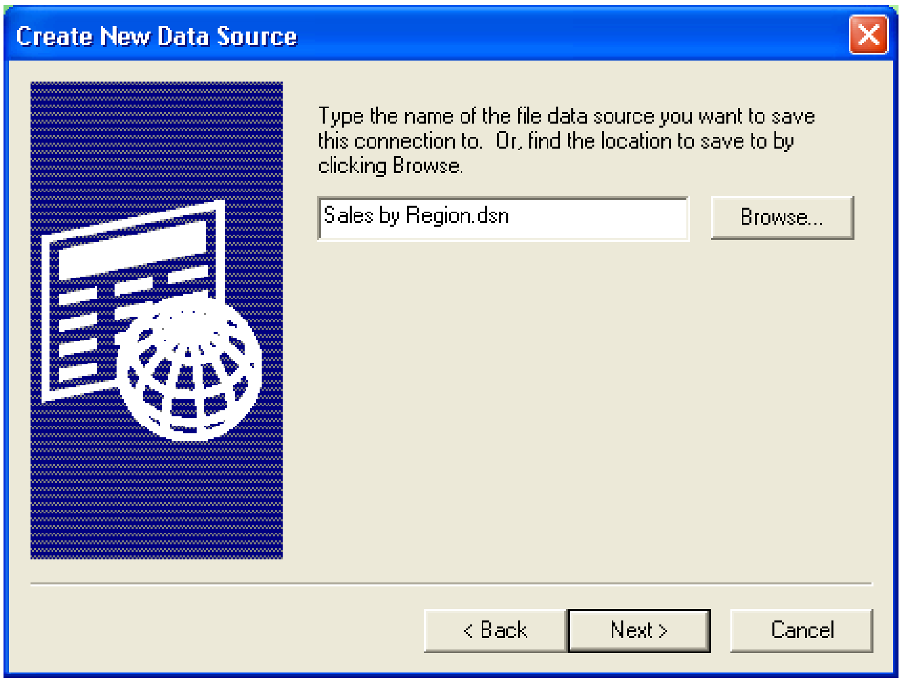
ODBC Drivers that are installed on your system:

Name	Version	Company
OpenLink Generic ODBC Driver (Unicode) [6.0]	6.00.00.00	OpenLink
OpenLink Generic ODBC Driver [6.0]	6.00.00.00	OpenLink
OpenLink Lite for Informix 2000 (Unicode) [6.0]	1.33.00.00	OpenLink
OpenLink Lite for Informix 2000 [6.0]	1.33.00.00	OpenLink
OpenLink Lite for Ingres II [6.0]	1.10.00.00	OpenLink
OpenLink Lite for Oracle 10i (Unicode) [6.0]	1.09.00.00	OpenLink
OpenLink Lite for Oracle 10i [6.0]	1.09.00.00	OpenLink
Solid		
SQL Server	2000.85.1117.00	Microsoft
SQLite		

An ODBC driver allows ODBC-enabled programs to get information from ODBC data sources. To install new drivers, use the driver's setup program.

3. Enter a name that describes your File DSN (e.g. "Sales by Region" as depicted below) for future use when interacting with it from ODBC compliant application and environments. At present to get full compatability within the Admin Assistant, you must end your dsn name with the extension ".dsn". So for this example the DSN name would be "Sales by Region.dsn".

Figure 3.26. File Data Sources

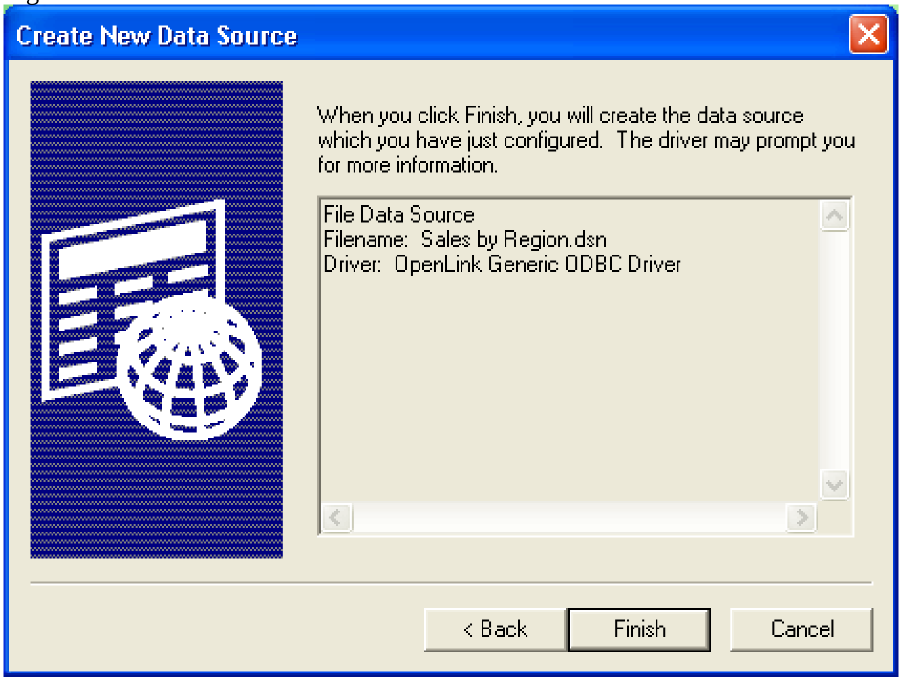


Type the name of the file data source you want to save this connection to. Or, find the location to save to by clicking Browse.

Sales by Region.dsn

4. Verify the displayed paramters are correct, and confirm by hitting Finish button.

Figure 3.27. File Data Sources



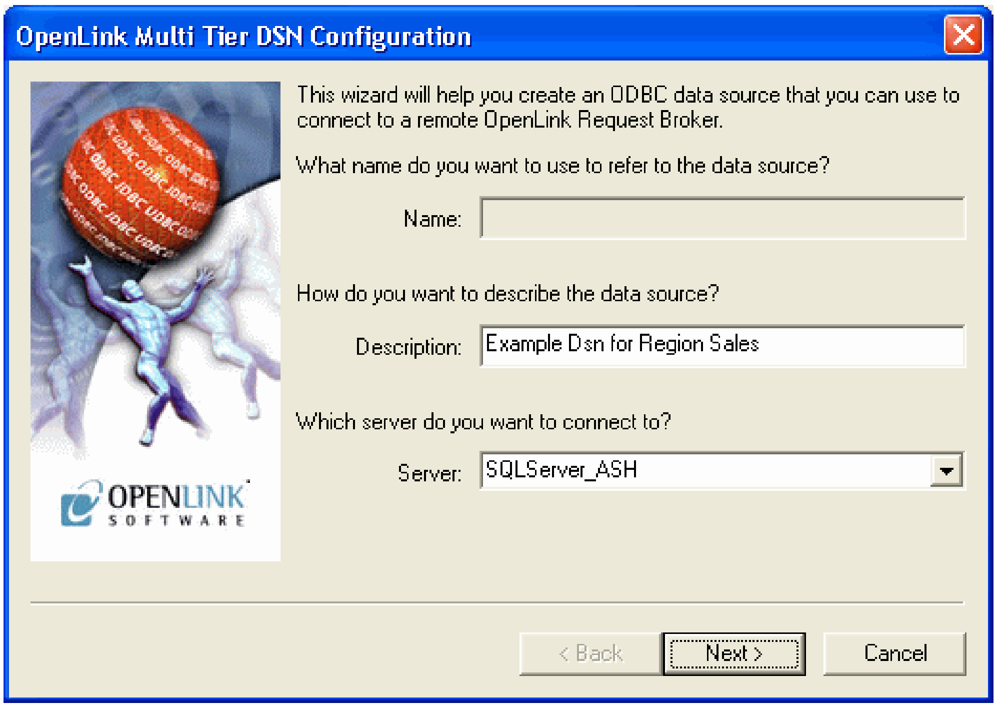
When you click Finish, you will create the data source which you have just configured. The driver may prompt you for more information.

File Data Source
Filename: Sales by Region.dsn
Driver: OpenLink Generic ODBC Driver

5. Enter a Description and define the location of the server either by selection, or enter it manually.

Note the name is not defined here as the filename defines the DSN.

Figure 3.28. File Data Sources


OpenLink Multi Tier DSN Configuration

This wizard will help you create an ODBC data source that you can use to connect to a remote OpenLink Request Broker.

What name do you want to use to refer to the data source?

Name:

How do you want to describe the data source?

Description: Example Dsn for Region Sales

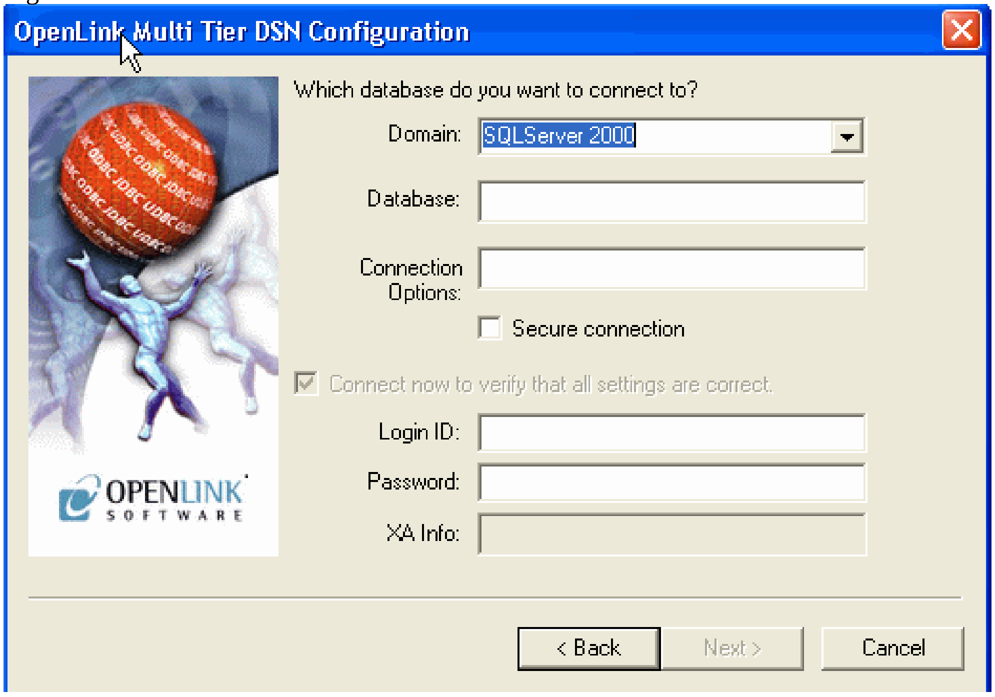
Which server do you want to connect to?

Server: SQLServer_ASH

6. Complete the fields of the ODBC File DSN. Enter Domain, Database, Server. See above for details of each.

   The File DSN must be verified immediately, so the Login ID and password are required. The Next button will be disabled until login details are provided.

   Figure 3.29. File Data Sources


OpenLink Multi Tier DSN Configuration

Which database do you want to connect to?

Domain: SQLServer 2000

Database:

Connection Options:

☐ Secure connection

☑ Connect now to verify that all settings are correct.

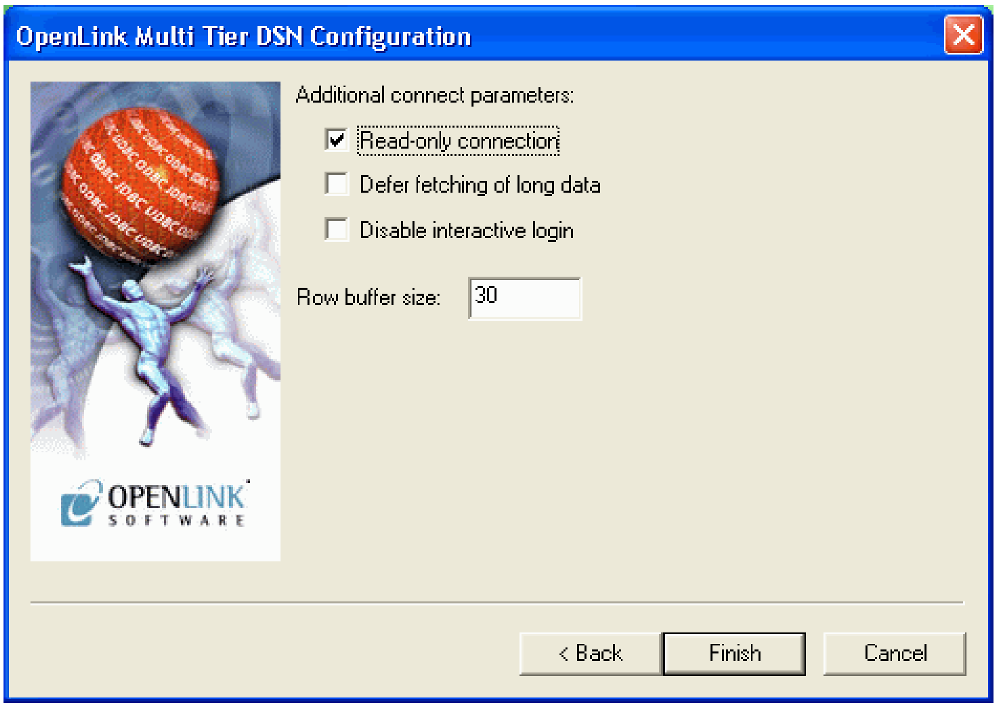
Login ID:

Password:

XA Info:

7. Complete the fields for additional connection parameters.

   Figure 3.30. Data Source Configuration



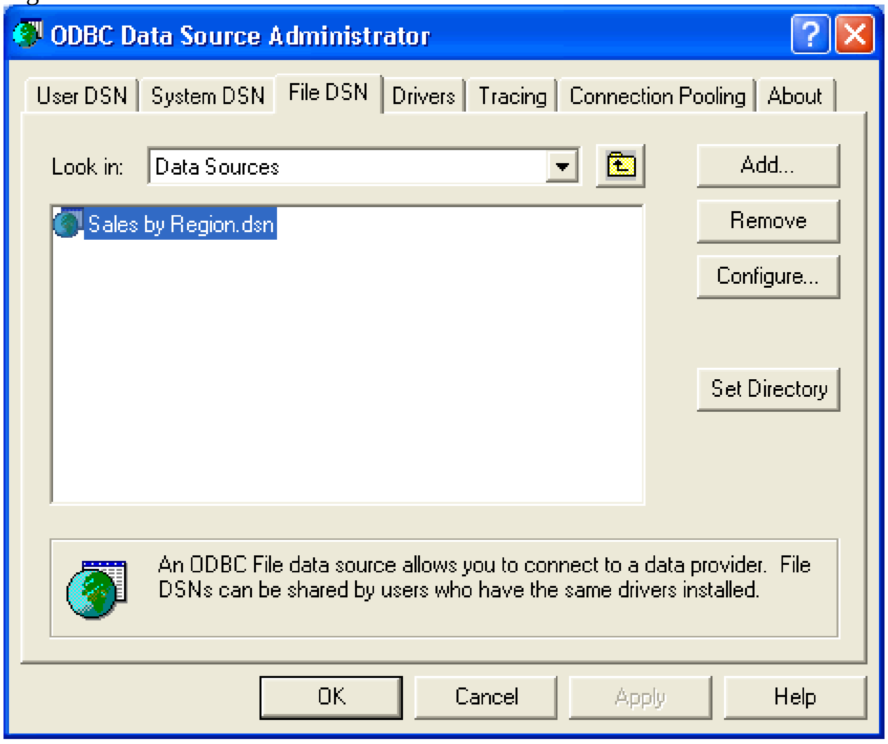
Additional connect parameters:

[✓] Read-only connection
[ ] Defer fetching of long data
[ ] Disable interactive login

Row buffer size: 30

8. This concludes the File DSN generation. The DSN will appear in the list under the File DSN Tab of the ODBC Driver Manger.

Figure 3.31. File Data Sources



Look in: Data Sources

Sales by Region.dsn

An ODBC File data source allows you to connect to a data provider. File DSNs can be shared by users who have the same drivers installed.

5.1.6 Advanced Settings

The settings in this section have default values which are recommended in most situations.

If you run a complex query through the database it may take a long time before any data is actually returned. In this situation you may need to increase the ReceiveTimeout value, described below, only to cater for it. 2 Minutes (default) is usually more than adequate.

5.1.6.1 32bit Client Advanced Settings.

Client global configuration information is contained in the sub section OPENLINK.INI of the registry. Windows 32bit platforms use the registry to store all configuration information. To edit or view to registry, run the regedit program. To view the OpenLink settings expand the following section:

```
        HKEY_LOCAL_MACHINE\SOFTWARE\ODBC\OPENLINK.INI
```

Important values that you may need to change are listed in the table below. The default values are recommended in most cases:

Table 3.1.

Sub Section	Key	Description
Communications		
	BrokerTimeout	The amount of time the client will wait for a connection to an agent. The default value is 30. Value is measured in seconds.
	ReceiveTimeout	The amount of time the client will wait for a query to return with data. The default value is 120. Value is measured in seconds.
	ShowErrors	With this set to yes any OpenLink messages will be displayed in a dialog box. If this is set to no, OpenLink messages will not be displayed.
ServerTypes		
	<Name>=	Each key name in this section represents a different provider type. To add a provider type simply add a new line using the same syntax as the default types.

5.2 Generic Client For Macintosh OS 8.6+

There are several ODBC Driver Managers distributed for the Classic Macintosh OS. OpenLink Software distributes iODBC, an Open Source, Cross-Platform, LGPL and BSDI licensed, ODBC 3.5-compliant Driver Manager maintained by OpenLink. Additional information regarding iODBC can be obtained from http://www.iodbc.org.

The iODBC Driver Manager requires Mac OS 9 or later; the OpenLink Generic Client Driver for ODBC may be used on System 7 through Mac OS 9, with any ODBC 2.x- or 3.x-compliant Driver Manager.

The OpenLink ODBC Client Components for Macintosh Classic comprise the following:

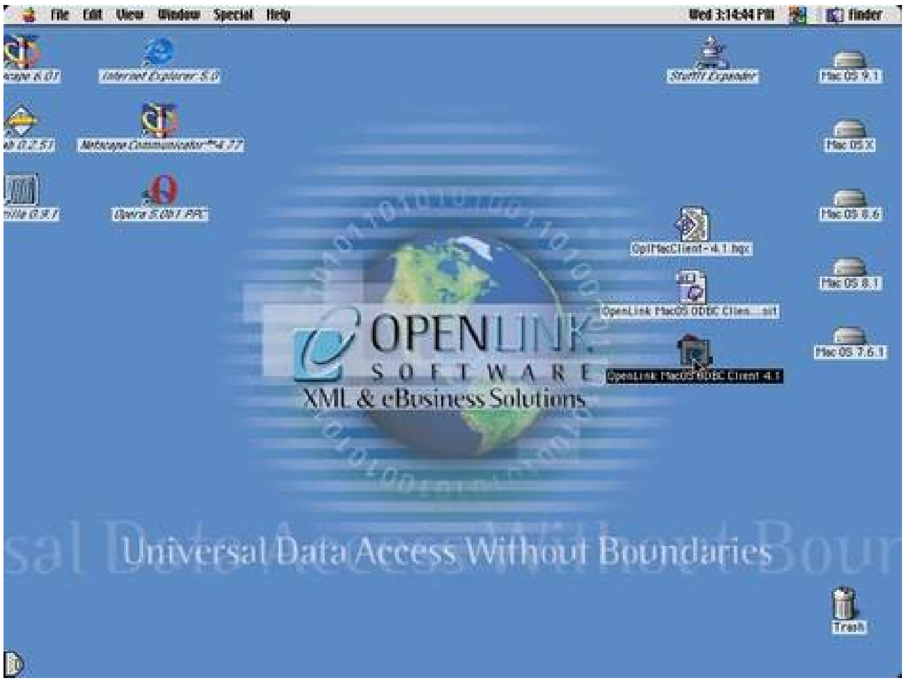
Table 3.2.

Component	**Purpose**
iODBC Driver Manager	A shared library that links ODBC compliant applications to ODBC Drivers
Generic ODBC Driver	A shared library that provides database connectivity and data access services to ODBC compliant applications
Sample ODBC Applications	Simple programs that can be used to verify your ODBC installation and working environment

5.2.1 Installation

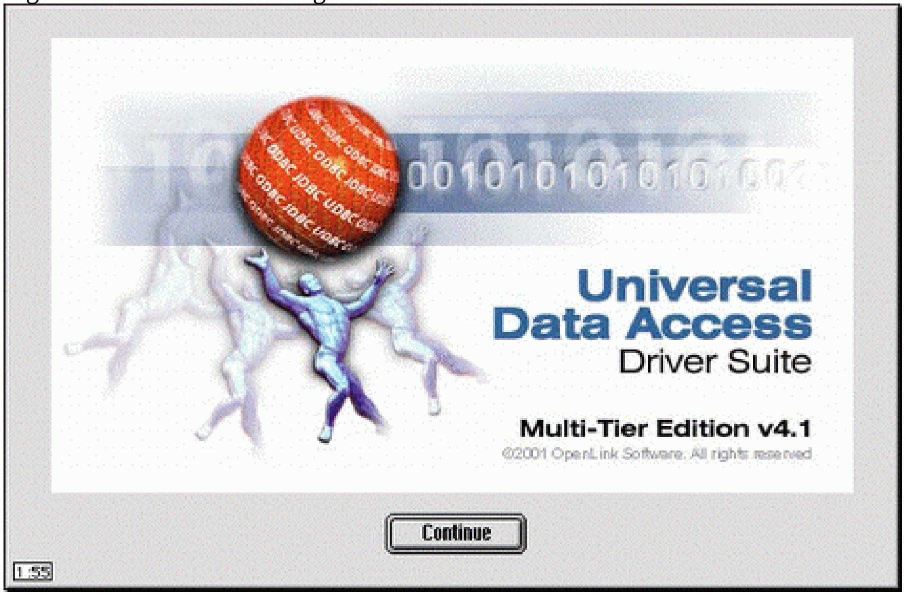
After downloading the OpenLink Data Access Driver Suite client components from our Website, uncompress the contents of the Stuffit format archive into a temporary location on your client machine. The Desktop is often best.

Figure 3.32. MacOSClass01.jpg



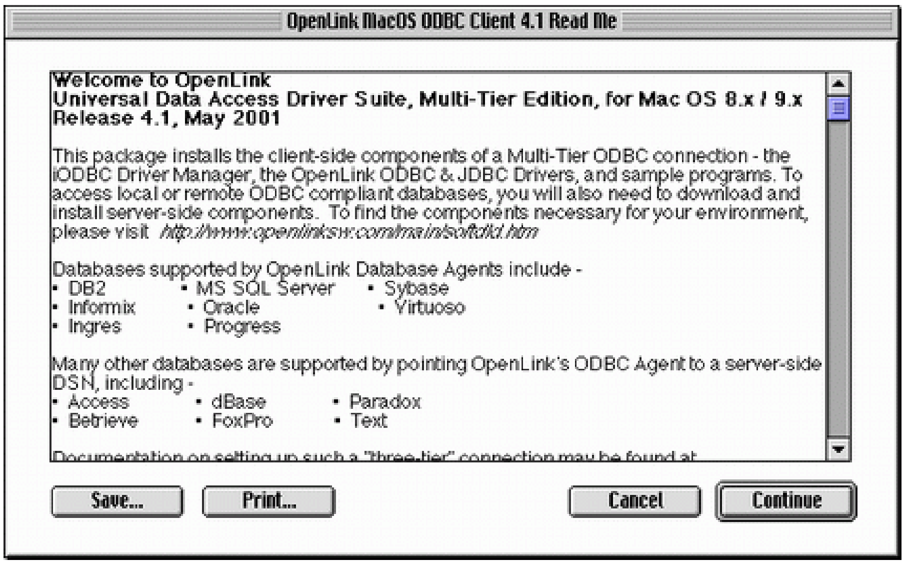
Double click the 'OpenLink MacOS ODBC Client 5.0' icon to launch the installer. Click OK at the splash screen.

Figure 3.33. MacOSClass02.gif



Check the contents of the 'Read Me'. Then click Continue.

Figure 3.34. MacOSClass03.gif



Welcome to OpenLink
Universal Data Access Driver Suite, Multi-Tier Edition, for Mac OS 8.x / 9.x
Release 4.1, May 2001

This package installs the client-side components of a Multi-Tier ODBC connection - the iODBC Driver Manager, the OpenLink ODBC & JDBC Drivers, and sample programs. To access local or remote ODBC compliant databases, you will also need to download and install server-side components. To find the components necessary for your environment, please visit *http://www.openlinksw.com/main/softdld.htm*

Databases supported by OpenLink Database Agents include -

• DB2	• MS SQL Server	• Sybase
• Informix	• Oracle	• Virtuoso
• Ingres	• Progress	

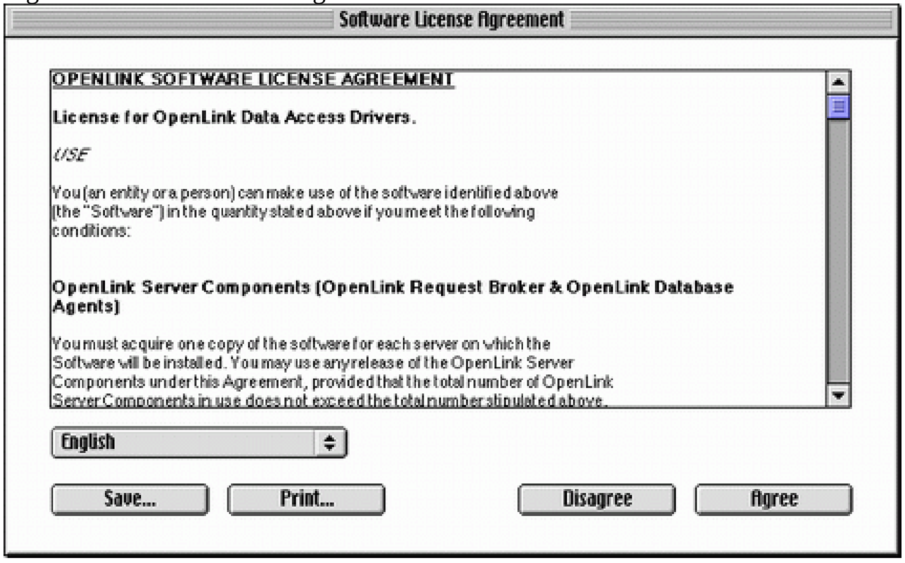
Many other databases are supported by pointing OpenLink's ODBC Agent to a server-side DSN, including -

• Access	• dBase	• Paradox
• Betrieve	• FoxPro	• Text

Documentation on setting up such a "three-tier" connection may be found at

Read and Agree to the Software License.

Figure 3.35. MacOSClass04.gif



OPENLINK SOFTWARE LICENSE AGREEMENT

License for OpenLink Data Access Drivers.

USE

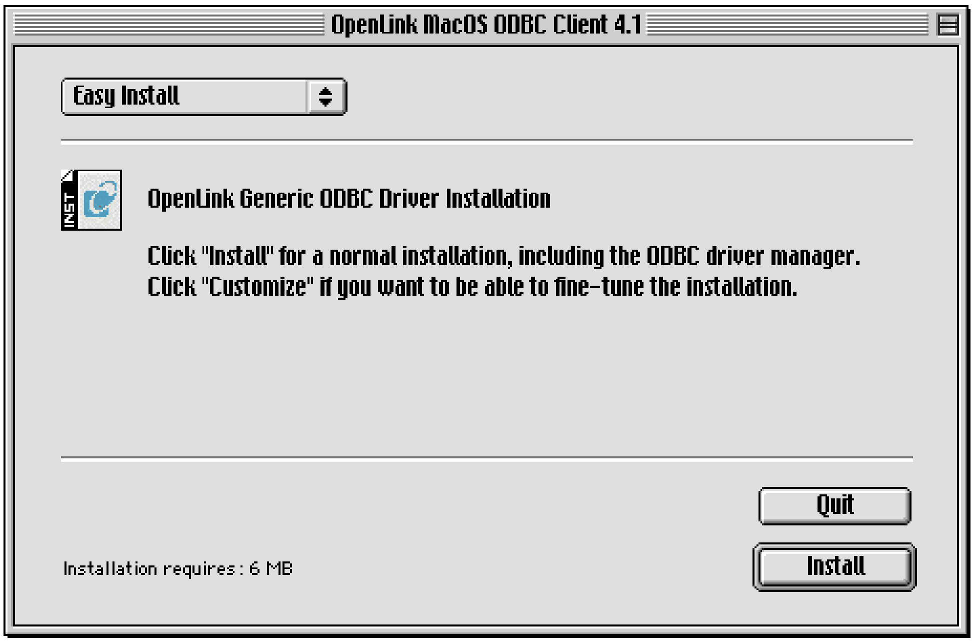
You (an entity or a person) can make use of the software identified above (the "Software") in the quantity stated above if you meet the following conditions:

OpenLink Server Components (OpenLink Request Broker & OpenLink Database Agents)

You must acquire one copy of the software for each server on which the Software will be installed. You may use any release of the OpenLink Server Components under this Agreement, provided that the total number of OpenLink Server Components in use does not exceed the total number stipulated above.

There are two types of installation. The Easy Install option is best for most users, and will install the iODBC Driver Manager, if no other Driver Manager is present.

Figure 3.36. MacOSClass05.gif



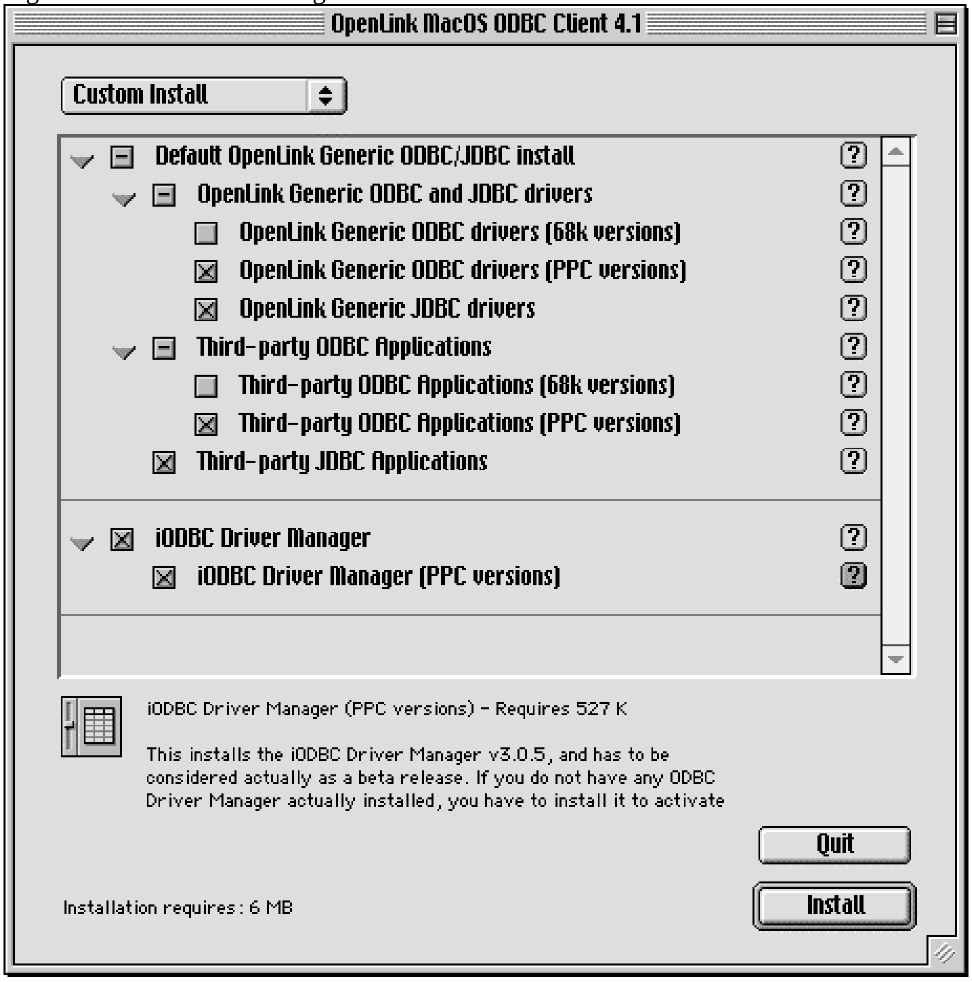
OpenLink Generic ODBC Driver Installation

Click "Install" for a normal installation, including the ODBC driver manager.
Click "Customize" if you want to be able to fine-tune the installation.

Installation requires : 6 MB

If you need to replace an older Driver Manager (Merant, Visigenic, Intersolv), select the Custom Install, and check off all appropriate components. The iODBC Driver Manager will only run on PowerPC machines, under Mac OS 9 or later. If you're installing on an older Mac, running Mac OS 8.1 or earlier, you can use Custom Install to get the 68K components you'll need.

Figure 3.37. MacOSClass06.gif



- Default OpenLink Generic ODBC/JDBC install
  - OpenLink Generic ODBC and JDBC drivers
    - ☐ OpenLink Generic ODBC drivers (68k versions)
    - ☒ OpenLink Generic ODBC drivers (PPC versions)
    - ☒ OpenLink Generic JDBC drivers
  - Third-party ODBC Applications
    - ☐ Third-party ODBC Applications (68k versions)
    - ☒ Third-party ODBC Applications (PPC versions)
  - ☒ Third-party JDBC Applications
- ☒ iODBC Driver Manager
  - ☒ iODBC Driver Manager (PPC versions)

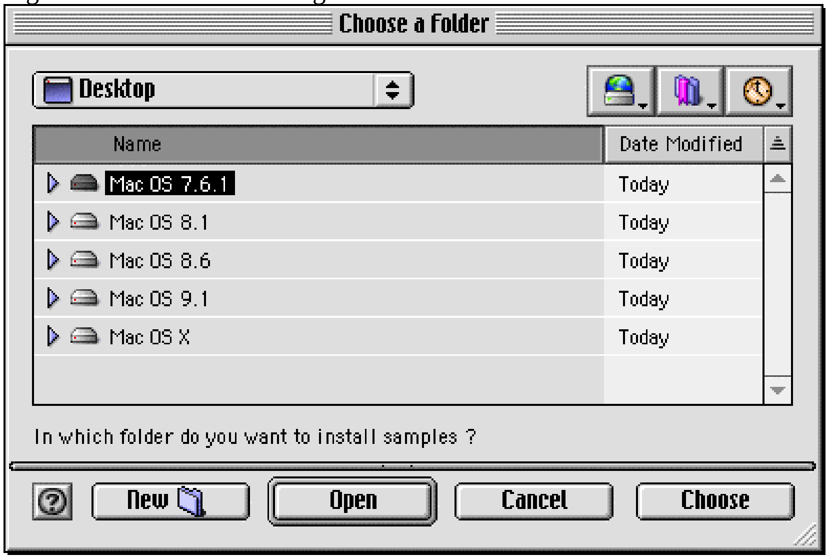
iODBC Driver Manager (PPC versions) – Requires 527 K

This installs the iODBC Driver Manager v3.0.5, and has to be considered actually as a beta release. If you do not have any ODBC Driver Manager actually installed, you have to install it to activate

Installation requires : 6 MB

Press the Install button to start the installation. You will be prompted to select a location for Sample Applications and other files.

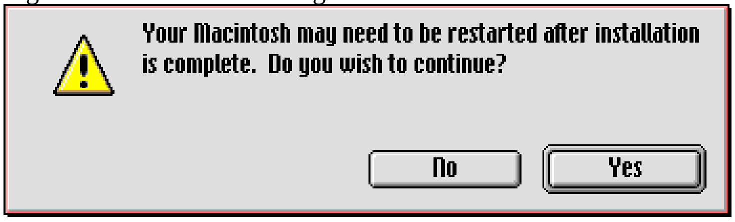
Figure 3.38. MacOSClass07.gif



In which folder do you want to install samples ?

You Macintosh will need to be restarted after installation has completed. You will be given the opportunity to save any open documents.

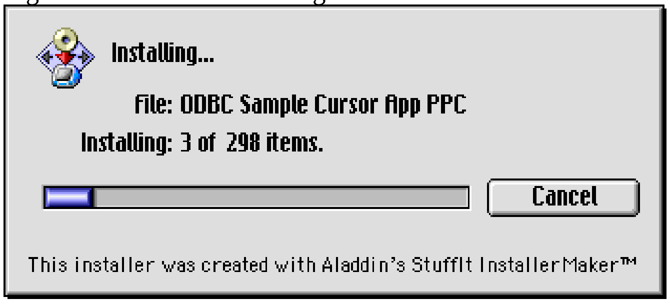
Figure 3.39. MacOSClass08.gif



Your Macintosh may need to be restarted after installation is complete. Do you wish to continue?

Installation will proceed.

Figure 3.40. MacOSClass09.gif



Installing...

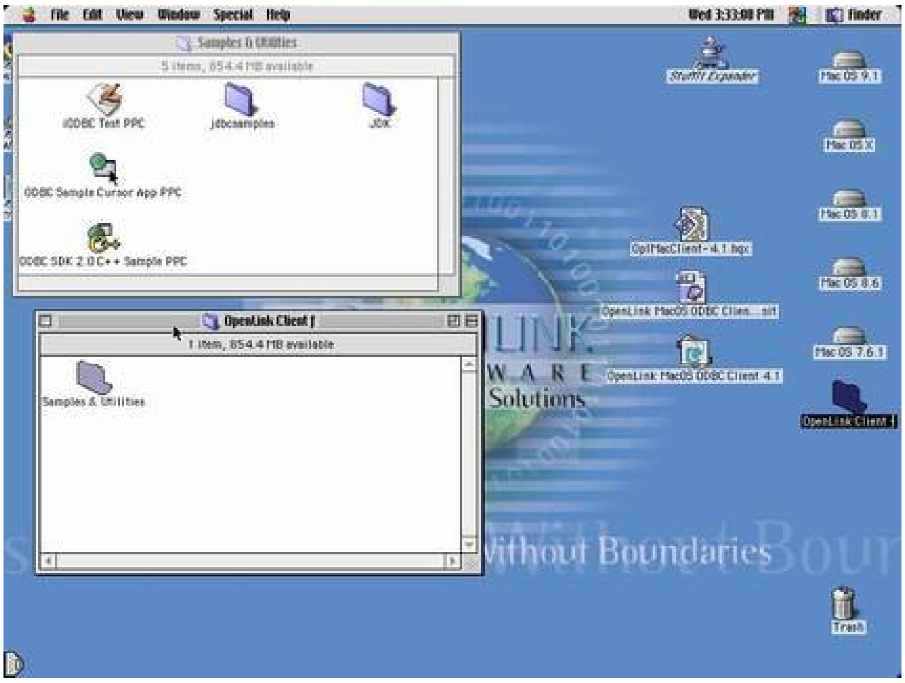
File: ODBC Sample Cursor App PPC

Installing: 3 of 298 items.

This installer was created with Aladdin's StuffIt InstallerMaker™

After installation you may safely delete the files in the temporary location ('OpenLink MacOS ODBC Client 4.1', 'OplMacClient-4.1.sit', mmoczzzz.hqx).

Figure 3.41. MacOSClass10.jpg



5.2.2 Data Source Configuration

Once you have completed the installation process, proceed to create a new ODBC DSN using the OpenLink ODBC Administrator (formerly ODBC Setup PPC) control panel.

The OpenLink ODBC Administrator for Mac Classic currently supports the creation of User and File DSNs.

If in Multiple Users mode, a User DSN is only available to the user who creates the data source. Its parameters are stored in that user's settings file: `Preferences/ODBC Preferences PPC`.

If in Single User mode, a User DSN is only available to the whole system so that any user, including the system itself, will be able to use that data source. Its parameters are stored in the System settings file: `System Folder/Preferences/ODBC Preferences PPC`.

A File DSN is a special 'mobile' data source that stores the data source information associated with the Driver in a file, which may then be copied and shared among different users and ODBC application host machines.

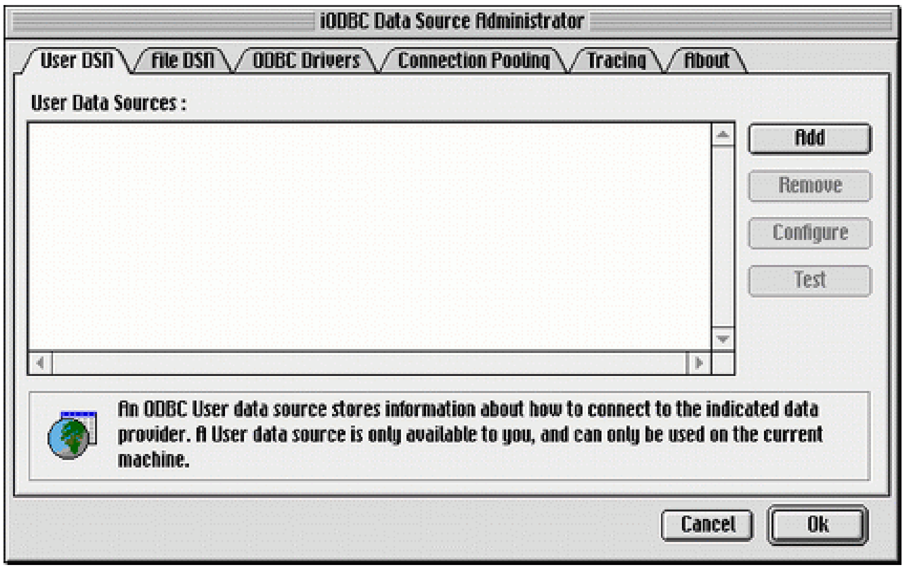
File DSNs are usually created when the same DSN needs to be used by many different users, from many different workstations. File DSNs may be passed from Macintosh to Macintosh, or User to User in Multiple Users mode, and will be usable by anyone with the same ODBC Drivers installed. By default, File DSNs are found in: `System Folder/Preferences/ODBC File Data Sources.`

Most often, a User DSN will be appropriate for use on a Macintosh running the Classic Mac OS.

5.2.2.1 User Data Source Names

To create a new User DSN, open the ODBC Setup PPC control panel. Hit the Add button.

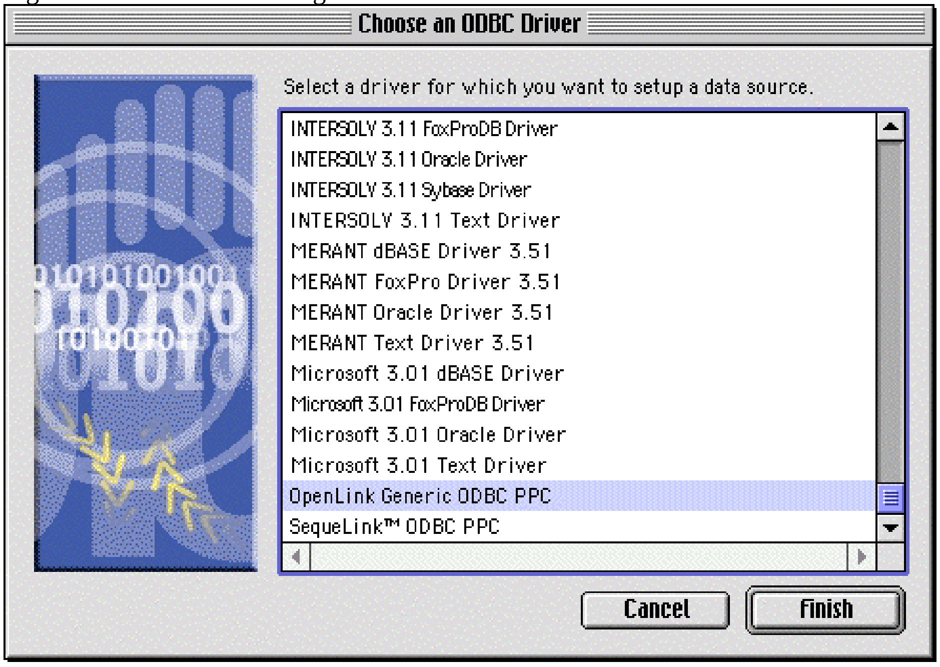
Figure 3.42. MacOSClass11.gif



An ODBC User data source stores information about how to connect to the indicated data provider. A User data source is only available to you, and can only be used on the current machine.

Pick the ODBC Driver to be used to create your ODBC DSN, in the case of the OpenLink Universal Data Access Driver Suite 5.0, this would be the Driver identified below as "OpenLink Generic ODBC PPC".

Figure 3.43. MacOSClass12.gif

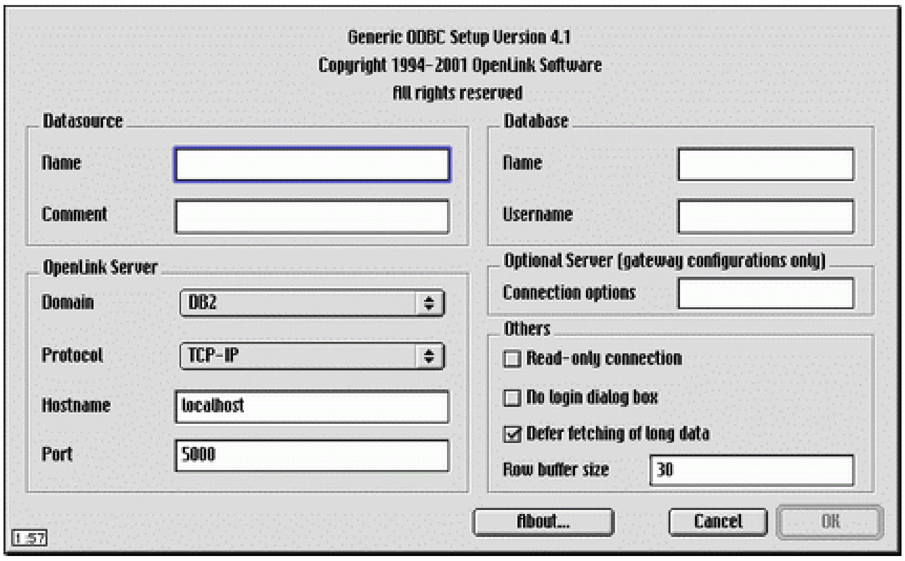


Select a driver for which you want to setup a data source.

INTERSOLV 3.11 FoxProDB Driver
INTERSOLV 3.11 Oracle Driver
INTERSOLV 3.11 Sybase Driver
INTERSOLV 3.11 Text Driver
MERANT dBASE Driver 3.51
MERANT FoxPro Driver 3.51
MERANT Oracle Driver 3.51
MERANT Text Driver 3.51
Microsoft 3.01 dBASE Driver
Microsoft 3.01 FoxProDB Driver
Microsoft 3.01 Oracle Driver
Microsoft 3.01 Text Driver
OpenLink Generic ODBC PPC
SequeLink™ ODBC PPC

Once you have selected the "OpenLink Generic 32 Bit Driver v5.0", you will be presented with the OpenLink ODBC Data Source Configuration dialog (depicted below).

Figure 3.44. MacOSClass13.gif


Generic ODBC Setup Version 4.1
Copyright 1994-2001 OpenLink Software
All rights reserved

Datasource
Name
Comment

Database
Name
Username

OpenLink Server
Domain: DB2
Protocol: TCP-IP
Hostname: localhost
Port: 5000

Optional Server (gateway configurations only)
Connection options

Others
☐ Read-only connection
☐ No login dialog box
☑ Defer fetching of long data
Row buffer size: 30

The Datasource settings are only important to your Client machine; the OpenLink Server, Database, and Optional Server settings should be provided by your Database Administrator, or whoever is handling the Server-side components of the OpenLink Universal Data Access Suite.

5.2.3 Note:

The connections will not succeed until Server-side components are installed and properly configured.

Explanations of each field follows:

Datasource Name. The name of the ODBC DSN. This is how you will interact with the OpenLink ODBC Driver from within ODBC compliant applications once your ODBC DSN has been created.

Comment. Additional information that further describes the ODBC DSN that you are creating. This is generally visible whenever a list of DSNs is generated. This option corresponds to the Description parameter in the preference files.

Domain. This is how you pick the Database Engine Type (also called a DBMS, or DataBase Management System) that your ODBC DSN is to be associated with, e.g. Informix 7, Oracle 9, Progress 8.3B, etc. This option corresponds to the ServerType parameter in the preference files.

Protocol. Select the network transport protocol used to connect your ODBC DSN with a remote database engine. This will be TCP/IP in most installations. This option corresponds to the Protocol parameter in the preference files.

Hostname. Enter the hostname or IP address that identifies a Server Machine running OpenLink Server Components, that speak the OpenLink Data Access Protocol. This option corresponds to the Host parameter in the preference files.

Port. Enter the TCP port number of the Broker to contact. This corresponds to the Listen parameter on the target Broker. This option corresponds to the Port parameter in the preference files.

Database Name. This is how an actual database name within the Provider Type Domain is identified, for instance "stores7" indicates an "Informix 7" database called "stores7". This option corresponds to the Database parameter in the preference files.

Username. The default database UserID to be used when logging on to a remote database engine (identified by the Provider Type above). Generally left blank on shared machines. This option corresponds to the User parameter in the preference files.

Server Name. This is where you place any database specific database connection options. This field in a majority of cases should be left blank by default. Your Database Administrator (DBA) should give you any special settings. This option corresponds to the Options parameter in the preference files.

Read-only connection. Forces the connection is to be "Read-only". Make sure the checkbox is unchecked to request a "Read/Write" connection. This option corresponds to the ReadOnly parameter in the preference files.

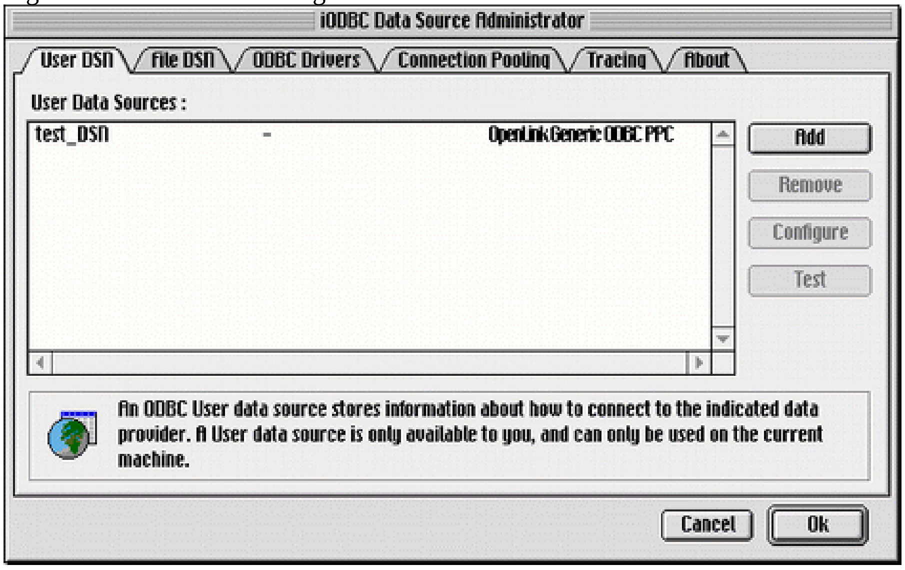
No Login Dialog Box. Suppress the ODBC "Username" and "Password" login dialog box when interacting with your ODBC DSN from within an ODBC compliant application. This should be unchecked if the database requires this information. This option corresponds to the NoLoginBox parameter in the preference files.

Defer fetching of long data. Check this box to defer the fetching of long data. See the Release Notes section for more details. This option corresponds to the DeferLongFetch parameter in the preference files.

Row Buffer Size. This attribute specifies the number of records to be transported over the network in a single network hop. Values can range from 1 to 999. Smaller values are generally appropriate when each row contains large records; larger values are generally appropriate for smaller record sizes. Adjusting this value up and down can improve or degrade performance substantially, both for your own connection and for other users of the same Database, so check with your DBA before changing it from the default (30). This option corresponds to the FetchBufferSize parameter in the preference files.

Click OK, and you'll see your new DSN in the list.

Figure 3.45. MacOSClass14.gif



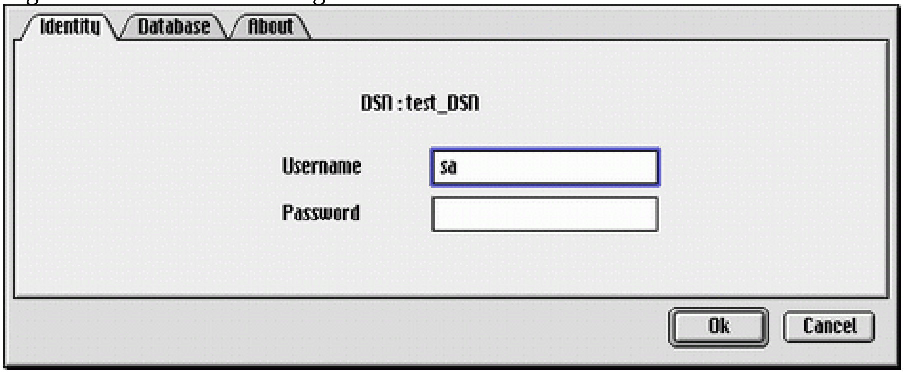
User Data Sources :

test_DSN	-	OpenLink Generic ODBC PPC

An ODBC User data source stores information about how to connect to the indicated data provider. A User data source is only available to you, and can only be used on the current machine.

Now, to verify that your settings are correct and all is functional, select the new DSN, and click Test. You'll be presented with the following dialog.

Figure 3.46. MacOSClass15.gif



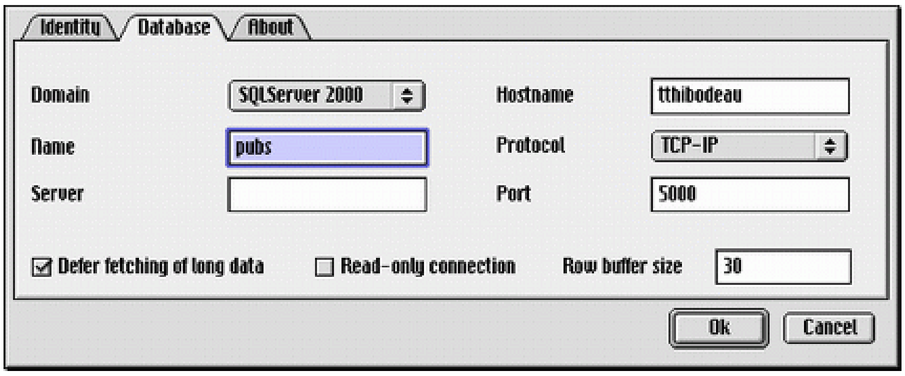
DSN : test_DSN

Username sa

Password

Input the correct password for the username shown. You can connect as another user, if you wish, by simply typing a different username in the field. To verify that your settings have all "stuck", you can click the Database tab:

Figure 3.47. MacOSClass16.gif



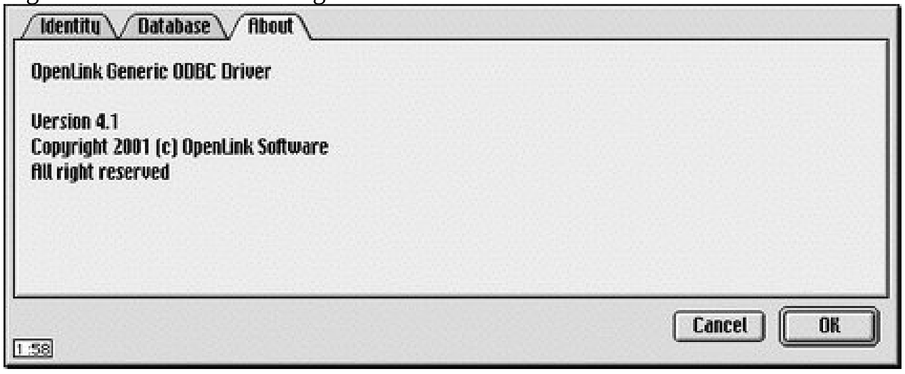
Identity | Database | About

Domain	SQLServer 2000	Hostname	tthibodeau
Name	pubs	Protocol	TCP-IP
Server		Port	5000

☑ Defer fetching of long data ☐ Read-only connection Row buffer size 30

The "About" tab shows you basic information about the Driver on which the DSN is based.

Figure 3.48. MacOSClass17.gif

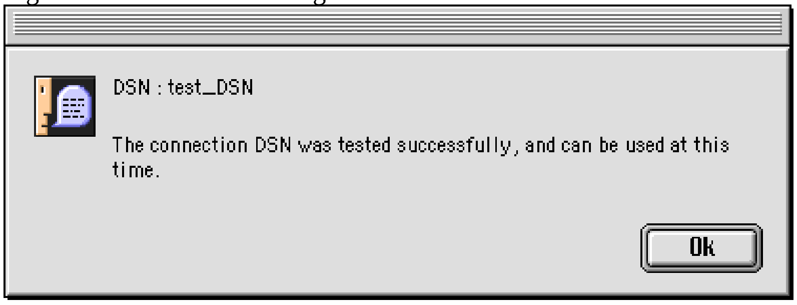


OpenLink Generic ODBC Driver

Version 4.1
Copyright 2001 (c) OpenLink Software
All right reserved

When you click OK, your Mac will think for a moment, while it makes a connection to the remote Data Source. If all is well, you will be presented with this Success message:

Figure 3.49. MacOSClass18.gif



DSN : test_DSN

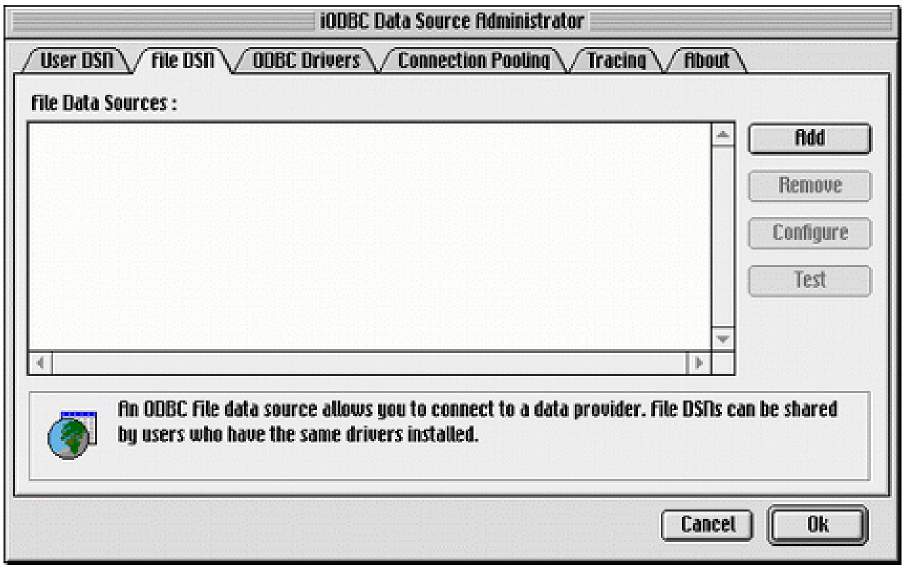
The connection DSN was tested successfully, and can be used at this time.

5.2.3.1 Configuring File Data Sources

To create a new File DSN, open the ODBC Setup PPC control panel. Click on the File DSN tab, and follow the steps below:

Click the Add button.

Figure 3.50. MacOSClass19.gif



iODBC Data Source Administrator

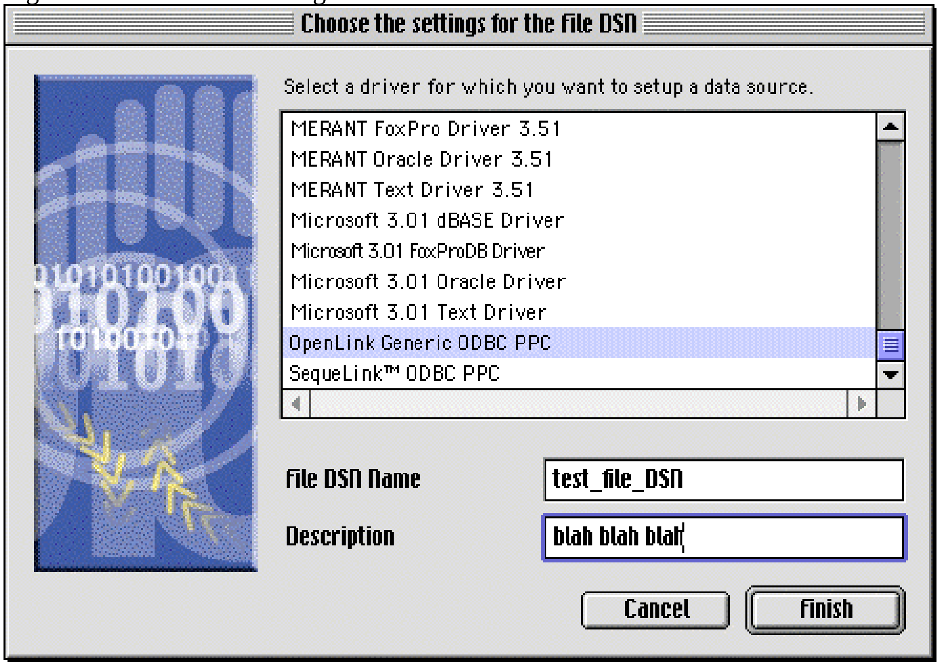
User DSN | File DSN | ODBC Drivers | Connection Pooling | Tracing | About

File Data Sources :

An ODBC File data source allows you to connect to a data provider. File DSNs can be shared by users who have the same drivers installed.

Pick the ODBC Driver to be used to create your ODBC DSN. As with the User DSN, this would be the Driver identified below as "OpenLink Generic ODBC PPC". Input a name for the File DSN, and any description. Click "Finish" or "OK".

Figure 3.51. MacOSClass20.gif



Choose the settings for the File DSN

Select a driver for which you want to setup a data source.

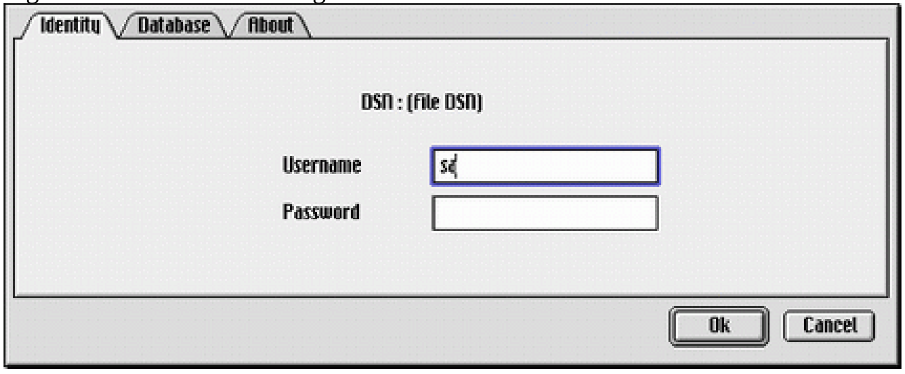
MERANT FoxPro Driver 3.51
MERANT Oracle Driver 3.51
MERANT Text Driver 3.51
Microsoft 3.01 dBASE Driver
Microsoft 3.01 FoxProDB Driver
Microsoft 3.01 Oracle Driver
Microsoft 3.01 Text Driver
OpenLink Generic ODBC PPC
SequeLink™ ODBC PPC

File DSN Name: test_file_DSN

Description: blah blah blah

File DSNs are configured through the login screens, as if you were connecting through a User DSN. Starting with the Identity tab, input the appropriate Username and Password. (The Password will not be saved in the DSN, as this would present a significant security risk.)

Figure 3.52. MacOSClass21.gif



Identity | Database | About

DSN : (File DSN)

Username: sa

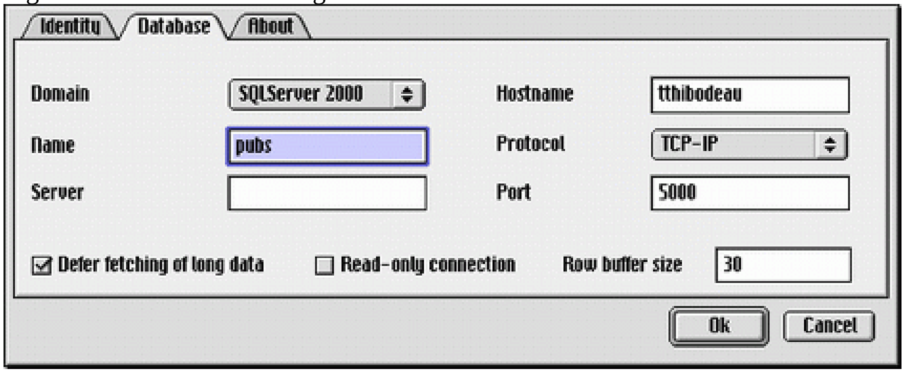
Password:

The OpenLink Server, Database, and Optional Server settings seen in the User DSN configuration screen are combined into the Database tab. Again, these should be provided by your Database Administrator, or whoever is handling the Server-side components of the OpenLink Universal Data Access Suite.

5.2.4 Note:

The connections will not succeed until Server-side components are installed and properly configured.

Figure 3.53. MacOSClass16.gif



Domain	SQLServer 2000	Hostname	tthibodeau
Name	pubs	Protocol	TCP-IP
Server		Port	5000

☑ Defer fetching of long data ☐ Read-only connection Row buffer size 30

Explanations of each field in the Database tab follows:

Domain. This is how you pick the Database Engine Type (also called a DBMS, or DataBase Management System) that your ODBC DSN is to be associated with, e.g. Informix 7, Oracle 9, Progress 8.3B, etc. This option corresponds to the ServerType parameter in the preference files.

Name. This is how an actual database name within the Provider Type Domain is identified, for instance "stores7" indicates an "Informix 7" database called "stores7". This option corresponds to the Database parameter in the preference files.

Server. This is where you place any database specific database connection options. This field in a majority of cases should be left blank by default. Your Database Administrator (DBA) should give you any special settings. This option corresponds to the Options parameter in the preference files.

Hostname. Enter the hostname or IP address that identifies a Server Machine running OpenLink Server Components, that speak the OpenLink Data Access Protocol. This option corresponds to the Host parameter in the preference files.

Protocol. Select the network transport protocol used to connect your ODBC DSN with a remote database engine. This will be TCP/IP in most installations. This option corresponds to the Protocol parameter in the preference files.

Port. Enter the TCP port number of the Broker to contact. This corresponds to the Listen parameter on the target Broker. This option corresponds to the Port parameter in the preference files.

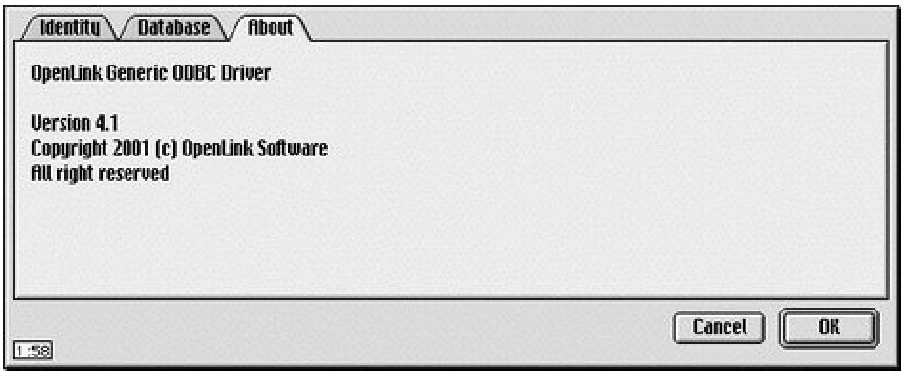
Defer fetching of long data. Check this box to defer the fetching of long data. See the Release Notes section for more details. This option corresponds to the DeferLongFetch parameter in the preference files.

Read-only connection. Forces the connection is to be "Read-only". Make sure the checkbox is unchecked to request a "Read/Write" connection. This option corresponds to the ReadOnly parameter in the preference files.

Row Buffer Size. This attribute specifies the number of records to be transported over the network in a single network hop. Values can range from 1 to 999. Smaller values are generally appropriate when each row contains large records; larger values are generally appropriate for smaller record sizes. Adjusting this value up and down can improve or degrade performance substantially, both for your own connection and for other users of the same Database, so check with your DBA before changing it from the default (30). This option corresponds to the FetchBufferSize parameter in the preference files.

The About tab, here, refers to the Driver you've selected for this DSN.

Figure 3.54. MacOSClass17.gif

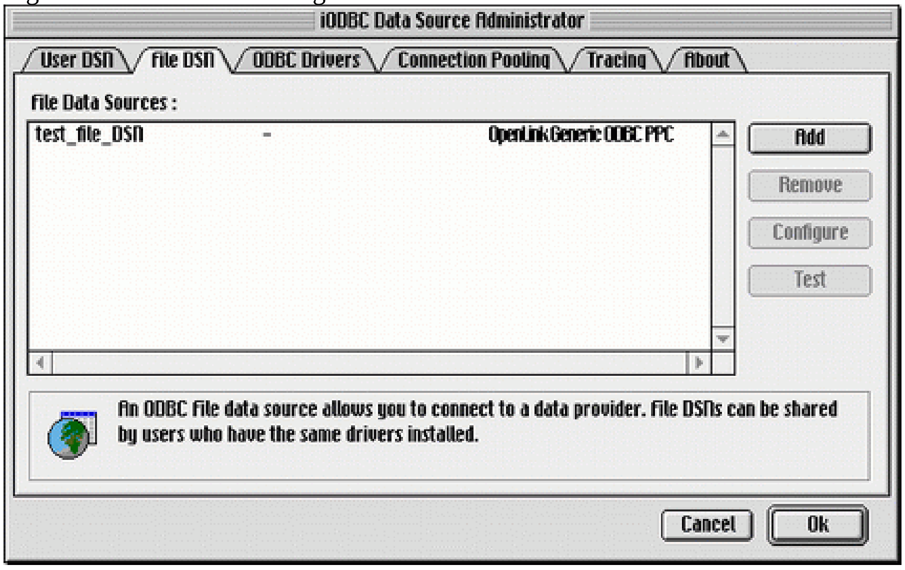


OpenLink Generic ODBC Driver

Version 4.1
Copyright 2001 (c) OpenLink Software
All right reserved

Click OK when all settings are correct, and you will see the new DSN available in the File DSN list. Sharing this DSN with other users and machines may be accomplished by going to the 'ODBC File Data Sources' folder, found in the 'Preferences' folder, within your System Folder. Other Macintosh Classic OpenLink clients may use the file by placing it in the same location on their machine.

Figure 3.55. MacOSClass22.gif



File Data Sources :

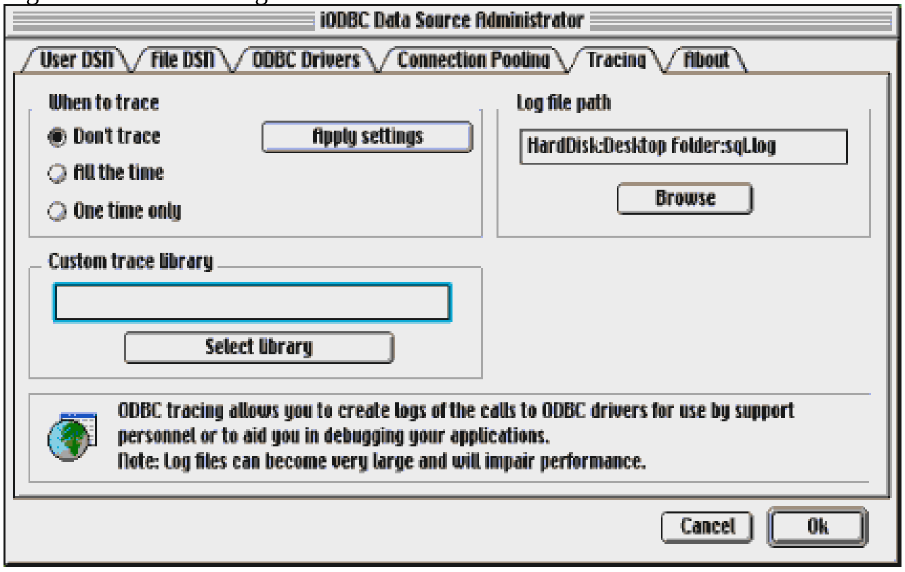
test_file_DSN - OpenLink Generic ODBC PPC

An ODBC File data source allows you to connect to a data provider. File DSNs can be shared by users who have the same drivers installed.

5.2.5 Tracing Data Sources

From the Tracing Tab of the iODBC Data Source Administrator, the level of Tracing can be configured.

Figure 3.56. MacTrace.gif



When to trace
● Don't trace
○ All the time
○ One time only

Log file path
HardDisk:Desktop Folder:sql.log

Custom trace library

ODBC tracing allows you to create logs of the calls to ODBC drivers for use by support personnel or to aid you in debugging your applications.
Note: Log files can become very large and will impair performance.

When to trace. Use this option to turn tracing off or on. For a One-time only trace, the trace is made for the duration of the next connection. Click the Apply button to make your selection active. This option corresponds to the Trace and TraceAutoStop parameters in the preference files.

Custom trace library. Use the Select Library button to browse the machine for a library file that contains the code to intercept the trace output. This field should generally be left blank. When setting this parameter, you may use the Browse button to locate the library, or specify it manually. If specified manually, note that the location must be specified as it would be in a Terminal session. You may use ~/ as a shortcut to your user Home directory. This option corresponds to the TraceDLL parameter in the preference files.

Log file path. Specify the complete location of the file to which you wish to write the trace. You may use the Browse button to locate or create the file, or specify it manually. If specified manually, note that the location must be specified as it would be in a Terminal session. You may use ~/ as a shortcut to your user Home directory. This option corresponds to the TraceFile parameter in the preference files.

5.2.5.1 Trace Parameters in ODBC.INI

When the ODBC tracing parameters are changed, they are saved in the [ODBC] section of the ODBC.preference or odbc.ini file, whichever is active on your system.

The following example is for a one time trace to sql.log file on the active user's desktop.

```
[ODBC]
Trace         = 1
TraceAutoStop = 1
TraceDLL      =
TraceFile     = ~/Desktop/sql.log
```

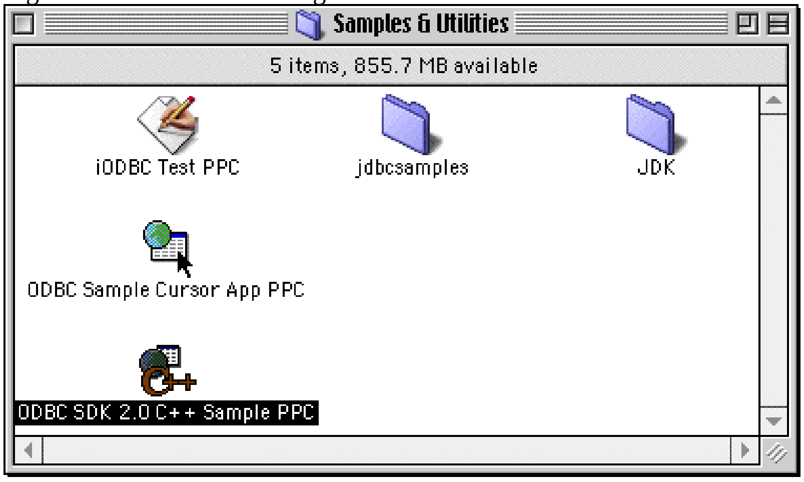
Once the tracing of a connection is complete, the Trace and TraceAutoStop parameters will both be set to 0.

5.2.6 Testing Data Sources

5.2.6.1 ODBC SDK C++ Sample

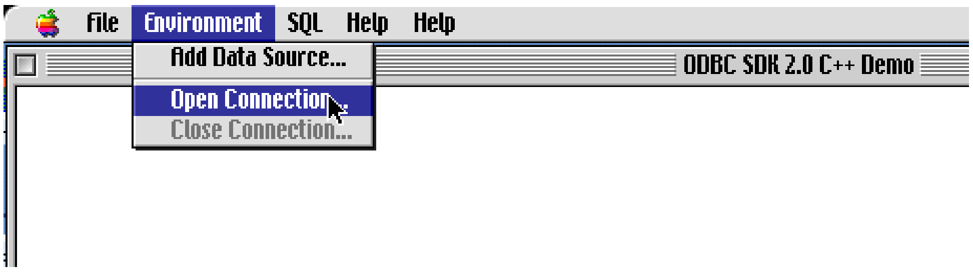
Use the 'ODBC SDK 2.0 C++ Sample PPC' application to test a DSN:

Figure 3.57. MacOSClass23.gif


Samples & Utilities
5 items, 855.7 MB available
iODBC Test PPC
jdbcsamples
JDK
ODBC Sample Cursor App PPC
ODBC SDK 2.0 C++ Sample PPC

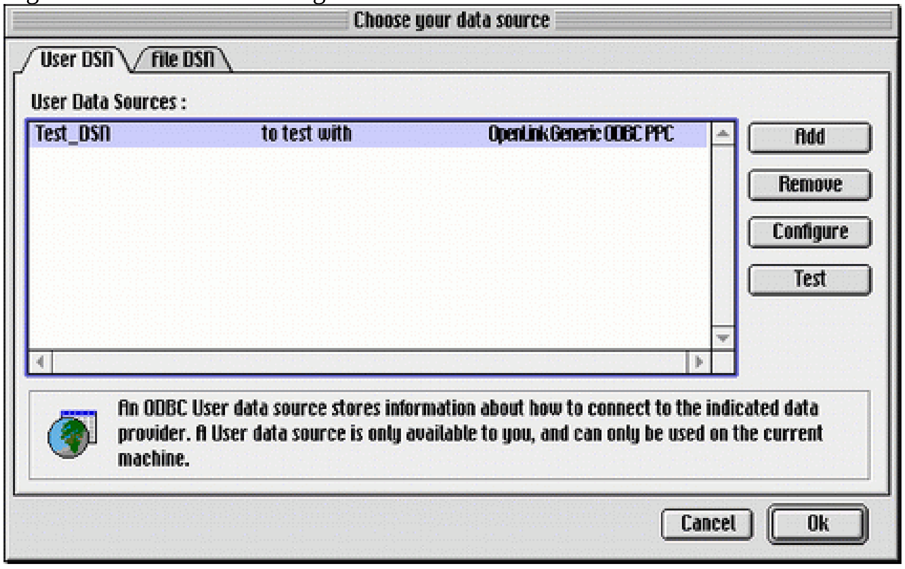
From the Environment menu, select "Open Connection...".

Figure 3.58. MacOSClass24.gif



Choose a User DSN, or press the File DSN tab, and select a File DSN.

Figure 3.59. MacOSClass25.gif

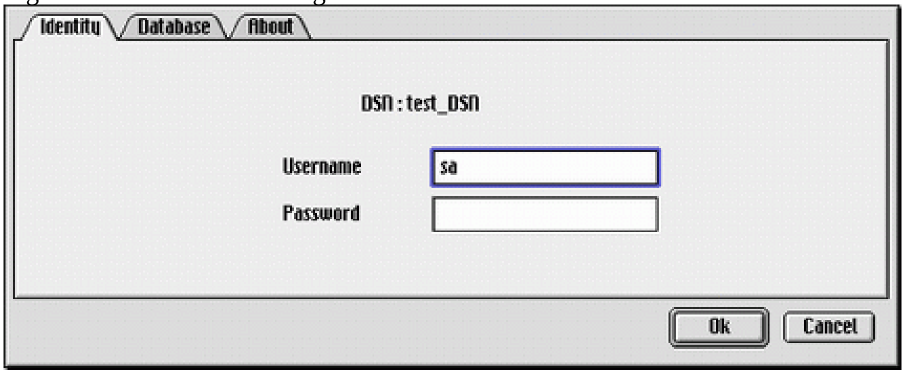


Test_DSN to test with OpenLink Generic ODBC PPC

An ODBC User data source stores information about how to connect to the indicated data provider. A User data source is only available to you, and can only be used on the current machine.

With a DSN highlighted, press OK. Input Username and Password as appropriate, and click OK.

Figure 3.60. MacOSClass15.gif



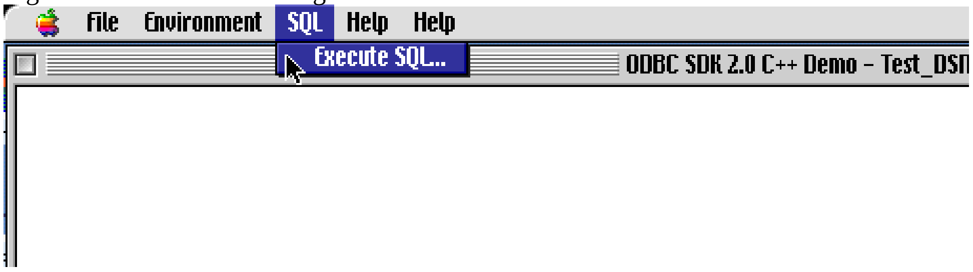
DSN : test_DSN

Username sa

Password

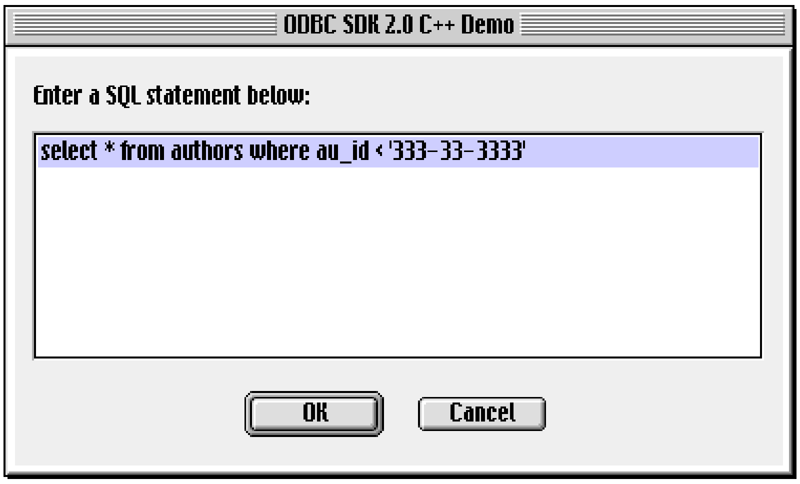
From the SQL menu select Execute SQL...

Figure 3.61. MacOSClass26.gif



Enter an SQL query in the dialog.

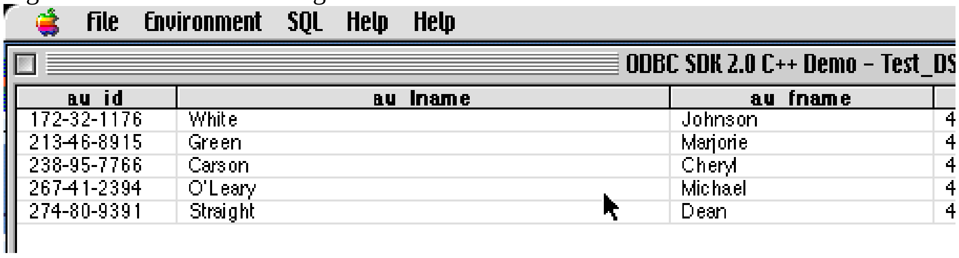
Figure 3.62. MacOSClass27.gif



```
select * from authors where au_id < '333-33-3333'
```

With success, the results will be shown.

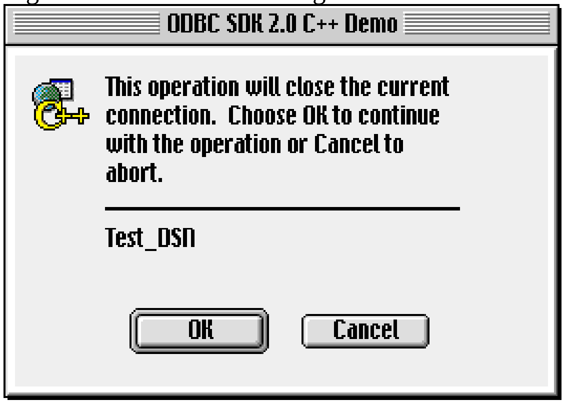
Figure 3.63. MacOSClass28.gif



au_id	au_lname	au_fname	
172-32-1176	White	Johnson	4
213-46-8915	Green	Marjorie	4
238-95-7766	Carson	Cheryl	4
267-41-2394	O'Leary	Michael	4
274-80-9391	Straight	Dean	4

When you QUIT the application, you will see a warning message about your DSN connection. Please note you will continue to consume one licensed connection until the application is terminated.

Figure 3.64. MacOSClass29.gif



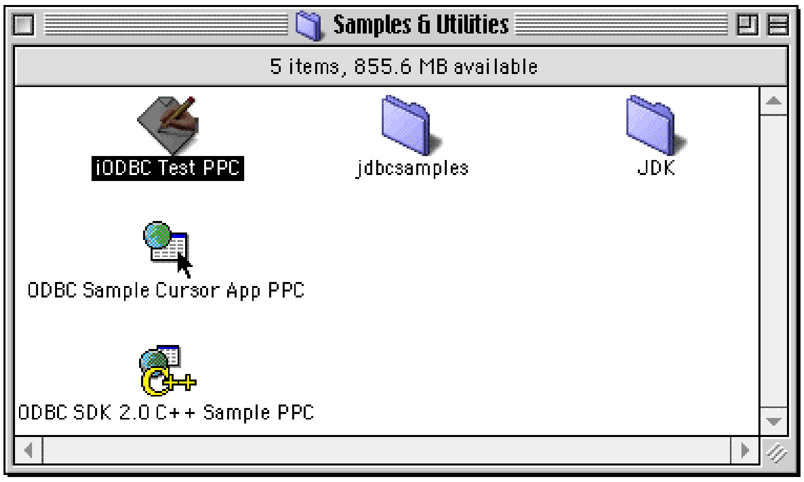
This operation will close the current connection. Choose OK to continue with the operation or Cancel to abort.

Test_DSN

5.2.6.2 iODBC Test PPC

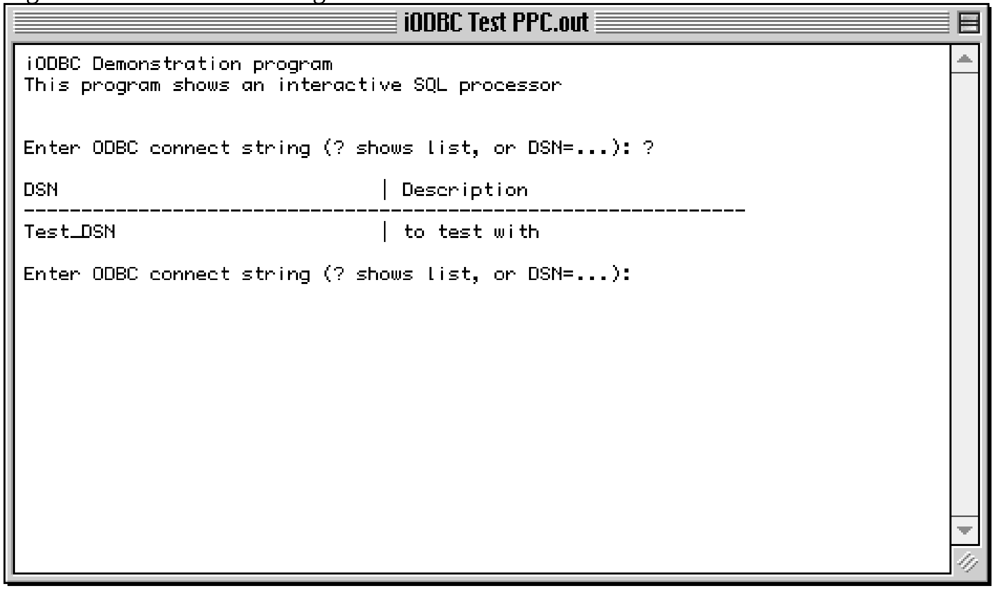
Use the 'iODBC Test PPC' application to test a DSN:

Figure 3.65. MacOSClass30.gif


Samples & Utilities
5 items, 855.6 MB available
iODBC Test PPC
jdbcsamples
JDK
ODBC Sample Cursor App PPC
ODBC SDK 2.0 C++ Sample PPC

Type a question mark ('?') and hit <return> to get a list of available User DSNs.

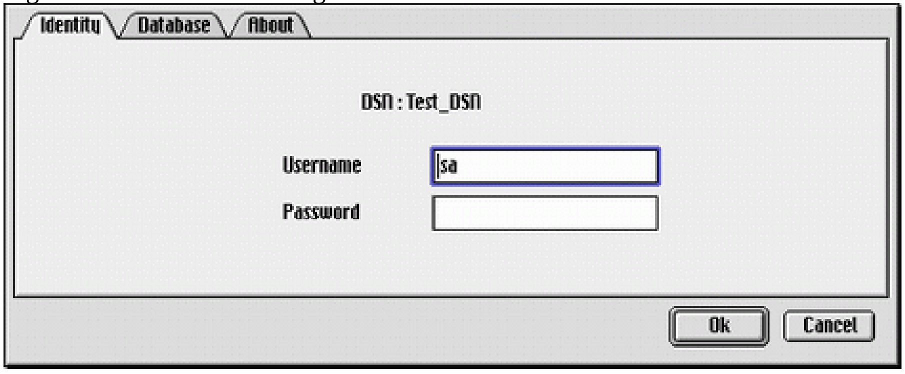
Figure 3.66. MacOSClass31.gif



```
iODBC Demonstration program
This program shows an interactive SQL processor

Enter ODBC connect string (? shows list, or DSN=...): ?

DSN                             | Description
-----------------------------------------------------------------
Test_DSN                        | to test with

Enter ODBC connect string (? shows list, or DSN=...):
```

Type a connect string, using one of the listed User DSNs, following the format DSN=<DSN>;UID=<username>;PWD=<password>. You may leave the UID and/or PWD parameters out of your string; if you do, you will be presented with the usual log-in dialog:

Figure 3.67. MacOSClass32.gif



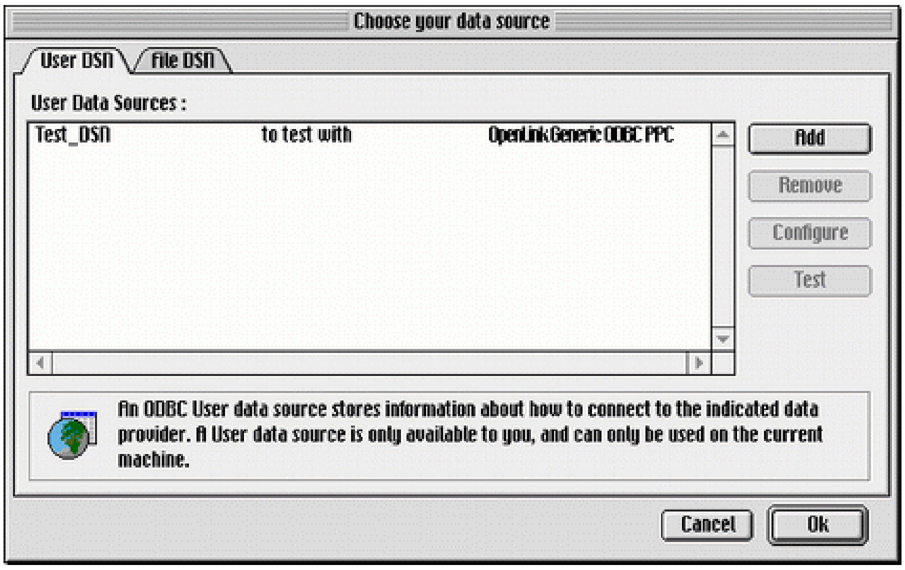
DSN : Test_DSN

Username: sa

Password:

If you do not wish to type so much, need to add a DSN, or wish to connect to a File DSN, type anything else, and hit the <return> key. You will be presented with the DSN Chooser dialog, in which you may do select an existing User or File DSN, or create a new DSN of either type. Instructions on creation may be found earlier in this chapter.

Figure 3.68. MacOSClass33.gif



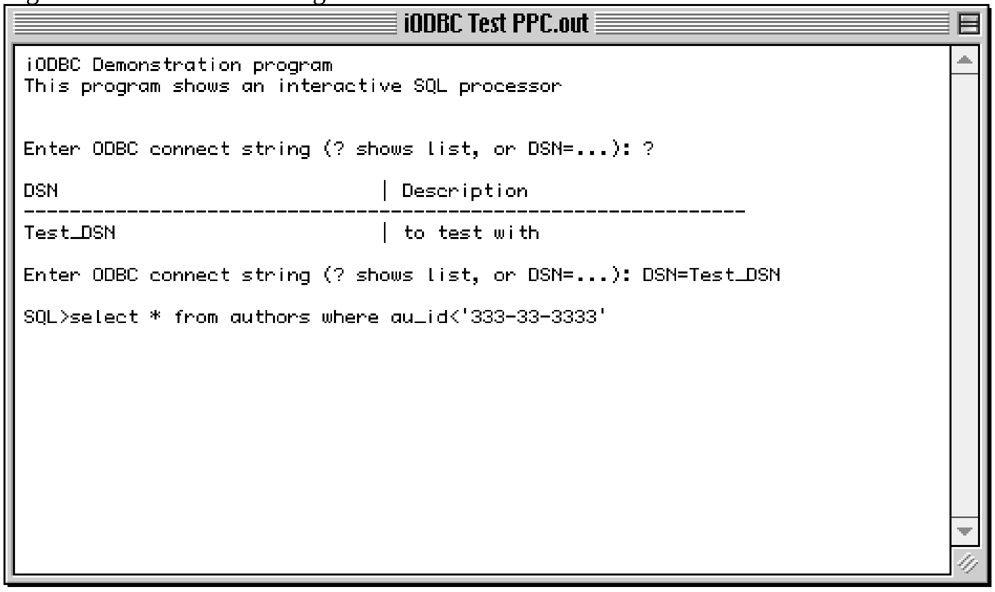
User Data Sources :

```
Test_DSN          to test with          OpenLink Generic ODBC PPC
```

An ODBC User data source stores information about how to connect to the indicated data provider. A User data source is only available to you, and can only be used on the current machine.

When the 'SQL>' prompt comes up, you can enter any SQL statement, including UPDATE, INSERT, DELETE, or SELECT, followed by <return>.

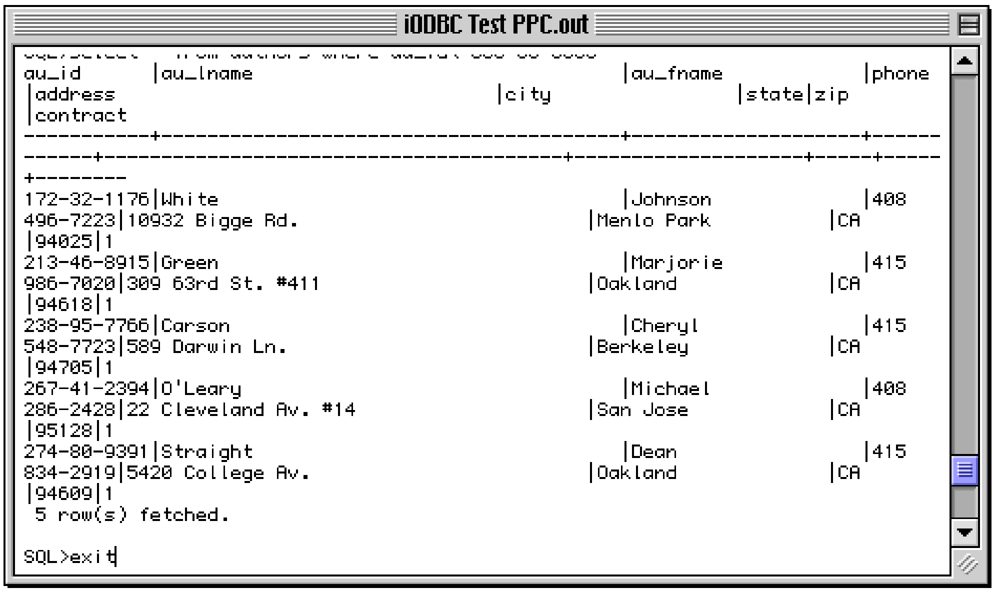
Figure 3.69. MacOSClass34.gif



```
iODBC Demonstration program
This program shows an interactive SQL processor


Enter ODBC connect string (? shows list, or DSN=...): ?

DSN                               | Description
-------------------------------------------------------------------
Test_DSN                          | to test with

Enter ODBC connect string (? shows list, or DSN=...): DSN=Test_DSN

SQL>select * from authors where au_id<'333-33-3333'
```

Results are returned in plain text, to your query window.

Figure 3.70. MacOSClass35.gif

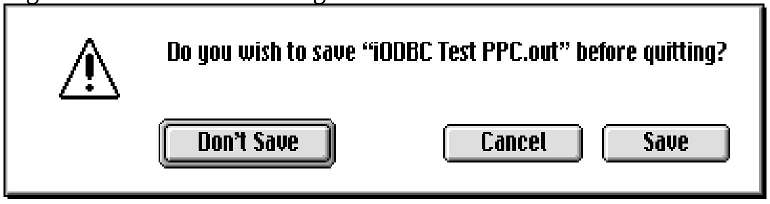


```
au_id      |au_lname                                |au_fname            |phone
|address                               |city                |state|zip
|contract
-----------+----------------------------------------+--------------------+------
------+---------------------------------------+--------------------+-----+-----
+--------
172-32-1176|White                                   |Johnson             |408
496-7223|10932 Bigge Rd.                         |Menlo Park          |CA
|94025|1
213-46-8915|Green                                   |Marjorie            |415
986-7020|309 63rd St. #411                       |Oakland             |CA
|94618|1
238-95-7766|Carson                                  |Cheryl              |415
548-7723|589 Darwin Ln.                          |Berkeley            |CA
|94705|1
267-41-2394|O'Leary                                 |Michael             |408
286-2428|22 Cleveland Av. #14                    |San Jose            |CA
|95128|1
274-80-9391|Straight                                |Dean                |415
834-2919|5420 College Av.                        |Oakland             |CA
|94609|1
 5 row(s) fetched.

SQL>exit
```

Type 'exit<return>' to close your DSN connection. The prompt 'Again (y/n) ?' is asking whether you wish to connect to another DSN. Respond y(es) or n(o) and hit <return>. Yes brings you back to the 'Enter ODBC string' prompt; no returns 'Have a nice day.'

When you QUIT the application, you will be prompted to save the output text. Default is "Don't save".

Figure 3.71. MacOSClass38.gif



5.3 Mac OS X

OpenLink Software's components are fully supported on Mac OS X versions 10.1.x through 10.4.x ("Puma" to "Tiger"). In most ways, these look the same to users. However, there are some significant differences to be aware of between these Operating Systems.

As shipped from Apple, Mac OS X did not include any ODBC support until Jaguar. The ODBC Driver Manager, Data Source Administrator, etc. - all had to be delivered with the Drivers. When Apple introduced Darwin, an Open-Source project meant to form the kernel of Mac OS X, OpenLink were determined to port its open-source iODBC Driver Manager to the new platform. With the introduction of the Preview Release of Mac OS X, the traditional set of dynamic libraries was broadened to include a system of Frameworks to encourage the development of fully ODBC compliant, native Mac OS X drivers and client applications.

That set of Frameworks, along with the OpenLink ODBC Administrator (then known as the iODBC Administrator), has been included with all OpenLink installations for Mac OS X since 10.0. The OpenLink ODBC Administrator supports all core features of ODBC, and presents driver-specific DSN configuration panels, as defined by the Driver developer, through the use of Setup Libraries.

Jaguar's release marked Apple's recognition that Data Access was an important part of an Enterprise Operating System. Already part of the standard distribution of Darwin, the basic iODBC dynamic libraries are now a part of the standard installation of Mac OS X 10.2. Apple also included their own version of an ODBC Administrator, as a proof-of-concept. Apple's Administrator permits Driver Registration, Tracing, and all other core features of ODBC; however, among other hard edges, all DSN configuration must be done by manually entering Keyword-Value pairs. Further, the user must know what Keywords to use, along with their acceptable Values.

The OpenLink Generic Multi-Tier Driver for ODBC for Mac OS X is distributed in a single disk image (.dmg) file, which contains a Macintosh Installer mpkg.

The OpenLink ODBC Client Components for Mac OS X comprise the following:

Table 3.3.

Component	Purpose
iODBC Driver Manager	A shared library that links ODBC compliant applications to ODBC Drivers
Generic ODBC Driver	A shared library that provides database connectivity and data access services to ODBC compliant applications
Sample ODBC Applications	Simple programs that can be used to verify your ODBC installation and working environment

5.3.1 Installation

Double-click the mpkg to start the installation process.

Figure 3.72. OpenLinkUDA-4.5.dmg

You must have an Administration username and password to install the OpenLink Generic Multi-Tier Driver for ODBC. The Jaguar Installer will immediately prompt you to enter your Mac OS X Username and Password.

Figure 3.73. Authentication

The installer will display a "Welcome" message. Click "Continue".

Figure 3.74. Welcome

The next screen will display the Read-Me file, including any last-minute updates to these documents. Please read carefully, and click "Continue" when finished.

Figure 3.75. Read Me

The next screen will display the License Agreement for the OpenLink Single-Tier Driver for ODBC. Please read, and click "Continue".

Figure 3.76. Software License Agreement

You will be prompted to "Agree" to continue the installation, or "Diagree" to abort.

Figure 3.77. Agree or Disagree to Licensing

You will be asked to select a Destination Volume. Generally, this should be your Mac OS X boot volume. Click on the desired disk icon, and then click "Continue".

Figure 3.78. Choose Destination

You may now choose the Easy Install, or if you are an experienced user, you may Customize which components are installed. OpenLink generally recommends the Easy Install. If you have installed OpenLink or iODBC components in the past, click "Upgrade" to continue; otherwise, click "Install".

Figure 3.79. Easy Install

Figure 3.80. Custom Install

You will be shown a graphical progress bar as Installation progresses, followed by System Optimization.

Figure 3.81. Installation Progress

When the process is complete, you will be told that the software was successfully installed. Click "Close" and your new database driver for ODBC is ready for use!

Figure 3.82. Installation Proceeds

5.3.2 Data Source Configuration

The OpenLink ODBC Administrator for Mac OS X currently supports creation of User and System Data Source Names (DSNs). Support for File DSNs will be delivered in a future release.

A User DSN is only available to the user who creates the data source. Its parameters are stored in that user's settings file `~/Library/Preferences/ODBC.preference`.

A System DSN is available to the whole system so that any user, including the system itself, will be able to use that data source. Its parameters are stored in the System settings file `/Library/Preferences/ODBC.preference`.

A File DSN is a special 'mobile' data source that stores the data source information associated with the Driver in a file, which may then be copied and shared among different users and ODBC application host machines.

5.3.2.1 Creating User or System Data Sources

The steps for creating a DSN are as follows:

1. Launch the OpenLink ODBC Administrator, found in the `/Applications/Utilities/` directory.

   Figure 3.83. OpenLink ODBC Administrator icon

2. Click the tab for the kind of DSN you wish to create - User or System. Press the Add button to begin creating a new Data source.

   Figure 3.84. OpenLink ODBC Administrator, User DSN tab

3. Select the Driver to be used to create your ODBC DSN. In this example make sure "OpenLink Generic ODBC Driver" is highlighted. Then click the Finish button.

   Figure 3.85. Choose an ODBC Driver

   ## 5.3.3 Note:

   The connections will not succeed until Server-side components are installed and properly configured.
4. The ODBC Driver manager now launches the selected driver code so that the DSN may be configured. In this example it will present the OpenLink ODBC Data Source Configuration dialog (depicted below).

   Configure the fields of the ODBC "System" or "User" DSN Configuration dialog in the manner explained below:

   Name. (Datasource) The name of the ODBC DSN, this is how you will interact with the OpenLink ODBC Driver from within ODBC compliant applications once your ODBC DSN has been created.

   Description. Additional information that further describes the ODBC DSN that you are creating.

   Server. Selected Zero Configuration service name or `<hostname or IP address>:<port>` of running Broker.

   The easiest way to complete this field is to select a machine from the drop-down list-box. This list of servers is automatically generated by the Zero Configuration service, so you only need to chose which machine you wish to reach.

   Alternatively, enter the hostname or IP address that identifies a Server Machine running OpenLink Server Components, that speak the OpenLink Data Access Protocol. Follow this with a colon and the TCP/IP Port number of the Broker to contact. This corresponds to the Listen parameter on the target Broker.

   This example shall now assume the selection of a server from a picklist.

   Figure 3.86. OpenLink Generic ODBC Setup Dialog

5. Click on the Next button or the Connection tab. Now select the database and configuration details:

   #### 5.3.3.1 Domain

   This is how you pick the Database Engine Type that your ODBC DSN is to be associated with e.g. Informix 7, Oracle 7, Progress 7 etc.

   The default offered from the Zero Configuration is typically accepted. An alternative compatable Domain may be chosen for a specific customisation.

   Database. This is how an actual database name within the Provider Type Domain is identified, for instance "stores7" indicates an "Informix 7" database called "stores7". This option corresponds to the Database parameter in the preference files.

   Connection Options. This is where you place any database specific database connection options. This field in a majority of cases should be left blank by default.

   ##### 5.3.3.1.1 Connect now..

   When this tick box is checked, a test connection is made to verify the Data Source connection.

   If there is no check then the Login ID and Password fields are ignored, and no test is performed.

   Login ID. The default database UserID to use when logging on to a remote database engine (identified by the Domain above).

   Password. The Password for the login of the above UserID.

   Figure 3.87. OpenLink Generic ODBC Setup Dialog

6. Click on the Next button or the Options tab.

   Now define additonal connection parameters:

   Read-only connection. Specify whether the connection is to be "Read-only". Make sure the checkbox is unchecked to request a "Read/Write" connection.

   Defer fetching of long data. Check this box to defer the fetching of long data.

   Disable interactive login. Suppress the ODBC "Username" and "Password" login dialog box when interacting with your ODBC DSN from within an ODBC compliant application.

   Row Buffer Size. This attribute specifies the number of records to be transported over the network in a single network hop. Values can range from 1 to 99.

   Figure 3.88. OpenLink Generic ODBC Setup Dialog

7. The list of parameters for the DSN are now shown. The Test Data Source button will trigger a test of the DSN using the existing login parameters.

   Click on the Next button or the Test tab.

   Figure 3.89. OpenLink Generic ODBC Setup Dialog

8. Click Finish to save your DSN, which will then be visible in the OpenLink ODBC Administrator's listing.

   Figure 3.90. OpenLink ODBC Administrator, User DSN tab

5.3.4 Testing Data Sources

Any DSN can be tested for basic connectivity, to ensure that the basic parameters have been set correctly, through the OpenLink ODBC Administrator.

1. Click anywhere on the graphic to dismiss the About box. Click OK to save your DSN, which will then be seen in the OpenLink ODBC Administrator's listing.

   Figure 3.91. OpenLink ODBC Administrator, User DSN tab

2. Now, to verify that your settings are correct and all is functional, select the new DSN, and click Test. You'll be presented with the following dialog:

   Figure 3.92. OpenLink Login Dialog, Identity tab

3. Input the correct password for the username shown. You can connect as another user, if you wish, by simply typing a different username in the field. To verify that your settings have all "stuck", you can click the Database tab:

   Figure 3.93. OpenLink Login Dialog, Database tab

4. The "About" tab shows you basic information about the Driver on which the DSN is based.

   Figure 3.94. OpenLink Login Dialog, About tab

5. When you click OK, your Mac will think for a moment, while it makes a connection to the remote Data Source. If there is a problem with the connection, you will be presented with an error message, such as this, resulting from a bad password:

   Figure 3.95. OpenLink ODBC Administrator, DSN Test, Initial Error Message

6. If you receive two error messages, the first will usually provide the clues to resolution; the second is typically a generic error, such as this rejected connection notice:

   Figure 3.96. OpenLink ODBC Administrator, DSN Test, Secondary Error Message

7. If the DSN connects as expected, you will be presented with this Success message:

   Figure 3.97. OpenLink ODBC Administrator, DSN Test, Success Message

5.3.4.1 iODBC Test.command (or /usr/bin/odbctest)

For more thorough testing, you can use iODBC Test, a sample application installed along with the OpenLink Generic Driver for ODBC. iODBC Test is a simple command-line, ODBC compliant, Interactive SQL client.

Either double-click the `/Applications/iODBC/iODBC Test.command` script or manually launch the Terminal application (found in `/Applications/Utilities/`).

Figure 3.98.

If you used the script, you can skip ahead to the next paragraph; otherwise, ensure that the environment variables ODBCINI and ODBCINSTINI are set correctly and enter the command /usr/bin/odbctest.

At the prompt, type a question-mark (?) followed by the ENTER key, to show a list of available DSNs. Enter a connect string at the next prompt, using the syntax:

```
DSN=<DSN Name>[;UID=<username>[;PWD=<password>]]
```

Text enclosed in angle brackets < > should be replaced by your environment's specifics; text enclosed in square brackets [ ] is optional. The only mandatory parameter is the `"DSN="`. All other connect parameters may be set within the DSN itself. User authentication in the connect string will override similar information stored in the DSN. The following connect strings are all valid, assuming a DSN named test_dsn exists:

```
DSN=test_dsn
DSN=test_dsn;UID=;PWD=
DSN=test_dsn;UID=fred
DSN=test_dsn;UID=fred;PWD=
DSN=test_dsn;UID=fred;PWD=derf
```

A successful connection will cause display of the Driver's full version string, as it reports to any client application, followed by a SQL> prompt.

```
Enter ODBC connect string (? shows list, or DSN=...):
DSN=test_dsn;UID=fred;PWD=derf
Driver: 04.20.0402 OpenLink SQL Server Lite Driver
SQL>
```

Once the connection is made, any valid SQL query conforming to ODBC syntax (where applicable) may be executed. Use the command "quit" to close the connection.

If the query executes successfully you will see a table of the data returned by the query; if unsuccessful, you will see the entire error message returned from the back-end DBMS.

Use the command "quit" to close the connection. The prompt Again (y/n)? is asking whether you wish to connect to another DSN. Respond y(es) or n(o) and hit <return>. Yes brings you back to the Enter ODBC string prompt; no returns "Have a nice day".

```
SQL>quit
Again (y/n) ? n
Have a nice day.
```

5.3.5 Troubleshooting ODBC Connections and Sessions

From the Tracing Tab of the OpenLink ODBC Administrator, the level of Tracing can be configured.

Figure 3.99. OpenLink ODBC Administrator, Tracing tab

When to trace. Use this option to turn tracing off or on. For a One-time only trace, the trace is made for the duration of the next connection. Click the Apply button to make your selection active. This option corresponds to the Trace and TraceAutoStop parameters in the preference files.

Custom trace library. Use the Select Library button to browse the machine for a library file that contains the code to intercept the trace output. This field should generally be left blank. When setting this parameter, you may use the Browse button to locate the library, or specify it manually. If specified manually, note that the location must be specified as it would be in a Terminal session. You may use `~/` as a shortcut to your user Home directory. This option corresponds to the TraceDLL parameter in the preference files.

Log file path. Specify the complete location of the file to which you wish to write the trace. You may use the Browse button to locate or create the file, or specify it manually. If specified manually, note that the location must be specified as it would be in a Terminal session. You may use ~/ as a shortcut to your user Home directory. This option corresponds to the TraceFile parameter in the preference files.

Trace Parameters in Preference Files. When the ODBC tracing parameters are changed, they are saved in the [ODBC] section of the ODBC.preference or odbc.ini file, whichever is active on your system.

The following example shows the settings for a one-time trace to a sql.log on the active user's Desktop.

```
[ODBC]
Trace         = 1
TraceAutoStop = 1
TraceDLL      =
TraceFile     = ~/Desktop/sql.log
```

Once the tracing of a connection is complete, the Trace and TraceAutoStop parameters will both be set to 0.

Trace. This parameter corresponds to the When to trace option in the Tracing tab. Set to 1, tracing is on; set to 0, tracing is off.

TraceAutoStop. This parameter corresponds to the One-time only option in the tracing tab. Set to 1, tracing will automatically stop after the next connection is closed; set to 0, tracing must be manually stopped. When tracing is automatically stopped, the Driver Manager will set both Trace and TraceAutoStop parameters to 0.

TraceDLL. This parameter corresponds to the Custom trace library option in the tracing tab. When setting this parameter, note that the file must be fully specified as it would be in a Terminal session. You may use `~/` as a shortcut to your user Home directory.

5.3.5.1 TraceFile

This parameter corresponds to the Log file path option in the tracing tab. When setting this parameter, note that the file must be fully specified as it would be in a Terminal session. You may use `~/` as a shortcut to your user Home directory.

As of iODBC 3.51.0, there are now more variables that may be included in the filename:

$P. This is the process-id (pid) of the application invoking the driver manager - allowing for per-process log-files

$U. This is the user-id under which the process is currently running

$T. This is the timestamp in YYYYMMDDHHmmSS format

$H. This is the home-directory of the user as whom the process is running

These options are case-insensitive. Hence you can specify

```
        TraceFile=/tmp/iodbc-$U-$T-$P.log
```

if you wish.

5.4 Generic Client for UNIX

ODBC Driver Manager exists in various forms under UNIX, but the definitive Driver Manager for UNIX is known as the "iODBC Driver Manager". Additional information regarding iODBC can be obtained from: www.iodbc.org.

The OpenLink ODBC Client Components for UNIX comprise the following :

- iODBC Driver Manager - A shared library that links ODBC applications to ODBC Drivers
- Generic ODBC Driver - A shared library (the file "oplodbc.so" or "oplodbc.sl") that provides database connectivity and data access services to ODBC-based clients (these can be applications written and compiled using the iODBC SDK)
- Sample ODBC Application - A simple program that can be used to verify your ODBC installation and working environment.

5.4.1 Installation

The OpenLink ODBC Client Components for UNIX are contained within a compressed TAR archive file in the format "XXoczzzz.taz", where the "XX" represents a two-letter platform-specific code.

In addition, the optional ODBC Data Source Administrator and JDBC Samples are contained in an archive of format "XXl5adzz.taz" (availability varies by platform).

Note: The ODBC Data Source Administrator offered for configuring the Client Component is a stripped down version of the OpenLink Admin Assistant used for configuring the server components.

These files are automatically presented to you via the OpenLink Software Download wizard when you enter UNIX as your client operating system.

The steps that follow describe the installation process:

1. Move the "install.sh" and all downloaded *.taz" files to an installation directory of your choice
2. Type in one of the following commands to extract the contents of the compressed TAR archive files:

   ```
   sh install.sh
   ```

   or

   ```
   install.sh
   ```

   or

   ```
   ./install.sh
   ```
3. Setup your operating environment by executing the command:

   ```
   . openlink.sh
   ```

   you can also place the following entry in your ".profile" file:

   ```
   . openlink.sh
   ```
4. Proceed to the configuration stage of this process.

5.4.2 Configuration

The main configuration activity involves setting up logical references to the actual backend database engines that you wish to access via your UNIX based ODBC Driver. These local references are called Universal Data Source Names (UDSNs) and they are responsible for linking ODBC clients with actual OpenLink Data Access Drivers.

The OpenLink Admin Assistant is a Server Based HTML utility that enables you to manage UDSNs via your Web Browser. This utility provides wizards and a forms based user interface for performing it's tasks.

During the installation for Unix, the install.sh script will generate a file called bin/iodbc-admin-httpd.sh. This is a shell script used to start and stop the HTTP based iODBC Administrator. The usage is as follows:

```
$ sh iodbc-admin-httpd.sh start  # Starts the HTTP based iODBC Administrator
$ sh iodbc-admin-httpd.sh stop   # Tries to stop the HTTP based iODBC Administrator
$ sh iodbc-admin-httpd.sh status # Shows if the program is running
```

In the sections that follow, a step by guide and illustrative screen shots are used to demonstrate both approaches to setting up ODBC DSNs.

In the examples below lets presume that we are trying to create a UNIX based ODBC DSN on our machine called "opllinux" that will connect us to a Microsoft SQL Database on a Windows 95/98/NT Server. The critical database connection and network information for this setup (aka connection attributes) are as follows:

Network Alias of a Windows 95/98/NT/2000 Server machine (typically your application server) running OpenLink Server Components: "ntappserver"

Network Alias of a Windows 95/98/NT/2000 Server machine running Microsoft SQL Server: "pluto"

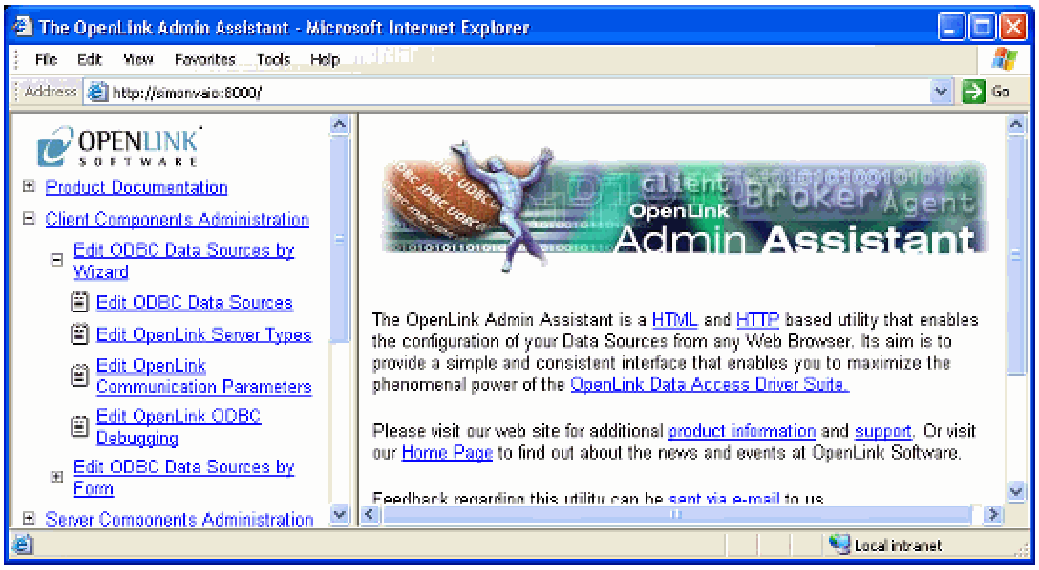
Microsoft SQL Server Database Name: "pubs"

5.4.2.1 Wizard Based Data Source Management

1. Start up the web-based Admin Assistant available for use, if not already started:

   ```
   bash$ sh iodbc-admin-httpd.sh start
   ```
2. Open up your Internet Browser and then enter the following URL: http://localhost:8000/ (note the OpenLink Web Assistant listens at port 8000 by default, this value is set at installation time).
3. Expand the menu by clicking on the "Client Components Administration", then "Data Source Name Configuration", and "Edit Data Sources by Wizard".

   Figure 3.100.


The OpenLink Admin Assistant - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://simonvaio:8000/
Product Documentation
Client Components Administration
Edit ODBC Data Sources by Wizard
Edit ODBC Data Sources
Edit OpenLink Server Types
Edit OpenLink Communication Parameters
Edit OpenLink ODBC Debugging
Edit ODBC Data Sources by Form
Server Components Administration
OpenLink Admin Assistant
Local intranet

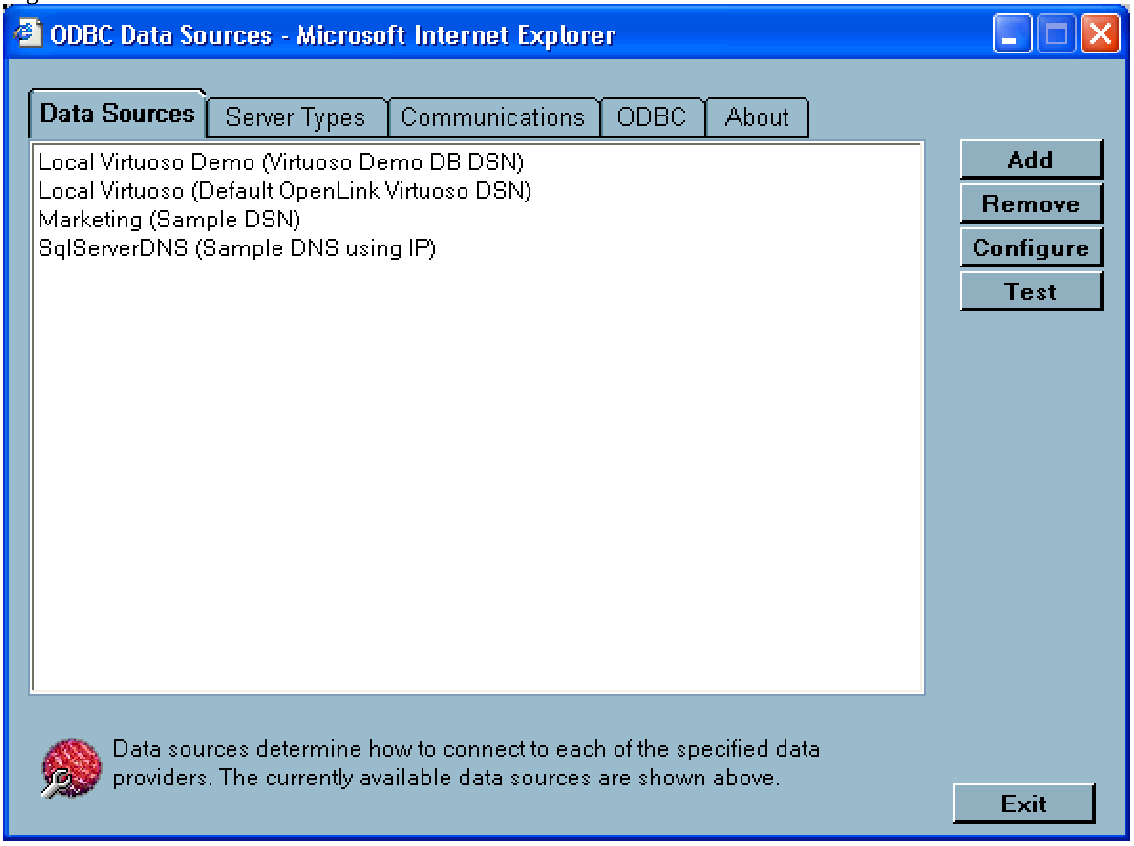
The OpenLink Admin Assistant is a HTML and HTTP based utility that enables the configuration of your Data Sources from any Web Browser. Its aim is to provide a simple and consistent interface that enables you to maximize the phenomenal power of the OpenLink Data Access Driver Suite.

Please visit our web site for additional product information and support. Or visit our Home Page to find out about the news and events at OpenLink Software.

Feedback regarding this utility can be sent via e-mail to us.

4. Click on the "Edit ODBC Data Sources" hyperlink, this takes you into the actual ODBC Data Source configuration wizard.

Figure 3.101. Data Source Wizard

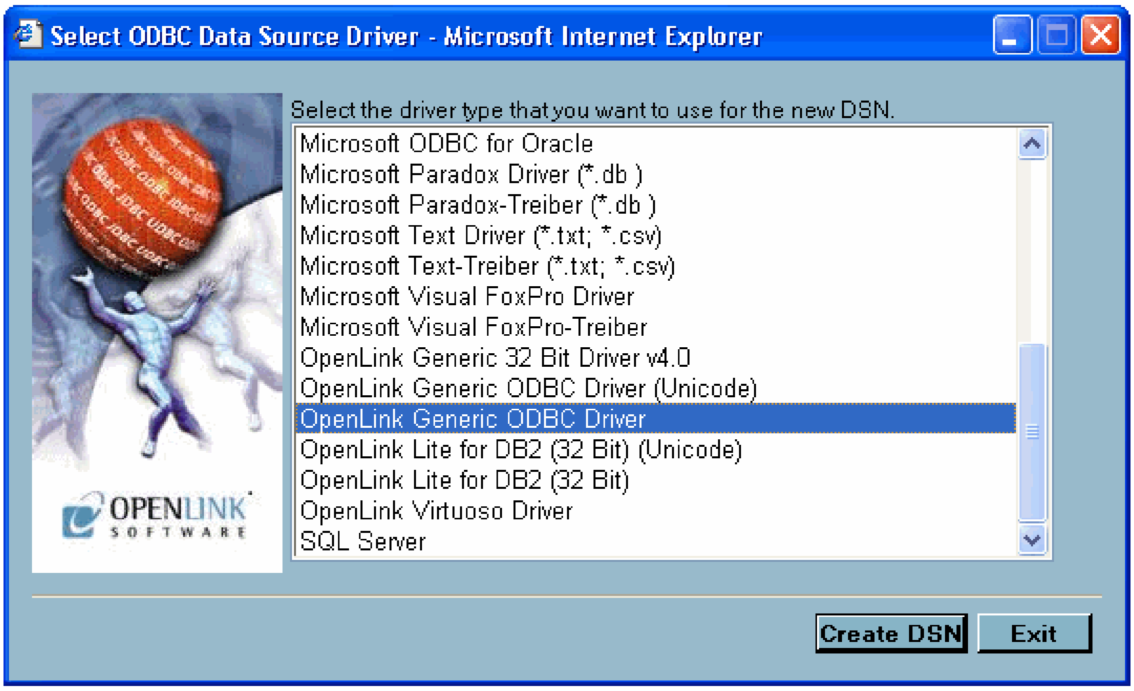


Local Virtuoso Demo (Virtuoso Demo DB DSN)
Local Virtuoso (Default OpenLink Virtuoso DSN)
Marketing (Sample DSN)
SqlServerDNS (Sample DNS using IP)

Data sources determine how to connect to each of the specified data providers. The currently available data sources are shown above.

5. Click the "Add" button to commence the process of creating a new ODBC DSN, the wizard presents you with a list of ODBC Drivers installed on your system, select the driver identified as "OpenLink Generic ODBC Driver" and then click on the "Create DSN" button.

Figure 3.102. Data Source Wizard


Select ODBC Data Source Driver - Microsoft Internet Explorer

Select the driver type that you want to use for the new DSN.

- Microsoft ODBC for Oracle
- Microsoft Paradox Driver (*.db)
- Microsoft Paradox-Treiber (*.db)
- Microsoft Text Driver (*.txt; *.csv)
- Microsoft Text-Treiber (*.txt; *.csv)
- Microsoft Visual FoxPro Driver
- Microsoft Visual FoxPro-Treiber
- OpenLink Generic 32 Bit Driver v4.0
- OpenLink Generic ODBC Driver (Unicode)
- OpenLink Generic ODBC Driver
- OpenLink Lite for DB2 (32 Bit) (Unicode)
- OpenLink Lite for DB2 (32 Bit)
- OpenLink Virtuoso Driver
- SQL Server

6. Enter values into the "Name", "Description" and "Server" fields as follows:

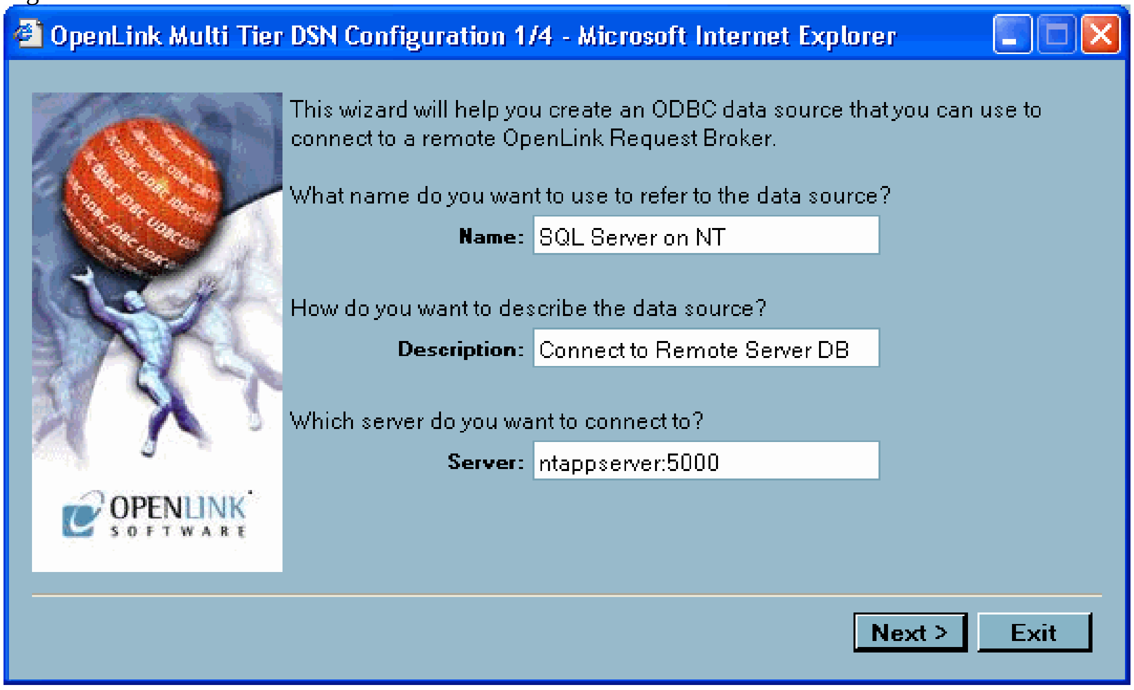
   Name. Enter values that uniquely identify the DSN being created, our example uses the name "SQL Server on NT" to indicate that this DSN will be connecting you to a SQL Server database on an NT server.

   Description. Enter values that provide additional information that helps in describing the purpose of the DSN that you are creating.

   Server. Enter the hostname or IP address that identifies a Server Machine running OpenLink Server Components, that speak the OpenLink Data Access Protocol. Follow this with a semicolon and a TCPIP Port number of the Broker to contact if the default of 5000 is not used. This field corresponds to the Listen parameter on the target Broker.

   Once completed click on the "Next" button.

   Figure 3.103. Data Source Wizard


OpenLink Multi Tier DSN Configuration 1/4 - Microsoft Internet Explorer

This wizard will help you create an ODBC data source that you can use to connect to a remote OpenLink Request Broker.

What name do you want to use to refer to the data source?

Name: SQL Server on NT

How do you want to describe the data source?

Description: Connect to Remote Server DB

Which server do you want to connect to?

Server: ntappserver:5000

7. Now select the database and configuration details:

5.4.2.1.1 Domain

This is how you pick the Database Engine Type that your ODBC DSN is to be associated with e.g. Informix 7, Oracle 7, Progress 7 etc.

The default offered from the Zero Configuration is typically accepted. An alternative compatable Domain may be chosen for a specific customisation.

5.4.2.1.2 Database

This is how an actual database name within the Provider Type Domain is identified, for instance "stores7" indicates an "Informix 7" database called "stores7". This option corresponds to the Database parameter in the preference files.

In this case our example uses the database "pubs"

Connection Options. This is where you place any database specific database connection options. This field in a majority of cases should be left blank by default. In this case enter valid SQL Server database server connection values (where "-s pluto" represent an actual SQL Server instance currently available on your network).

5.4.2.1.3 Connect now..

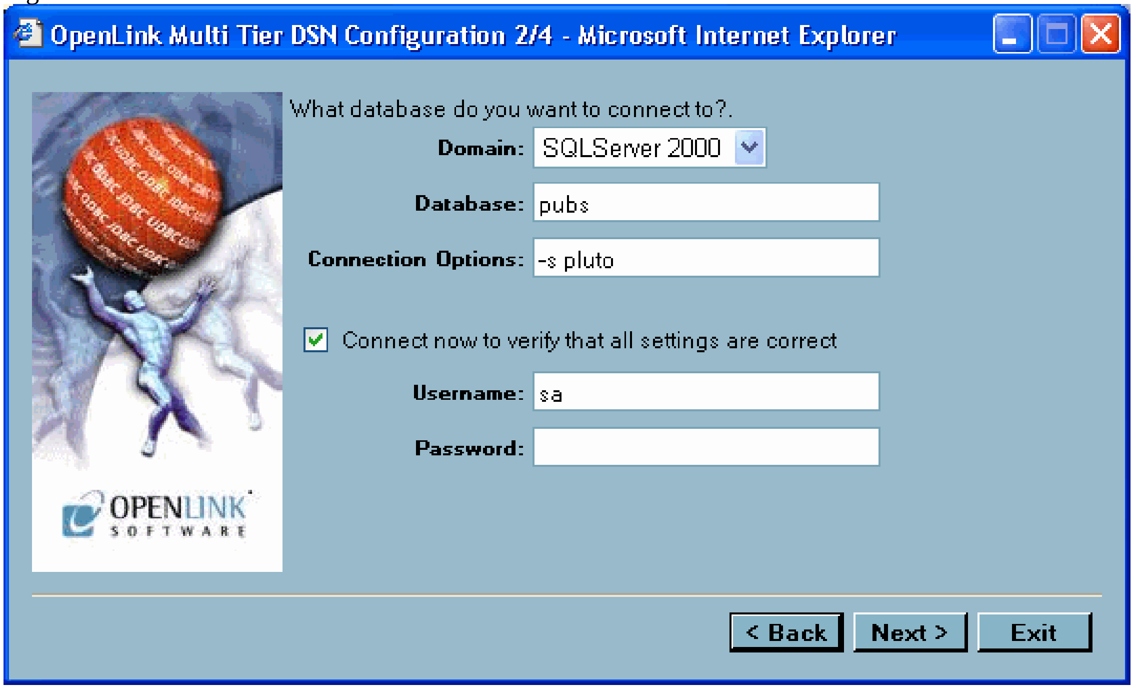
When this tick box is checked, a test connection is made to verify the Data Source connection.

If there is no check then the Login ID and Password fields are ignored, and no test is performed.

Login ID. The default database UserID to use when logging on to a remote database engine (identified by the Domain above).

Password. The Password for the login of the above UserID.

Figure 3.104. Data Source Wizard


OpenLink Multi Tier DSN Configuration 2/4 - Microsoft Internet Explorer
OPENLINK SOFTWARE
< Back
Next >
Exit

What database do you want to connect to?

Domain: SQLServer 2000

Database: pubs

Connection Options: -s pluto

✓ Connect now to verify that all settings are correct

Username: sa

Password:

Click on the "Next" button.

8. Now define additonal connection parameters:

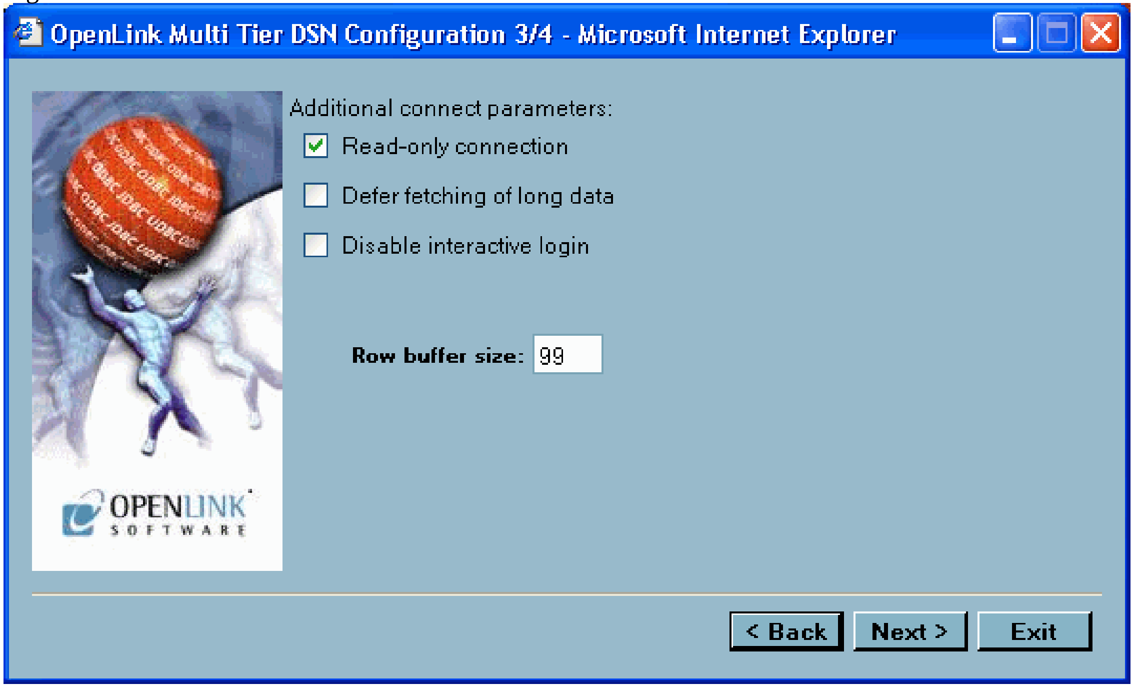
   Read-only connection. Specify whether the connection is to be "Read-only". Make sure the checkbox is unchecked to request a "Read/Write" connection.

   Defer fetching of long data. Check this box to defer the fetching of long data.

Disable interactive login. Suppress the ODBC "Username" and "Password" login dialog box when interacting with your ODBC DSN from within an ODBC compliant application.

Row Buffer Size. This attribute specifies the number of records to be transported over the network in a single network hop. Values can range from 1 to 99.

Figure 3.105. Data Source Wizard


OpenLink Multi Tier DSN Configuration 3/4 - Microsoft Internet Explorer
OPENLINK SOFTWARE
< Back
Next >
Exit

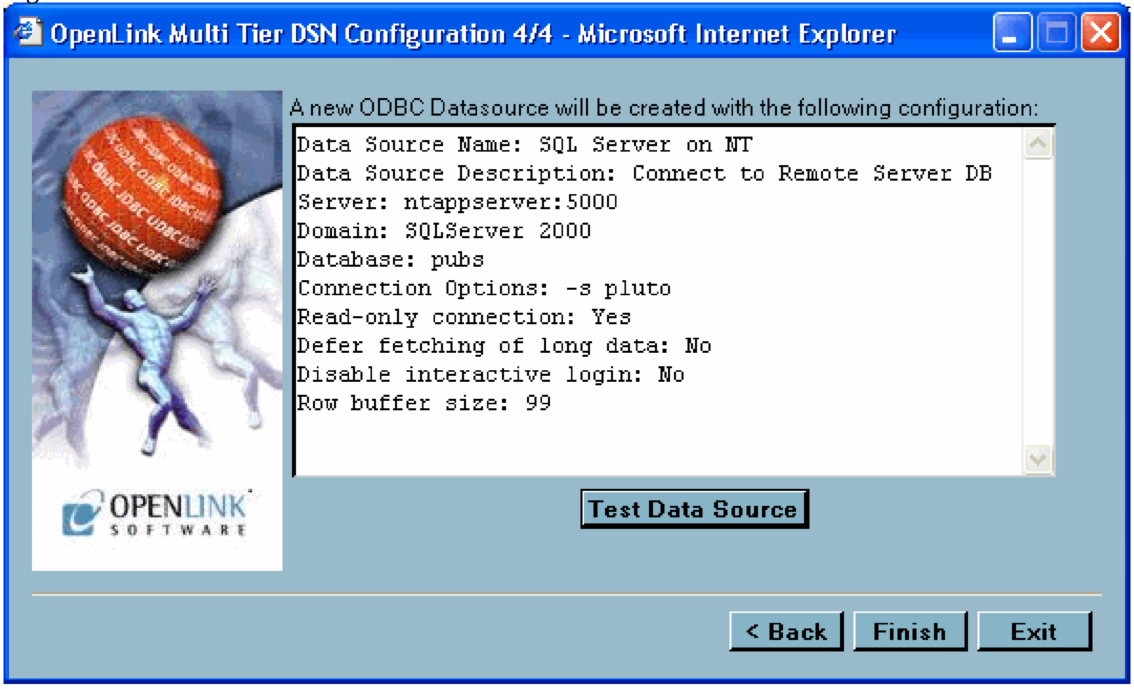
Additional connect parameters:

- [✓] Read-only connection
- [ ] Defer fetching of long data
- [ ] Disable interactive login

Row buffer size: 99

9. The list of parameters for the DSN are now shown. The Test Data Source button will trigger a test of the DSN using the existing login parameters.

Figure 3.106. Data Source Wizard


OpenLink Multi Tier DSN Configuration 4/4 - Microsoft Internet Explorer
OPENLINK SOFTWARE
Test Data Source
< Back
Finish
Exit

A new ODBC Datasource will be created with the following configuration:

```
Data Source Name: SQL Server on NT
Data Source Description: Connect to Remote Server DB
Server: ntappserver:5000
Domain: SQLServer 2000
Database: pubs
Connection Options: -s pluto
Read-only connection: Yes
Defer fetching of long data: No
Disable interactive login: No
Row buffer size: 99
```

10. You have now completed entering all the values that make up your new ODBC DSN, these values are collectively known as your ODBC DSN Attributes. Click on the "Finish" button in order to store these values permanently on your hard disk.
11. Click on the "exit" button to exit the ODBC DSN configuration wizard

5.4.2.2 Form Based Data Source Management

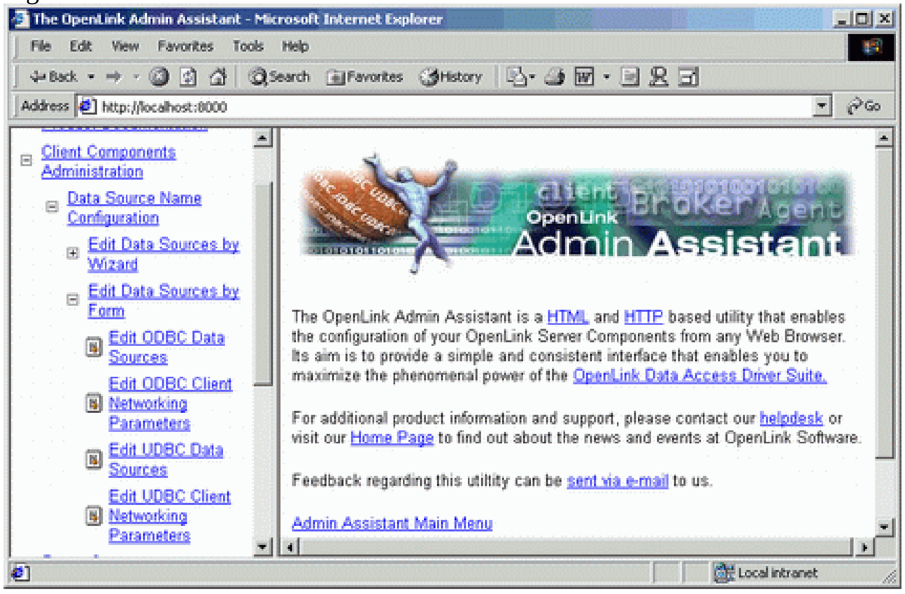
The OpenLink Admin assistant also allows the more experience OpenLink ODBC user to manage ODBC DSNs via a forms based interface. Like the wizard based approach this is done entirely from within your browser. In the sections that follow, a step by guide and illustrative screen shots are used to demonstrate the process of creating the same ODBC DSN created in the prior section using the Wizard approach.

1. Start up the web-based Admin Assistant available for use, if not already started:

   ```
   bash$ sh iodbc-admin-httpd.sh start
   ```
2. Enter the following URL into your Web Browser (if the Admin Assistant isn't already initialized): http://localhost:8000 You will be presented with a screen similar to the one below. Notice that the "Client Component Administration", "Data Source Names Configuration" and "Edit Data Sources By Form" hyperlinks have been expanded.

   Click on the "Edit ODBC Data Sources" hyperlink to commence the process of creating a new ODBC DSN.

   Figure 3.107. Data Source Forms


The OpenLink Admin Assistant - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://localhost:8000
Go
Client Components Administration
Data Source Name Configuration
Edit Data Sources by Wizard
Edit Data Sources by Form
Edit ODBC Data Sources
Edit ODBC Client Networking Parameters
Edit UDBC Data Sources
Edit UDBC Client Networking Parameters
client Broker Agent
OpenLink
Admin Assistant
Local intranet

The OpenLink Admin Assistant is a HTML and HTTP based utility that enables the configuration of your OpenLink Server Components from any Web Browser. Its aim is to provide a simple and consistent interface that enables you to maximize the phenomenal power of the OpenLink Data Access Driver Suite.

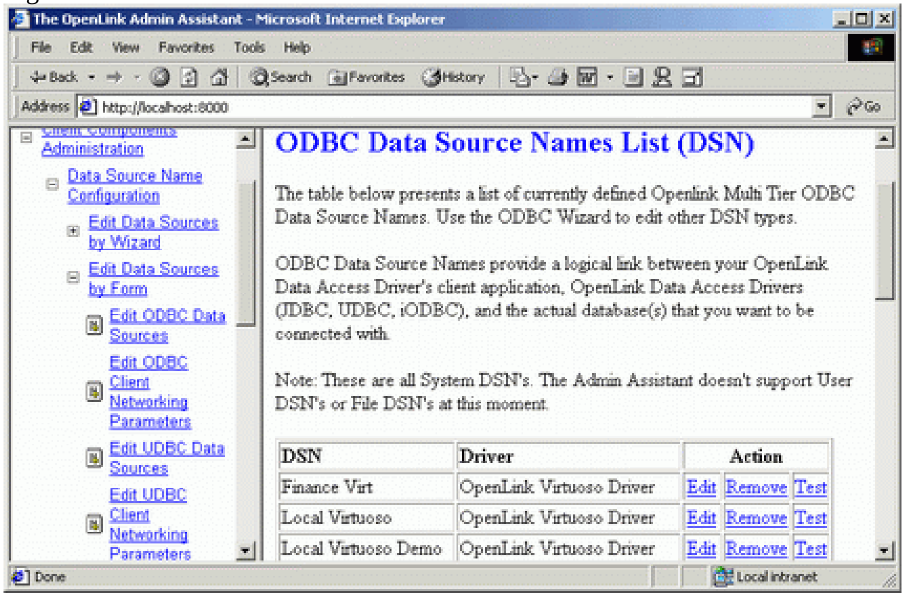
For additional product information and support, please contact our helpdesk or visit our Home Page to find out about the news and events at OpenLink Software.

Feedback regarding this utility can be sent via e-mail to us.

Admin Assistant Main Menu

3. On the right side of the Admin Assistant pane is your start page for configuring ODBC DSNs using the Forms approach. This page presents to you a list of currently configured ODBC DSNs on the machine. Click the "Add" hyperlink in the Action Column

   Figure 3.108. Data Source Forms


The OpenLink Admin Assistant - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://localhost:8000
Go
Client Components Administration
Data Source Name Configuration
Edit Data Sources by Wizard
Edit Data Sources by Form
Edit ODBC Data Sources
Edit ODBC Client Networking Parameters
Edit UDBC Data Sources
Edit UDBC Client Networking Parameters
Done
Local intranet

ODBC Data Source Names List (DSN)

The table below presents a list of currently defined Openlink Multi Tier ODBC Data Source Names. Use the ODBC Wizard to edit other DSN types.

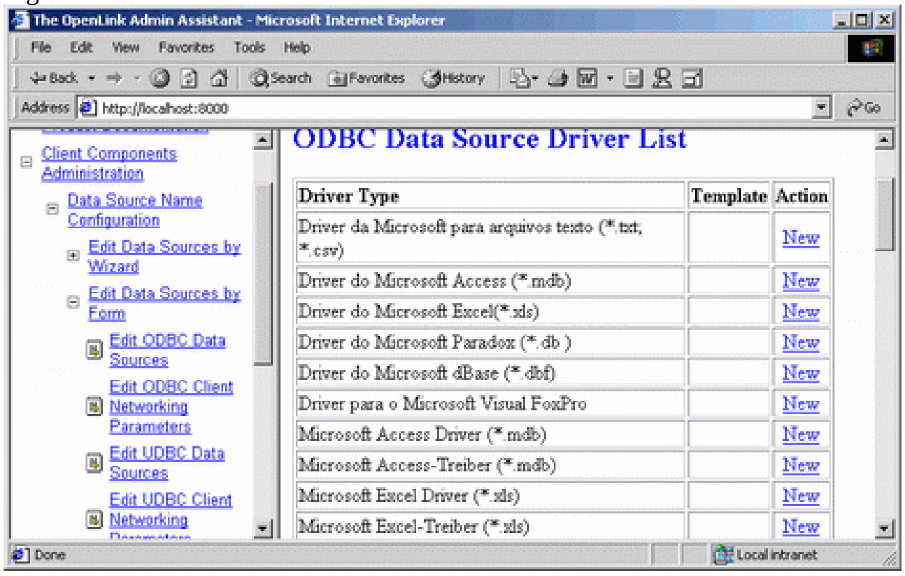
ODBC Data Source Names provide a logical link between your OpenLink Data Access Driver's client application, OpenLink Data Access Drivers (JDBC, UDBC, iODBC), and the actual database(s) that you want to be connected with.

Note: These are all System DSN's. The Admin Assistant doesn't support User DSN's or File DSN's at this moment.

DSN	Driver	Action		
Finance Virt	OpenLink Virtuoso Driver	Edit	Remove	Test
Local Virtuoso	OpenLink Virtuoso Driver	Edit	Remove	Test
Local Virtuoso Demo	OpenLink Virtuoso Driver	Edit	Remove	Test

4. You are now presented with a table listing that comprises ODBC Drivers installed on your system, move on to the row that identifies the ODBC Driver that you will be creating your DSN for, then click on the "New" hyperlink.

Figure 3.109. Data Source Forms


The OpenLink Admin Assistant - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://localhost:8000
Go
Client Components Administration
Data Source Name Configuration
Edit Data Sources by Wizard
Edit Data Sources by Form
Edit ODBC Data Sources
Edit ODBC Client Networking Parameters
Edit UDBC Data Sources
Edit UDBC Client Networking
Done
Local intranet

ODBC Data Source Driver List

Driver Type	Template	Action
Driver da Microsoft para arquivos texto (*.txt; *.csv)		New
Driver do Microsoft Access (*.mdb)		New
Driver do Microsoft Excel(*.xls)		New
Driver do Microsoft Paradox (*.db)		New
Driver do Microsoft dBase (*.dbf)		New
Driver para o Microsoft Visual FoxPro		New
Microsoft Access Driver (*.mdb)		New
Microsoft Access-Treiber (*.mdb)		New
Microsoft Excel Driver (*.xls)		New
Microsoft Excel-Treiber (*.xls)		New

5. Enter values into the fields presented on the ODBC DSN form as follows:

Name. enter values that uniquely identify the DSN being created, our example uses the name "SQL Server on NT" to indicate that this DSN will be connecting you to a SQL Server database on an NT server.

Comment. enter values that provide additional information that helps in describing the purpose of the DSN that you are creating.

UserName. enter a valid username for the database that you are connecting to, you can leave this blank and be prompted for values at actual database connect time.

Database Name. enter the name of an actual SQL Server database, in this case our example uses the database "pubs"

Read-only connection. check this box if you require a read only session.

No Login Dialog Box. check this box if you do not to be prompted by your ODBC Driver for username and password dialog box at connect time.

Defer fetching of long data. Check this box to defer the fetching of long data. See the Release Notes section for more details.

Database Server. enter database server connection values for the database that your are connecting to, in this case enter valid SQL Server database server connection values (where "-s pluto" represent an actual SQL Server instance currently available on your network).

Server Type. enter a value that identifies the type of OpenLink Agent that will serve your ODBC client.

Protocol. Chose the type of network connection to be used from TCP/IP, SPX/IPX and DECnet. Note some operating systems do not support all the choices. In most cases uses TCP/IP.

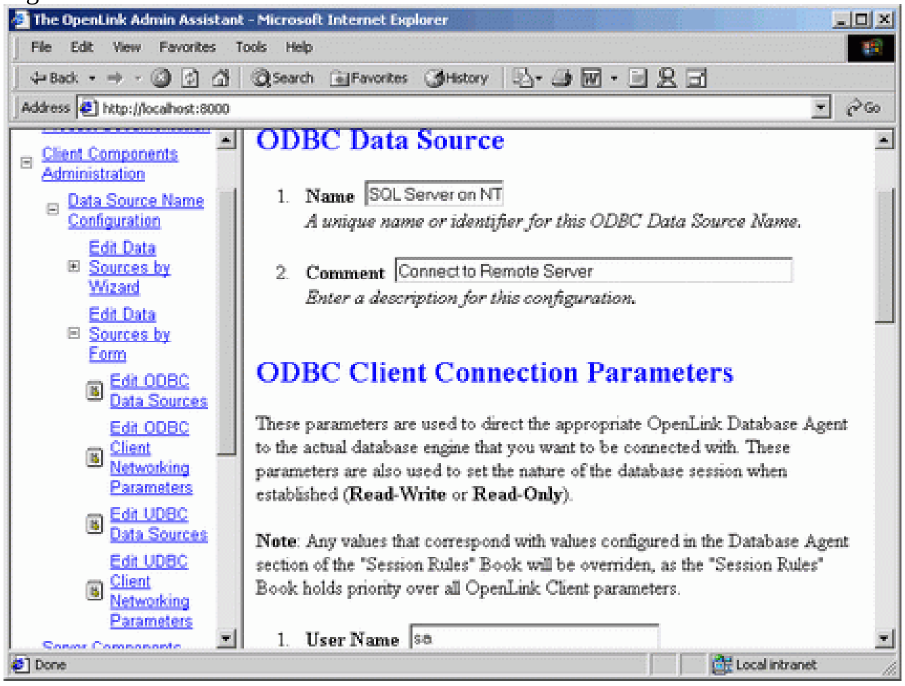
Hostname. enter a value that identifies the server machine running your OpenLink Server Components.

Port. Enter the TCP port number of the Broker to contact. This corresponds to the Listen parameter on the target Broker.

Row Buffer Size. enter a value that represents the number of records that you would like your ODBC driver to retrieve during each network hop. A network hop represents the number of times your OpenLink ODBC send a message across the network to retrieve records from your remote database server. The feature can be used to improve ODBC record retrieval performance.

Note: The screen shot below is a snapshot of the ODBC DSN for, click on the right-hand scroll bar to see all the fields described above.

Figure 3.110. Data Source Forms


The OpenLink Admin Assistant - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://localhost:8000
Go
Client Components Administration
Data Source Name Configuration
Edit Data Sources by Wizard
Edit Data Sources by Form
Edit ODBC Data Sources
Edit ODBC Client Networking Parameters
Edit UDBC Data Sources
Edit UDBC Client Networking Parameters
Server Components
Done
Local intranet

ODBC Data Source

1. **Name** SQL Server on NT
A unique name or identifier for this ODBC Data Source Name.

2. **Comment** Connect to Remote Server
Enter a description for this configuration.

ODBC Client Connection Parameters

These parameters are used to direct the appropriate OpenLink Database Agent to the actual database engine that you want to be connected with. These parameters are also used to set the nature of the database session when established (**Read-Write** or **Read-Only**).

Note: Any values that correspond with values configured in the Database Agent section of the "Session Rules" Book will be overriden, as the "Session Rules" Book holds priority over all OpenLink Client parameters.

1. **User Name** sa

6. Click on the "Add" button at the foot of the page to complete the creation of your new ODBC DSN.
7. The DSN may then be tested, or you can return to the DSN list.

5.4.2.3 ODBC Driver Session Settings

A number of configuration session parameters are available to adminstrators of OpenLink ODBC Drivers, these parameters can be managed via the Web Browser based Admin Assistant or by manually editing the file "udbc.ini" situated in the "bin" sub-directory of your OpenLink installation directory. These parameter enable you tailor the behaviour of your ODBC Drivers for UNIX in line with the requirements of your ODBC based solutions and any general infrastructural requirements that you may have.

The list of configurable session parameters and their descriptions are as follows:

Table 3.4.

Parameter	Default Value	Description
BrokerTimeout	30	The time (in secs) that the OpenLink ODBC client application will wait for the OpenLink Request Broker to accept or reject a database connection.
ReceiveTimeout	60	The time (in secs) that the OpenLink ODBC client will wait for an ODBC request to be completed.
RetryTimeout	5	The amount of wait time (in secs) before the OpenLink ODBC client attempts to re-execute a failed call. After each attempt this value is doubled. The life time of this value never exceeds the BrokerTimeout during intial connection establishment, and never exceeds the ReceiveTimeout when sessions have been established.
SendSize	4096	The size (in kilobytes) of the OpenLink ODBC client's outward bound message packets.
ReceiveSize	16000	The size (in kilobytes) of the OpenLink ODBC client's server bound message packets.
DebugFile	empty	When this variable contains a valid file and path reference, all ODBC API calls will be logged and stored in the file name referenced.

5.4.3 Sample Application

OpenLink also provides a sample ODBC based dynamic SQL application that enables you verify usability of your ODBC installation. The sample application is situated within the "samples" sub-directory below your OpenLink installation directory. The ODBC sample application is called "odbctest".

The following steps guide you through the process of successfully utilising this sample application. This exercise presumes that we are connecting to a DSN called "SQL Server on NT", which connects us to a remote SQL Server Database hosted on a machine called "pluto" via the OpenLink Server components on an Windows 95/98/NT/2000 application server called "ntappserver".

1. Ensure that your OpenLink Request Broker is up and running on the machine "ntappserver" (you can quickly confirm this by opening up your browser and entering the following URL: http://ntappserver:8000)
2. At your UNIX command prompt type in the following command:

   ```
   odbctest
   ```
3. Enter a full or partial ODBC connect string at the ODBC applications command prompt, some examples are listed below:

   - for a list of DSNs on your system enter "?"
   - to connect to the DSN called "SQL Server on NT" type (this is a partial connect string): DSN=SQL Server On NT
   - to enter a username and a blank password combination along with the DSN type (this is a partial connect string only because we have a seperate server hosting the OpenLink Server and Microsoft SQL Server components): DSN=SQL Server on NT;UID=sa;PWD=
   - to enter a directive that instructs the OpenLink Server components to connect to the remote SQL Server hosted on the machine called "pluto", type the following (this is a full connect string for this particular scenario): DSN=SQL Server on NT;UID=sa;PWD=;OPTIONS=-s pluto
4. If the previous step is successful you are now ready to execute SQL interactively against your remote database, to do this enter the following SQL command:

   ```
   select * from authors
   ```
5. To quit this application type in "exit" at the SQL command line prompt.

5.4.4 UDBC Data Source Configuration

UDBC shares its API with ODBC, and is provided for platforms that do not have a suitable ODBC driver manager - it is for legacy unix systems which did not support dynamic library functions. As such, it is deprecated, and documented here only for completeness.

UDBC, like ODBC, is based on the notion of logical references to database backends through the use of Data Source Names. Under UDBC, these are described as Universal Data Source Names (UDSNs) due to the cross data access standard nature of these Data Source Names (they are usable by OpenLink Drivers for ODBC and JDBC).

The OpenLink Universal Data Access Driver Suite includes an HTTP based utility called the OpenLink Admin Assistant that enables you create, configure, and manage UDSNs via a Web Browser. Since this is HTTP based, you can even do this remotely, security measures exist avoid unwanted tampering. The Admin Assistant provides a wizards and HTML forms based user interfaces.

You identify your UDSN file to UDBC Drivers via the Environment Variable "UDBCINI". Your OpenLink UDBC SDK installation process will place a sample of this file in the 'doc' installation subdirectory. The OpenLink shell script ("openlink.sh") will look for the runtime copy of this file in the 'bin' directory of your installation, and export it to the $UDBCINI environment variable. Manually, the UDBCINI environment variable is set using the command:

```
UDBCINI=/usr/openlink/bin/udbc.ini ; export UDBCINI
```

The above presumes that your OpenLink installation's base directory is: /usr/openlink. Use an appropriate directory in its place.

5.4.5 See Also:

The UDBC Client Components SDK chapter.

Configuring UDBC data sources using: Wizards Based Assistant or Forms Based Assistant

5.5 New Features

- ### 5.5.1 Unicode Driver support

  Database agents support the latest releases of all supported database engines this includes:

  1. Oracle 8i & 9i
  2. Microsoft SQL Server 2000
  3. DB/2 v7
  4. ODBC Agent
  5. JDBC Agent
  6. Informix 9
  7. Sybase Adaptive Server 12.5
  8. Progress 9 (SQL-92)

  Note that there are two Multi-tier generic ODBC clients:

  1. With Unicode support olod5032u.dll (OpenLink Generic ODBC Driver (Unicode))
  2. Without Unicode support olod5032.dll (OpenLink Generic ODBC Driver)

  Both of these ODBC clients can be used with a Unicode enabled agent.
- Zero Configuration. The OpenLink Request Broker can now be configured to Broadcast ZeroConfig services of available Database agents on the network, which then can be discovered by the the OpenLink Multi-Tier ODBC client. Enabling ODBC Datasources to be configured and/or connections made with no knowledge of network topology.
- Significant SQL Server and SYBASE Driver Enhancements. Our Drivers are now built to communicate directly with Microsoft SQL Server and SYBASE ASE using the TDS protocol (the native wire protocol for both database servers). This also implies that no additional software is required post installation in order for our ODBC Drivers to communicate with these Drivers (this applies to the Single Tier format Drivers only). The use of TDS has also enabled us to double the performance of both our Single Tier and Multi-Tier Drivers for these databases.
- Array Optimisations on Select Queries. All drivers now support the SQLSetStmtAttr(SQL_ARRAY_SIZE) call for batch select statements, providing improved performance when re-execute select statements with bound paramters.
- Deferred Fetching. The Release 4 OpenLink driver family brings marked communications layer improvements. Central to these are improved implementation of the `SQLGetData` and `SQLPutData` ODBC function calls.

  Wholesale vs. Piecemeal Data Transfer. In previous releases of the drivers, parameter data at query execution was assembled in the OpenLink driver's client component and transferred to its server component in a single network hop. (The client and server components apply to both the Single- and Multi-Tier drivers; they refer to different layers within the driver entity.) Similarly, when fetching from a "long", or large binary data column, data was transferred from the driver's server component to the client component in a single network transfer. The only way data could manipulated in a piecemeal fashion, was within the ODBC application from the driver's client component (client side only).

  The Release 4 driver family now allow transferring parameter data in parts over the network between the client and server driver components. Once transferred, the fragmented column data are re-pieced together in their entirety within the client and server portions of the driver.

  Deferred Fetching. When fetching, data from columns with "long" data are only transferred between the server and client components if one of the following applies to that column:

  It has been "bound" by the application via the SQLBindCol API call

  It has been retrieved via the SQLGetData API call

This mechanism is referred to as deferred fetching. In this method, as "long" column data is not reassembled within the OpenLink ODBC client itself (rather, within the application), driver memory overhead incurred is dramatically reduced. Deferred fetching applies to the following "long" database column types:

ODBC agent

- SQL_LONGVARCHAR
- SQL_LONGVARBINARY

DB/2 agent

- SQL_BLOB
- SQL_CLOB
- SQL_DBCLOB
- If the long data compatibility option has been specified in the DB2 database
- SQL_LONGVARCHAR
- SQL_LONGVARBINARY
- SQL_LONGVARGRAPHIC

Oracle agent

- SQLT_BLOB
- SQLT_CLOB

Sybase agent

- CS_IMAGE_TYPE
- CS_TEXT_TYPE

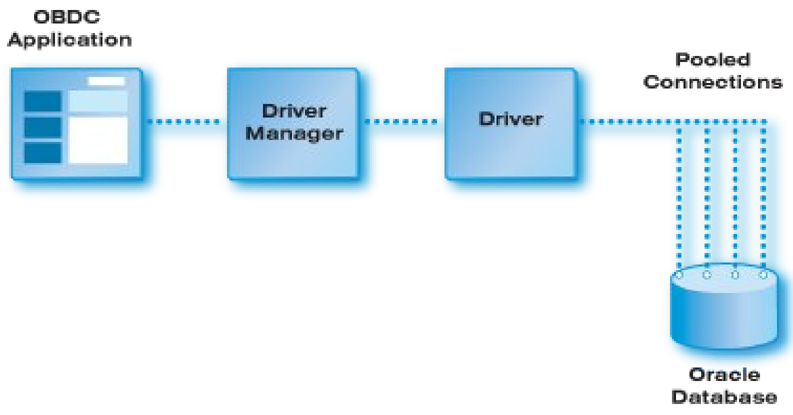
However, if a table contains a column defined as one of these "long" types, but the actual data stored in the column only fills a small proportion of the available space, deferred fetching is of no benefit. In these cases, performance may be improved by switching off the deferred fetching mechanism using the control in the OpenLink Generic Client data source setup dialog.

5.6 Oracle Connection Pooling Support

5.6.1 What it is

Oracle Connection-Pooling gives you the ability to retain a pool of open connections to the database within the ODBC Driver, thereby avoiding costly connection-establishment overhead for every connection from the client. Being implemented in the ODBC Driver means you do not have to rely on the Driver Manager component implementing connection-pooling for you, and have finer control over how it behaves too.

Figure 3.111. orapooling.png



5.6.2 Benefits

OCI separates the concept of database sessions (user logins) from physical connections (server attachments). By pooling connections, it is possible for an application to multiplex several sessions over fewer physical connections. Because the number of physical connections is less than the number of database sessions in use by the application and because the shared pool of physical connections typically has a corresponding back-end server pool containing an identical number of dedicated server processes; the number of back-end server processes is also reduced by connection pooling. Thus many more database sessions can be multiplexed. If the number of back-end server processes may cause scaling problems on the database, OCI connection pooling can be of benefit.

Client-side connection pooling by the driver manager may be done on a per-process basis. For instance, on Unix each application gets its own driver-manager instance in its process. In contrast, in a multi-threaded OpenLink agent with connection re-use enabled, an agent connection pool can span different client processes. Cross-client connection pooling may be preferable to client-side connection pooling in certain application domains and may offer better scalability. For instance, if the agent is used by an application server or web server.

5.6.3 How it Works

If OCI connection pooling is enabled, a separate connection pool is created for each Oracle instance connected to. All connections specifying the same server instance (i.e. the same TNS service name) share the same connection pool. The connection pool is reference counted and only destroyed when the last Lite/agent connection using it is closed down.

Each agent connection holds a reference to an OCI connection pool record or OCPR. The OCPR contains a handle to the OCI connection pool associated with the Oracle instance connected to. The agent maintains a hash table of OCPRs, keyed by instance name. Each OCPR is reference counted. The reference count for an OCPR is incremented each time an agent uses the connection pool to connect to the associated Oracle instance, and decremented when the agent connection is closed. The OCPR is destroyed, and the connection pool closed, when the last agent connection using it is closed.

When an agent is about to establish a connection to an Oracle instance, it searches for an OCPR for the instance. If one exists, a connection pool for that instance already exists. The agent requests an OCI connection from this pool and increments the reference count of the OCPR. If one doesn't exist, the agent creates an OCI connection pool for the instance by registering a new OCPR.

Note: different ODBC DSNs, using this driver and the same server instance, may specify different pool settings (max pool size, min pool size etc.) The settings of the first DSN to connect fixes the characteristics of the pool.

The maximum number of connection pools supported by an Oracle agent or Lite driver is 100. Each connection pool allows up to 500 connections, depending on the value set in the environment variable OCI_CONNPOOL_MAX or the Lite driver setup dialog. The maximum linger time is 300 seconds.

5.6.4 The OpenLink Perspective: what you do to use it

When an OCI connection pool is created, it is characterised by several parameters:

- the minimum number of connections to be opened when the pool is created;
- the maximum number of connections that can be opened in the pool;
    - ♦ When the maximum number of connections are open in the pool, and all the connections are busy, if a call needs a connection, it will wait until it gets one if connWait is TRUE, if FALSE an error is returned.
- the incremental number of connections to be opened when all the connections are busy and a call needs a connection.
    - ♦ This increment is used only when the total number of open connections is less than the maximum number of connections that can be opened in that pool.
- whether, when all connections in the pool are found to be busy and the number of connections has already reached the maximum, to wait for a connection or report an error.
- how long the last connection in the pool should linger before the pool is destroyed (this option is available for the single-tier WIN32 drivers only)

5.7 Multi-Tier-specific Connection-Pooling

5.7.1 Multi-Tier-specific configuration

The required parameters are configured through environment-variables - either use the web-based Administrative Assistant (normally located at <http://server:8000/>) navigating through `Server Components Administration', `Database Agent Administration', `Database Agent Settings by Form', select the name of the agent section used, and scroll to the section `OpenLink Agent Startup Parameters', or edit the rulebook file (oplrqb.ini) directly, to assert settings for the OCI_CONNPOOL_* settings below:

```
[Environment ORACLE90]
ORACLE_HOME        = /dbs/oracle90
ORACLE_SID         = ORCL
;ORACLE_SERVER      = T
;TWO_TASK           = P:
;ODBC_CATALOGS      = Y         ; Uncomment after loading odbccat9.sql
;MULTIPLEX_LDA      = 5         ; Allow 5 OpenLink clients on a single lda
;OPL_USR_TBLS_FIRST = Y         ; Sort SQLTables starting with user tables
SHOW_REMARKS       = N  ; Retrieve SQLColumns REMARKS field
CURSOR_SENSITIVITY = LOW        ; Set to HIGH after loading odbccat9.sql
;OCI_PREFETCH_ROWS  = 100       ; Number of rows to prefetch
;OCI_PREFETCH_MEMORY = 65535    ; Amount of memory to use for prefetching
;LD_LIBRARY_PATH = /dbs/oracle81/lib    ; Find shared libraries
;SHLIB_PATH      = /dbs/oracle81/lib    ; Find HP/UX shared libraries
;LIBPATH         = /dbs/oracle81/lib    ; Find AIX shared libraries
;NLS_LANG        = AMERICAN_AMERICA.UTF8 ; Uncomment for Unicode connections
OCI_CONNPOOL_MIN = 0          ; minimum initial connections in pool
OCI_CONNPOOL_INCR = 1         ; number of new connections when others busy
OCI_CONNPOOL_MAX = 1          ; sum of MIN+INCR
OCI_CONPOOL_WAIT = TRUE       ; whether new query waits or errors when all busy
OCI_CONNPOOL_LINGER = 0       ; seconds to linger after last connection cleared
```

OCI connection pooling is disabled by setting a minimum connection pool size of 0.

5.8 Oracle 10g Instant Client: Connection String Formats

The Oracle 10g Instant Client can be used in OpenLink single-tier and multi-tier environments. In a three-tier configuration, an OpenLink Oracle 10g agent residing on the middle-tier can use the Instant Client to connect to an Oracle instance on the third tier.

Instant Client allows you to run applications *without installing the standard Oracle client* (SQL*Net or Net8) or having an ORACLE_HOME. It supports two basic connection string formats:

- //host:[port][/service name]
    - e.g. //dbase-server-5:4321/ORDERS
- an Oracle Net keyword-value pair such as

```
(DESCRIPTION=(ADDRESS=(PROTOCOL=tcp) (HOST=dlsun242)
    (PORT=5521)) (CONNECT_DATA=(SERVICE_NAME=bjava21)))
```

Naming methods that require a configuration file (such as tnsnames.ora or sqlnet.ora) for name translation can also be used if the environment variable TNS_ADMIN is set to point to the directory containing the file.

5.8.1 OpenLink Multi-Tier Connections using Instant Client

In this scenario, the Instant Client need only be installed on the machine hosting the OpenLink Oracle agent. The valid combinations of connection parameters (NET 10 Service Name, Login ID and Password) accepted by the OpenLink Multi-Tier DSN Configuration Wizard are the same as detailed above for the Single-Tier driver, with the 'NET 10 Service Name' field in the multi-tier wizard taking the same values as the 'Connection String' field in the single-tier wizard. The environment variables ORACLE_HOME and ORACLE_SID need not be set in the rulebook. However, TNS_ADMIN must be set in the rulebook if translation of a local NET service name is required.

5.8.2 Oracle Client Installation related links

Oracle documentation: connection string formats

Oracle 10g Instant Client

6 Chapter 4. OpenLink ADO.NET Data Providers

Abstract
ADO.NET is the new feature-rich, data access Model used within the Microsoft.NET framework. It supports the data access requirements of the loosely coupled, multi-tier Web and Web services based applications of today. ADO.NET relies on the services of .NET Data Providers. These provide access to the underlying data source through four key objects: `Connection`, `Command`, `DataReader`, and `DataAdapter`.

The OpenLink .Net Data Providers are available as a suite of Managed and Unmanaged Data Providers providing a natural extension of our Universal Data Access Driver Suite. Thus providing .Net connectivity to the full range of Databases support in our Universal Data Access Driver suite. Two generic .Net Data Providers are available each exposing a common namespace/interface for accessing remote data sources as explained in the component architecture section below.

Table of Contents

6.1 Architecture

From the outset one of the main design goals of the .Net Data Provider was to implement a generic client layer enabling the re-use of our existing data access components for connecting to remote data sources. This has been achieved with both our Managed and Unmanaged .Net Data Providers, each of which have a common namespace that eliminates the need to recompile applications should the need arise to change the remote data source.

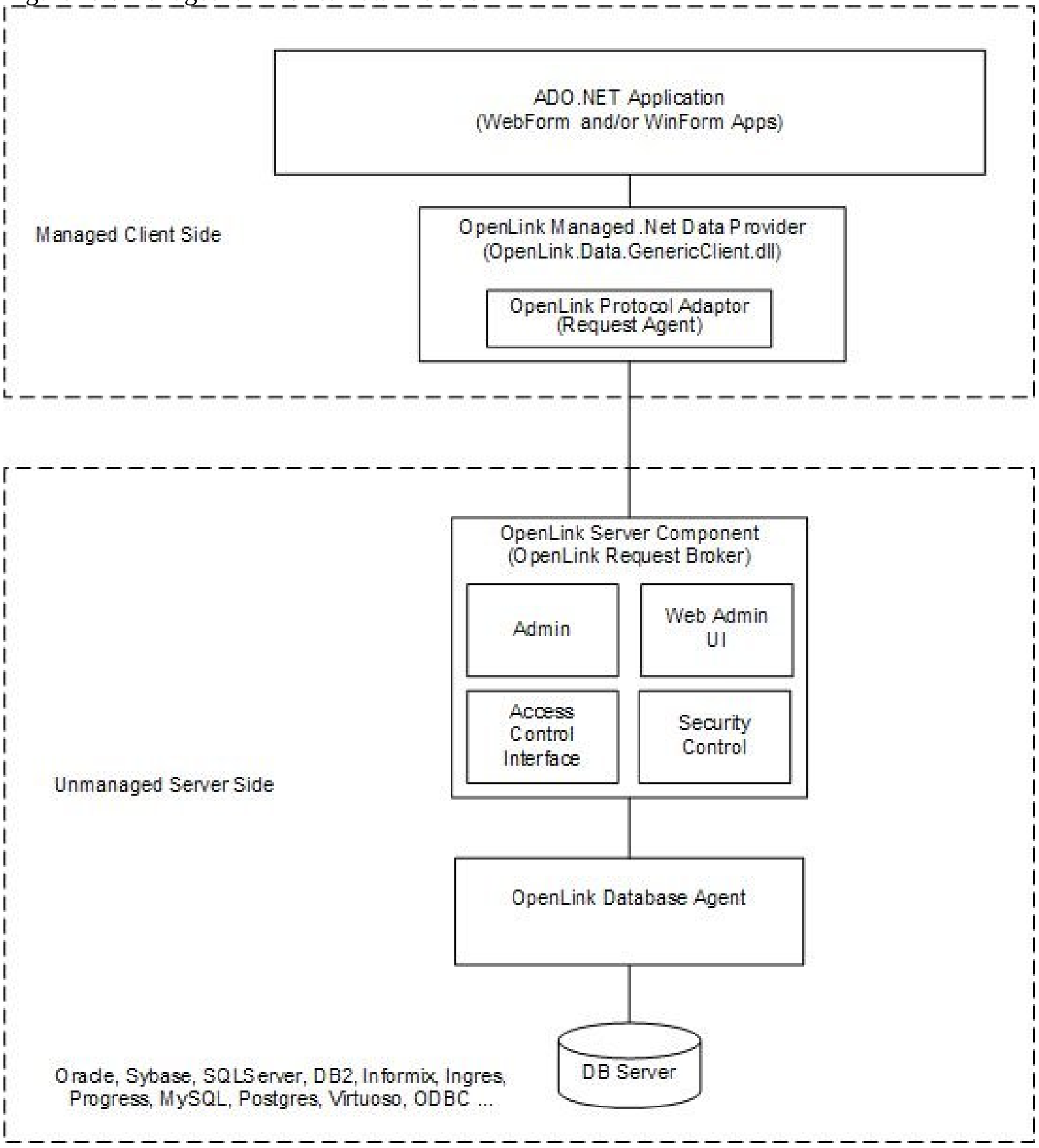
6.1.1 Managed Data Provider

The UDA managed .NET Data Provider delivers robust and secure data connectivity across all Databases support in the Universal Data Access suite, including all major databases - Microsoft SQL Server, Oracle, DB2, Sybase, Ingres II, Informix and Progress. The .NET Data Provider is built with managed code, enabling it to running completely within the

.NET Framework runtime delivering better security and performance. This Generic managed .Net Data Provider connects to the remote data source via one of two forms currently:

- *Multi-Tier Database Agents* - The OpenLink VDB(Virtual Database) layer has been ported to C# and linked into the Provider, enabling it to communicate directly with the OpenLink Multi-Tier Database agents installed on the remote machine (or via 3-Tier connection) as indicated in the diagram below:

  Figure 4.1. Managed VDB .Net Data Provider

  
  ADO.NET Application
  (WebForm and/or WinForm Apps)
  Managed Client Side
  OpenLink Managed .Net Data Provider
  (OpenLink.Data.GenericClient.dll)
  OpenLink Protocol Adaptor
  (Request Agent)
  OpenLink Server Component
  (OpenLink Request Broker)
  Admin
  Web Admin UI
  Access Control Interface
  Security Control
  Unmanaged Server Side
  OpenLink Database Agent
  Oracle, Sybase, SQLServer, DB2, Informix, Ingres, Progress, MySQL, Postgres, Virtuoso, ODBC ...
  DB Server

  The diagram above lists all the databases current supported in our Multi-Tier Data Access suite, but note that ODBC is also included which if chosen would enabled connectivity to any third party ODBC driver also via our ODBC agent(bridge).
- *TDS Protocol* - The TDS protocol has been ported to C# and linked into the Provider, enabling two 100% Managed Providers to be created capable of direct connectivity to Microsoft and Sybase SQLServer Databases without the need for any additional components on the Server, as indicated in the diagram below:

  Figure 4.2. Managed SQLServer .Net Data Provider

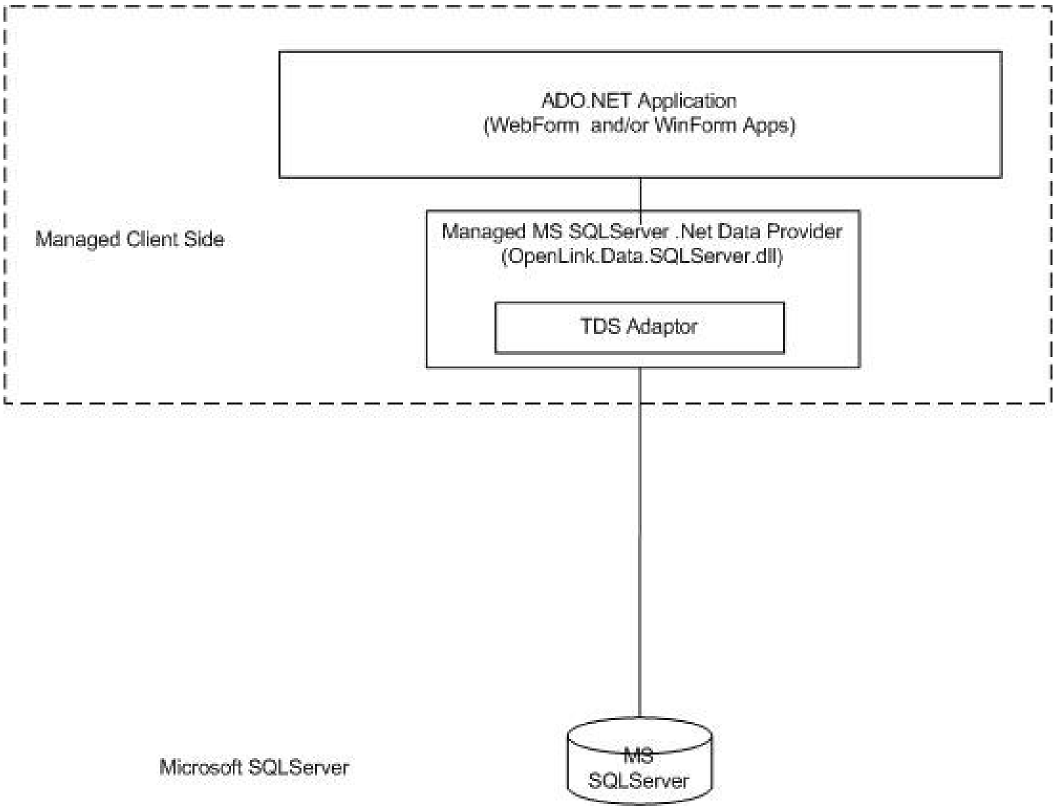
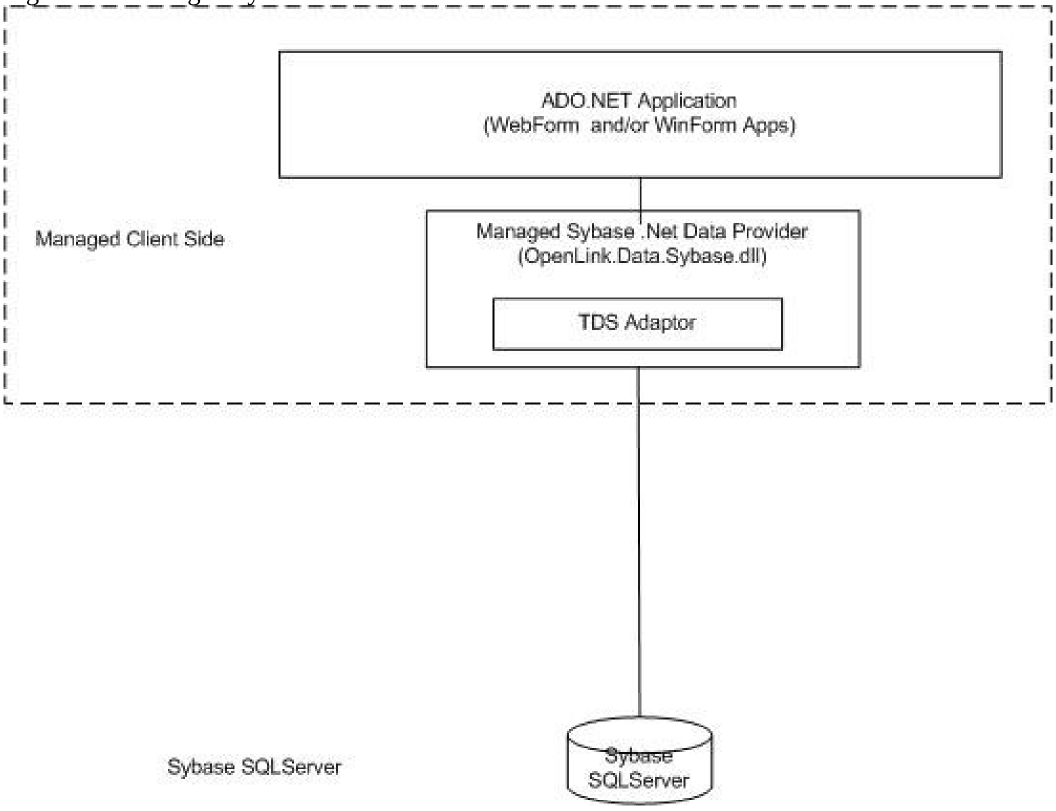

ADO.NET Application
(WebForm and/or WinForm Apps)
Managed Client Side
Managed MS SQLServer .Net Data Provider
(OpenLink.Data.SQLServer.dll)
TDS Adaptor
Microsoft SQLServer
MS SQLServer

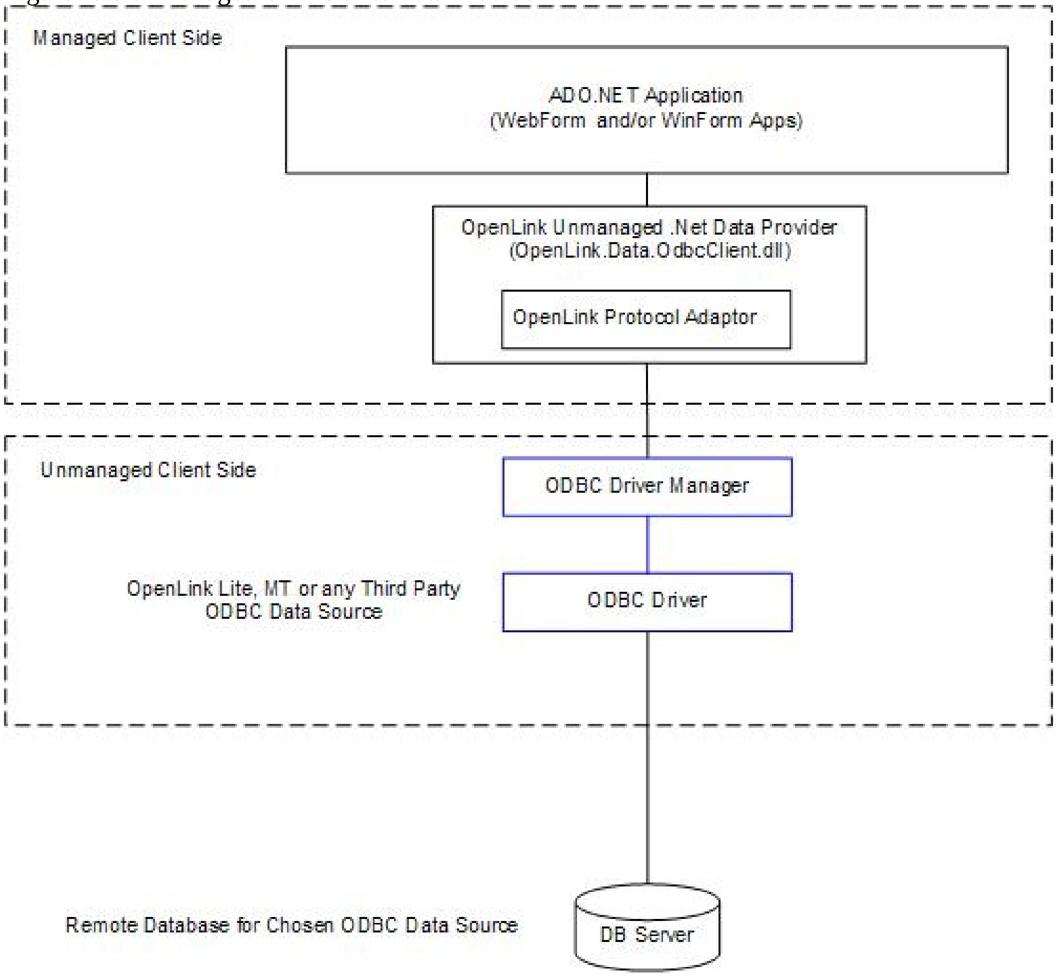
Figure 4.3. Managed Sybase .Net Data Provider


ADO.NET Application
(WebForm and/or WinForm Apps)
Managed Client Side
Managed Sybase .Net Data Provider
(OpenLink.Data.Sybase.dll)
TDS Adaptor
Sybase SQLServer
Sybase SQLServer

6.1.2 Unmanaged Data Provider

The UDA Unmanaged .NET Data Provider enables connectivity to any ODBC Data Source by acting as a Bridge between ADO.Net and ODBC. This Provider is provided as a stop-gap solution enabling connectivity to Data source for which managed .Net Providers are not already available as indicated in the diagram below, and as such does not provide the benefits of security and performance available from its Managed counterpart:

Figure 4.4. Unmanaged ODBC .Net Data Provider


Managed Client Side
ADO.NET Application
(WebForm and/or WinForm Apps)
OpenLink Unmanaged .Net Data Provider
(OpenLink.Data.OdbcClient.dll)
OpenLink Protocol Adaptor
Unmanaged Client Side
ODBC Driver Manager
OpenLink Lite, MT or any Third Party
ODBC Data Source
ODBC Driver
Remote Database for Chosen ODBC Data Source
DB Server

6.2 Developing .NET Data Provider Application

The namespaces for the OpenLink .Net Data Providers are:

```
OpenLink.Data.GenericClient     ; Managed VDB Generic Multi-Tier

OpenLink.Data.SQLServer         ; Managed Microsoft SQLServer

OpenLink.Data.Sybase            ; Managed Sybase SQLServer

OpenLink.Data.OdbcClient        ; Unmanaged ODBC based
```

A .NET data provider provides functionality for connecting to a data source, executing commands, and retrieving results. Those results can be processed directly, or placed in an ADO.NET DataSet for further processing while in a disconnected state. While in the DataSet, data can be exposed to the user, combined with other data from multiple sources, or passed remotely between tiers. Any processing performed on the data while in the DataSet can then be reconciled to the data source.

All .NET data providers are designed to be lightweight. They consist of a minimal layer between the data source and your code. This extends functionality without sacrificing performance.

There are four core objects that make up a .NET data provider. The following table describes those objects and their function.

Table 4.1. Core Classes

Object	Description
Connection	Establishes a connection to a specific data source and can begin a Transaction.
Command	Executes a command at a data source, and exposes Parameters.

Object	Description
DataReader	Exposes and reads a forward-only stream of data from a data source.
DataAdapter	Populates a DataSet and resolves updates with the data source.

Along with the core classes listed in the preceding table, a .NET data provider also contains the classes listed in the following table.

Table 4.2. Additional Classes

Object	Description
ClientPermission	Provided for .NET data provider code access security attributes.
CommandBuilder	A helper object that will automatically generate command properties of a DataAdapter or will derive parameter information from a stored procedure and populate the Parameters collection of a Command object.
Error	Exposes the information from a warning or error returned by a data source.
Exception	Returned when an error is encountered at the data source. For an error encountered at the client, .NET data providers throw a .NET Framework exception.
Parameter	Defines input, output, and return value parameters for commands and stored procedures.
Transaction	Enables you to enlist commands in transactions at the data source.

During the installation process the OpenLink .Net Data Provider is registered in the system GAC(Global Assembly Cache) enabling it for use by .Net applications. To use the OpenLink .NET Data Providers, the application must add an imports or using statement for the .Net Data Provider namespace, as the following code illustrates:

```
[Visual Basic]
Imports OpenLink.Data.GenericClient     // Managed VDB Generic Multi-Tier
or
Imports OpenLink.Data.SQLServer         // Managed Microsoft SQLServer
or
Imports OpenLink.Data.Sybase            // Managed Sybase SQLServer
or
Imports OpenLink.Data.OdbcClient        // Unmanaged ODBC based
[C#]
using OpenLink.Data.GenericClient;      // Managed VDB Generic Multi-Tier
or
using OpenLink.Data.SQLServer;          // Managed Microsoft SQLServer
or
using OpenLink.Data.Sybase;             // Managed Sybase SQLServer
or
using OpenLink.Data.OdbcClient;         // Unmanaged ODBC based
```

You must also include a reference to the .DLL when you compile your code. For example, if you are compiling a Microsoft® Visual C# program, your command line should include:

```
csc /r:OpenLink.Data.GenericClient.dll  ; Managed VDB Generic Multi-Tier
or
csc /r:OpenLink.Data.SQLServer.dll      ; Managed Microsoft SQLServer
or
csc /r:OpenLink.Data.Sybase.dll ; Managed Sybase SQLServer
or
csc /r:OpenLink.Data.OdbcClient.dll     ; Unmanaged ODBC based
```

6.3 OpenLink .NET Data Providers Connect String Attributes

This section details the Connect string attributes an ADO.Net application can pass to the OpenLink .Net Data Providers when attempting a connection to a remote database.

6.3.1 Managed Generic Multi-Tier .NET Data Provider (OpenLink.Data.GenericClient)

The tables below provides a list of the Connect String Attributes for the Data Provider:

Table 4.3. Connection String Keywords for the Generic .Net Provider

Keyword	Default Value
Used ID or UID	[""]
Password or PWD	[""]
Initial Catalog or DATABASE	[""]
Connect Timeout or Connection Timeout	[15]
Persist Security Info	[False]
Connection Lifetime	[0]
Min Pool Size	[0]
Max Pool Size	[100]
Persist Security Info	[False]
Pooling	[True]
Charset	["]
Host	[localhost]
Port	[5000]
Read Only or ReadOnly	[False]
Fetch Buffer Size or FBS or FetchBufferSize	[100]
Server Type or ServerType or SVT	[""]
Defer Long Fetch or DLF or DeferLongFetch	[False]
Encrypted	[False]
Options	[""]
Defer Long Fetch or DLF or DeferLongFetch	[False]
UNICODE	[False]
APPLICATION	["opldotNET"]

Sample connect string :

```
For Oracle 9 :
        "Host=localhost;Port=5000;SVT=Oracle 9.x;DATABASE=ORCL;UID=scott;PWD=tiger;
For SQLServer2000:
        "Host=localhost;Port=5000;SVT=SQLServer 2000;DATABASE=Northwind;UID=sa;PWD=;
```

6.3.2 Managed Microsoft SQLServer .NET Data Provider (OpenLink.Data.SQLServer)

The tables below provides a list of the Connect String Attributes for the Data Provider:

Table 4.4. Connection String Keywords for the Microsoft SQLServer .Net Provider

Keyword	Default Value
Used ID or UID	[""]
Password or PWD	[""]
Initial Catalog or DATABASE	[""]
Connect Timeout or Connection Timeout	[15]
Persist Security Info	[False]
Connection Lifetime	[0]
Min Pool Size	[0]
Max Pool Size	[100]
Persist Security Info	[False]
Pooling	[True]
Charset	["]
Host	[localhost]
Port	[1433]

Keyword	Default Value
Packet Size or PktSize	[8192] for SQLServer7/2000
SrvName or Server or Server Na	[""]
Server Type or ServerType or SVT	["SQLSERVER6"]
WorkstationId or Workstation Id	["localhost]
UNICODE	[False]
APPLICATION	["opldotNET"]
UseBrowseMode	["false"]

The following Keyword default values change for the SQLServer Provider :

```
UseBrowseMode=false - the SQLParser is used for parsing Select queries and getting the acc
for following schema table columns (returned by IDataReader.GetSchemaTable).
UseBrowseMode=true -  the "SET no_browsetable ON" comand is used with Select queries for g
and for accurate information for following schema table columns (returned by IDataReader.G
 Note that the MS SQLServer ADO.NET provider uses the "SET no_browsetable ON" mode.
The property SVT or ServerType supports the following values:
"SQLServer 7" or "SQLServer7" (TDS7 protocol is used)
"SQLServer 2000" or "SQLServer2000" (TDS8 protocol is used)
"SQLServer 6"  or "SQLServer 6.x"  or "SQLServer"  or "SQLServer6"(TDS4.2 protocol is used
Any another values (TDS4.2 protocol is used)
Sample connection string:
        "SVT=SQLServer 2000;DATABASE=Northwind;Host=localhost;Port=1433UID=sa;PWD=;
```

6.3.3 Managed Sybase .NET Data Provider (OpenLink.Data.Sybase)

The tables below provides a list of the Connect String Attributes for the Data Provider:

Table 4.5. Connection String Keywords for the Sybase .Net Provider

Keyword	Default Value
Used ID or UID	[""]
Password or PWD	[""]
Initial Catalog or DATABASE	[""]
Connect Timeout or Connection Timeout	[15]
Persist Security Info	[False]
Connection Lifetime	[0]
Min Pool Size	[0]
Max Pool Size	[100]
Persist Security Info	[False]
Pooling	[True]
Charset	["]
Host	[localhost]
Port	[4100]
Packet Size or PktSize	[512]
SrvName or Server or Server Na	[""]
Server Type or ServerType or SVT	["SYBASE4"]
WorkstationId or Workstation Id	["localhost]
UNICODE	[False]
APPLICATION	["opldotNET"]

The following Keyword default values change for the SQLServer Provider :

```
The property SVT or ServerType supports the following values:
        "Sybase 10"  or "Sybase10" (TDS5 protocol is used)
```

```
                "Sybase 11"  or "Sybase11" (TDS5 protocol is used)
                "Sybase 12"  or "Sybase12"  or "Sybase12.x" (TDS5 protocol is used)
        Any another values (TDS4.2 protocol is used)
        Sample connection string:
                "SVT=Sybase 12;DATABASE=pubs2;Host=localhost;Port=4100;UID=sa;PWD=;
```

6.3.4 Unmanaged .NET Data Provider (OpenLink.Data.OdbcClient)

The tables below provides a list of the Connect String Attributes for the Unmanaged (ODBC based) Data Provider:

Table 4.6. Connection String Keywords

Keyword	Default Value
DSN	[""]
UID	[""]
PWD	[""]

```
   Sample connect string:
        - Oracle ODBC DSN
           "DSN=Oracle;UID=scott;PWD=tiger"
        where 'Oracle' is the name of a valid ODBC DSN for connecting to an Oracle Database
```

6.4 OpenLink .NET Data Providers Class Implementation

This section details the classes exposed by the OpenLink .NET Data Providers, which users must be familiar with in order to make effective use of the providers.

Many of the OpenLink Generic Provider classes implement interfaces or inherit from the .NET Framework namespaces System.Data and System.Data.Common. A summary of the classes exposed by the provider is given below.

The table only shows classes or interfaces inherited from System.Data or System.Data.Common. Some of the OpenLink.Data.GenericClient classes implement additional interfaces or inherit from a class belonging to a namespace other than System.Data or System.Data.Common. These details are not shown. For full details refer to the detailed documentation for the appropriate class in OpenLink.Data.GenericClient Namespace.

Table 4.7. Classes

Classes	Implements / inherits System.Data or System.Data.Common interface / class	Description
OPLCommand	IDbCommand	Represents an SQL statement or stored procedure to execute against a database.
OPLCommandBuilder		Automatically generates single-table commands used to reconcile changes made to a DataSet with the associated data source.
OPLConnection	IDbConnection	Represents an open connection to a data source.
OPLDataAdapter	IDbDataAdapter, DbDataAdapter	Represents a set of data commands and a database connection that are used to fill a DataSet and update a data source.
OPLDataReader	IDataReader, IDataRecord	Provides a way of reading a forward-only stream of data rows from a data source.
OPLError		Holds information about a warning or error returned by a data source.
OPLErrorCollection		Represents a collection of one or more OPLError objects that give detailed information about an OPLException.
OPLErrorException		The exception that is generated when a warning or error is returned by a data source or the Generic Provider.

Classes	Implements / inherits System.Data or System.Data.Common interface / class	Description
OPLParameter	IDbDataParameter, IDataParameter	Represents a parameter to an OPLCommand and optionally, its mapping to a DataSet column.
OPLParameterCollection	IDataParameterCollection	Represents a collection of parameters relevant to an OPLCommand as well as their respective mappings to columns in a DataSet.
OPLPermission		Enables the Generic Provider to ensure that a user has a security level adequate to access a data source.
OPLPermissionAttribute		Associates a security action with a custom security attribute.
OPLTransaction	IDbTransaction	Represents a SQL transaction to be made at a data source.

Table 4.8. Delegates

Delegates	Description
OPLRowUpdatedEventHandler>	Represents a method that will handle the RowUpdated event of an OPLDataAdapter.
OPLRowUpdatingEventHandler	Represents a method that will handle the RowUpdating event of an OPLDataAdapter.

Table 4.9. Enumerations

Enumeration	Description
OPLDbType >	Specifies the data type of a field, property or OPLParameter.

6.5 OpenLink.Data Namespace

All the public classes in the OpenLink.Data namespace are detailed in this section. Note that all of the OpenLink .Net Data Providers use the same common class names, thus the descriptions below are applicable to all enabling maximum re-use of code across Providers.

IMPORTANT: Public class members inherited from a base class defined by the .NET Framework Class Library (FCL), or which implement an FCL defined interface method, are not described. For details of these inherited members, refer to the .NET FCL documentation for the base class or interface. Where an OpenLink class implements an ADO.NET interface and adds additional methods or properties, these extensions are documented.

In the class descriptions which follow, all code extracts showing partial class definitions, method signatures etc. are in C#. Examples for other .NET languages are not shown.

6.5.1 OPLCommand

Represents an SQL statement or stored procedure to execute against a database.

```
public class OPLCommand : Component, IDbCommand, ICloneable
```

6.5.1.1 Constructors

```
public OPLCommand ();
```

Initializes a new instance of the OPLCommand class.

```
public OPLCommand (string cmdText);
```

Initializes a new instance of the OPLCommand class with the text of the query.

```
public OPLCommand (string cmdText, OPLConnection connection);
```

Initializes a new instance of the OPLCommand class with the text of the query and an OPLConnection object.

```
public OPLCommand (string cmdText, OPLConnection connection,OPLTransaction txn);
```

Initializes a new instance of the OPLCommand class with the text of the query, an OPLConnection object and an OPLTransaction object.

6.5.1.2 Properties

6.5.1.2.1 PassThrough

```
public bool PassThrough {get; set;}
```

Allows the command text to be passed to the data source without being parsed by the Generic Provider.

6.5.1.2.2 Remarks

The PassThrough property defaults to false. When pass-through is disabled, the Generic Provider parses command text associated with the OPLCommand and queries the data source for additional metadata on tables referenced in a select statement. The command parser in the Generic Provider is limited to the SQL grammar defined by ODBC. If you wish to pass native SQL to the data source you should enable pass-through; however this will have the following side-effects:

Some of the metadata returned by OPLDataReader.GetSchemaTable may not be accurate, specifically the metadata resultset columns IsKeyColumn, BaseSchemaName, BaseCatalogName, BaseTableName.

The command behavior setting CommandBehavior.KeyInfo will not append missing key columns to the end of a select list.

OPLCommandBuilder may not work, depending on the select statement used.

6.5.2 OPLCommandBuilder

Automatically generates single-table commands used to reconcile changes made to a DataSet with the associated data source.

```
public sealed class OPLCommandBuilder : Component
```

6.5.2.1 Constructors

```
public OPLCommandBuilder ();
```

Initializes a new instance of the OPLCommandBuilder class.

```
public OPLCommandBuilder (OPLDataAdapter adapter);
```

Initializes a new instance of the OPLCommandBuilder class with the associated OPLDataAdapter object.

6.5.2.2 Methods

```
public static void DeriveParameters (OPLCommand command);
```

Retrieves parameter information from the stored procedure specified in the OPLCommand and populates the Parameters collection of the specified OPLCommand object.

```
protected override void Dispose (bool disposing);
```

Releases the resources used by the OPLCommandBuilder.

```
public OPLCommand GetDeleteCommand ();
```

Gets the automatically generated OPLCommand object required to perform deletions at the data source.

```
public OPLCommand GetInsertCommand ();
```

Gets the automatically generated OPLCommand object required to perform insertions at the data source.

```
public OPLCommand GetUpdateCommand ();
```

Gets the automatically generated OPLCommand object required to perform updates at the data source.

```
public void RefreshSchema ();
```

Refreshes the database schema information used to generate INSERT, UPDATE, or DELETE statements.

6.5.2.3 Properties

```
public OPLDataAdapter DataAdapter {get; set;}
```

Gets or sets an OPLDataAdapter object for which this OPLCommandBuilder object will generate SQL statements.

```
public string QuotePrefix {get; set;}
```

Gets or sets the beginning character or characters to use when working with database objects (for example, tables or columns) whose names contain characters such as spaces or reserved tokens.

```
public string QuoteSuffix {get; set;}
```

Gets or sets the ending character or characters to use when working with database objects, (for example, tables or columns), whose names contain characters such as spaces or reserved tokens.

6.5.3 OPLConnect

Represents an open connection to a data source.

```
public sealed class OPLConnection : Component, ICloneable, IDbConnection
```

6.5.3.1 Constructors

```
public OPLConnection();
```

Initializes a new instance of the OPLConnect class.

```
public OPLConnection(string connectionString);
```

Initializes a new instance of the OPLConnection class with the specified connection string.

6.5.3.2 Methods

```
protected override void Dispose (bool disposing);
```

Releases the resources used by the OPLConnection.

6.5.3.3 Events

```
public event StateChangeEventHandler StateChange;
```

Occurs when the state of the connection changes.

6.5.3.3.1 Remarks

The StateChange event is raised immediately after the State changes from Closed to Opened, or from Opened to Closed.

```
public event OPLInfoMessageEventHandler InfoMessage;
```

Returns warning messages from the DBMS.

```
public delegate void OPLInfoMessageEventHandler (object sender, OPLInfoMessageEventArgs e)

class OPLInfoMessageEventArgs;
```

6.5.4 OPLDataAdapter

Represents a set of data commands and a database connection that are used to fill a DataSet and update a data source.

6.5.4.1 Constructors

```
public OPLDataAdapter();
```

Initializes a new instance of the OPLDataAdapter class.

```
public OPLDataAdapter (OPLCommand selectCommand);
```

Initializes a new instance of the OPLDataAdapter class with the specified SQL SELECT statement.

```
public OPLDataAdapter (string selectCommandText, OPLConnection selectConnection);
```

Initializes a new instance of the OPLDataAdapter class with an SQL SELECT statement and an OPLConnection.

```
public OPLDataAdapter (string selectCommandText, string selectConnectionString);
```

Initializes a new instance of the OPLDataAdapter class with an SQL SELECT statement and a connection string.

6.5.4.2 Methods

```
public event OPLRowUpdatedEventHandler RowUpdated;
```

Occurs during an Update operation after a command is executed against the data source.

```
public event OPLRowUpdatingEventHandler RowUpdating;
```

Occurs during an Update operation before a command is executed against the data source.

6.5.5 OPLDataReader

Provides a way of reading a forward-only stream of data rows from a data source.

```
public sealed class OPLDataReader : MarshalByRefObject, IDataReader, IDataRecord, IDisposable, IEnumerabl
```

6.5.6 OPLError

Holds information about a warning or error returned by a data source.

```
public sealed class OPLError
```

6.5.6.1 Remarks

This class is created whenever an error occurs. Each instance of OPLError created is then managed by the OPLErrorCollection class, which in turn is created by the OPLException class.

6.5.6.2 Properties

```
public string Message {get;}
```

Gets a short description of the error.

```
public string SQLState {get;}
```

Gets the five character ODBC SQL state associated with the error.

```
public int NativeError {get;}
```

Gets the data source-specific error information.

6.5.7 OPLErrorCollection

Represents a collection of one or more OPLError objects that give detailed information about an OPLException.

```
public sealed class OPLErrorCollection : ICollection
```

6.5.7.1 Remarks

This class is created by OPLException to collect instances of the OPLError class. OPLErrorCollection always contains at least one instance of the OPLError class.

6.5.7.2 Methods

```
public IEnumerator GetEnumerator();
```

This member supports the .NET Framework infrastructure and is not intended to be used directly from your code.

6.5.7.3 Properties

```
public OPLError this[int i] {get;}
```

Gets the error at the specified index. In C#, this property is the indexer for the OPLErrorCollection class.

6.5.8 OPLException

The exception that is generated when a warning or error is returned by a data source or the Generic Provider.

```
public sealed class OPLException : SystemException
```

6.5.8.1 Properties

```
public OPLErrorCollection Errors {get;}
```

Gets a collection of one or more OPLError objects that give detailed information about the exception.

```
public override string Message {get;}
```

Gets the text describing the error.

6.5.9 OPLParameter

Represents a parameter to an OPLCommand and optionally, its mapping to a DataSet column.

```
public class OPLParameter : MarshalByRefObject, IDbDataParameter, IDataParameter, ICloneable
```

6.5.9.1 Constructors

```
public OPLParameter();
```

Initializes a new instance of the OPLParameter class.

```
public OPLParameter(string parameterName, object value);
```

Initializes a new instance of the OPLParameter class with the parameter name and a value of the new OPLParameter object.

```
public OPLParameter(string parameterName, OPLDbType type);
```

Initializes a new instance of the OPLParameter class with the parameter name and the data type.

```
public OPLParameter(string parameterName, OPLDbType type, int size);
```

Initializes a new instance of the OPLParameter class with the parameter name, the OPLDbType and the size.

```
public OPLParameter(string parameterName, OPLDbType type, int size, string sourceColumn);
```

Initializes a new instance of the OPLParameter class with the parameter name, the OPLDbType, the size, and the source column name.

```
public OPLParameter(string parameterName, OPLDbType type, int size, ParameterDirection direction, Boolean
```

Initializes a new instance of the OPLParameter class with the parameter name, the type of the parameter, the size of the parameter, a ParameterDirection, whether the parameter is nullable, the precision of the parameter, the scale of the parameter, the source column, a DataRowVersion to use, and the value of the parameter.

6.5.9.2 Methods

```
public override string ToString ();
```

Gets a string containing the ParameterName.

6.5.9.3 Properties

```
public OPLDbType OPLDbType {get; set;}
```

Gets or sets the OPLDbType of the parameter.

6.5.9.3.1 Remarks

The OPLDbType and DbType are linked. Therefore, setting the DbType changes the OPLType to a supporting OPLDbType. For a list of the supported data types, see the appropriate OPLDbType member.

6.5.10 OPLParameterCollection

Represents a collection of parameters relevant to an OPLCommand as well as their respective mappings to columns in a DataSet.

```
public class OPLParameterCollection : MarshalByRefObject, IDataParameterCollection, ICollection, IEnumera
```

6.5.10.1 Methods

```
public int Add (OPLParameter value);
```

Adds the specified OPLParameter to the OPLParameterCollection

```
public int Add (string parameterName, object value);
```

Adds an OPLParameter to the OPLParameterCollection given the parameter name and value.

```
public int Add (string parameterName, OPLDbType dbType);
```

Adds an OPLParameter to the OPLParameterCollection given the parameter name and data type.

```
public int Add (string parameterName, OPLDbType dbType, int size);
```

Adds an OPLParameter to the OPLParameterCollection given the the parameter name, data type, and column length.

```
public int Add (string parameterName, OPLDbType dbType, int size, string sourceColumn);
```

Adds an OPLParameter to the OPLParameterCollection given the parameter name, data type, column length, and source column name.

6.5.10.2 Properties

```
public OPLParameter this[int index] {get; set;}
```

Gets or sets the OPLParameter at the specified index

```
public OPLParameter this[string parameterName] {get; set;}
```

Gets or sets the OPLParameter with the specified name.

6.5.11 OPLRowUpdatedEventArgs

Provides data for the OPLDataAdapter.RowUpdated event.

6.5.11.1 Constructors

```
public OPLRowUpdatedEventArgs(DataRow row, IDbCommand command, StatementType statementType, DataTa
```

Initializes a new instance of the OPLRowUpdatedEventArgs class.

6.5.11.2 properties

```
public new OPLCommand Command {get;}
```

Gets the OPLCommand executed when Update is called.

6.5.12 OPLRowUpdatedEventHandler

Represents a method that will handle the RowUpdated event of an OPLDataAdapter.

```
public delegate void OPLRowUpdatedEventHandler(object sender, OPLRowUpdatedEventArgs e);
```

6.5.13 OPLRowUpdatingEventArgs

Provides data for the OPLDataAdapter.RowUpdating event.

6.5.13.1 Constructors

```
public OPLRowUpdatingEventArgs(DataRow row, IDbCommand command, StatementType statementType, DataT
```

Initializes a new instance of the OPLRowUpdatingEventArgs class.

6.5.13.2 properties

```
public new OPLCommand Command {get; set;}
```

Gets or sets the OPLCommand to execute when Update is called.

6.5.14 OPLRowUpdatingEventHandler

Represents a method that will handle the RowUpdating event of an OPLDataAdapter.

```
public delegate void OPLRowUpdatingEventHandler(object sender, OPLRowUpdatingEventArgs e);
```

6.5.15 OPLTransaction

Represents an SQL transaction to be made at a data source.

```
public sealed class OPLTransaction : MarshalByRefObject, IDbTransaction, IDisposable
```

6.6 Known Issues

6.6.1 Unsigned Type Handling

6.6.1.1 Unmanaged ODBC Provider

The provider assumes that all integer data returned by an ODBC driver is signed. The provider only examines the ODBC SQL type of the data to ascertain its type. But, the ODBC SQL type gives no indication of whether the data is signed or unsigned. The ODBC to .NET type mappings used internally for fetched data assume that all data is signed. There is the potential for overflow errors when fetching unsigned integer data, because the .NET types to which integer data is mapped are all signed. However, across the range of databases currently supported by OpenLink, this problem only manifests itself in one instance with the SQL Server TinyInt type. This is the only native unsigned type supported across the databases covered by OpenLink. However, if the ODBC provider is to support third party ODBC drivers, more databases could expose this fault.

The SQL Server TinyInt type has a range of 0 .. 255. It is returned by the provider as a .NET System.SByte type which has a range of 127 .. 128. At present, TinyInt values over 128 will not be returned correctly by the provider.

6.6.1.2 Managed Providers

The same general problem exists in the GenericClient managed provider. However, the provider handles SQL_TINYINT data as a special case, so the problem with the SQL Server TinyInt type does not arise. If the provider detects data fetched using the OpenLink CTYPE_UNS8 transport type, it changes the mapping for SQL_TINYINT from System.SByte to System.Byte. Any other type of unsigned integer data will not be handled correctly, but at present this does not arise as TinyInt is the only unsigned native DBMS type the provider encounters across our currently supported agents.

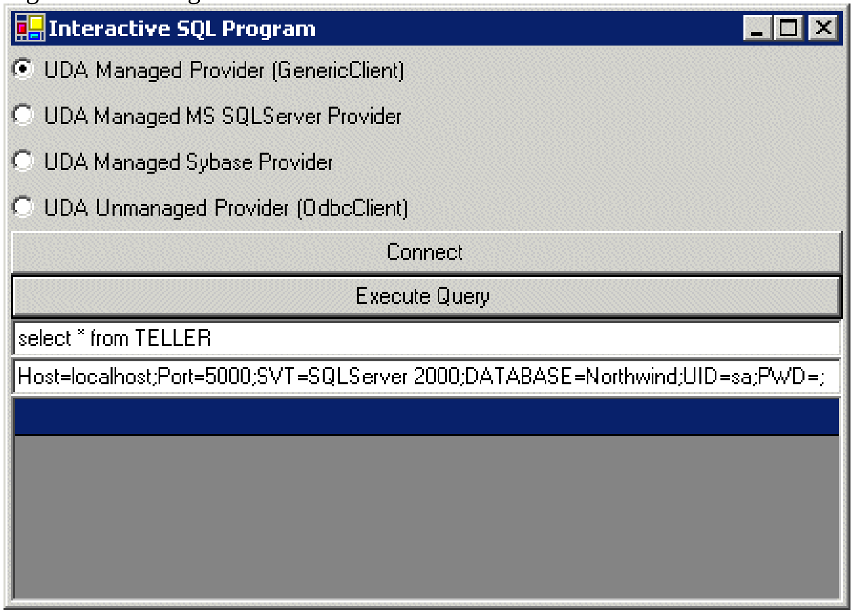
6.6.2 Timestamp Precision

Both the managed and unmanaged providers map the ODBC SQL_TYPE_TIMESTAMP type to the .NET DateTime type. The .NET DateTime type stores times with millisecond precision, whereas an ODBC SQL_C_TIMESTAMP struct can hold seconds accurate to nanoseconds. A loss of precision will result when fetching timestamp data from databases which hold fractional second data to greater than millisecond precision. The corresponding DateTime values returned to an application will be rounded to the nearest millisecond.

6.7 .Net Provider Test Program

A simple .Net interactive SQL program is provided for enabling a test connection to be made with the UDA Managed and/or Unmanaged .Net Data Providers, as indicated in the screen shots below:

Figure 4.5. Managed .Net Data Provider test connection



Interactive SQL Program

● UDA Managed Provider (GenericClient)
○ UDA Managed MS SQLServer Provider
○ UDA Managed Sybase Provider
○ UDA Unmanaged Provider (OdbcClient)

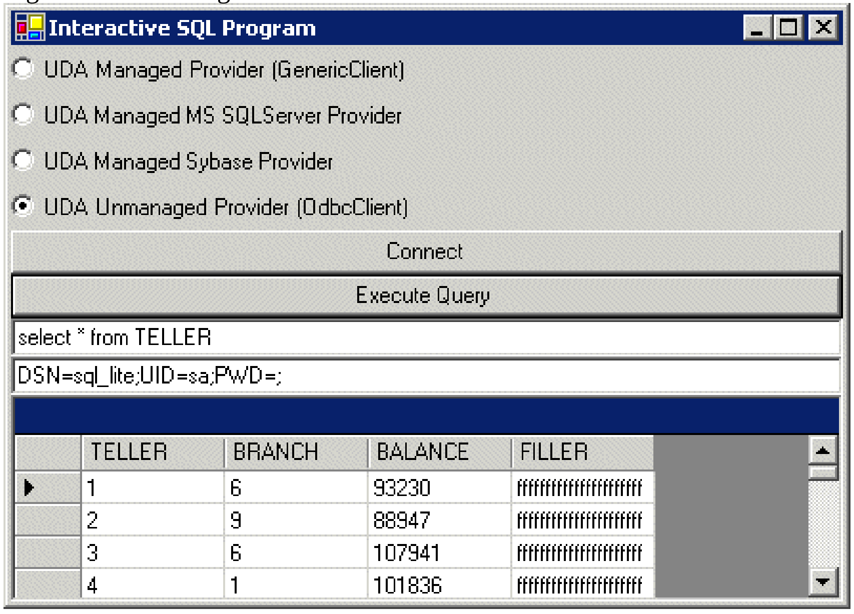
Connect

Execute Query

```
select * from TELLER
Host=localhost;Port=5000;SVT=SQLServer 2000;DATABASE=Northwind;UID=sa;PWD=;
```

The connect strings for the Managed .Net Providers (GenericClient, MS SQLServer and Sybase) are similar with the 'Port' attribute designating the location of the Service to which is connection will band made. The Unmanaged OdbcClient provider differs as indicated below:

Figure 4.6. Unmanaged .Net Data Provider test connection


Interactive SQL Program
UDA Managed Provider (GenericClient)
UDA Managed MS SQLServer Provider
UDA Managed Sybase Provider
UDA Unmanaged Provider (OdbcClient)
Connect
Execute Query

( ) UDA Managed Provider (GenericClient)
( ) UDA Managed MS SQLServer Provider
( ) UDA Managed Sybase Provider
[✓ UDA Unmanaged Provider (OdbcClient)]

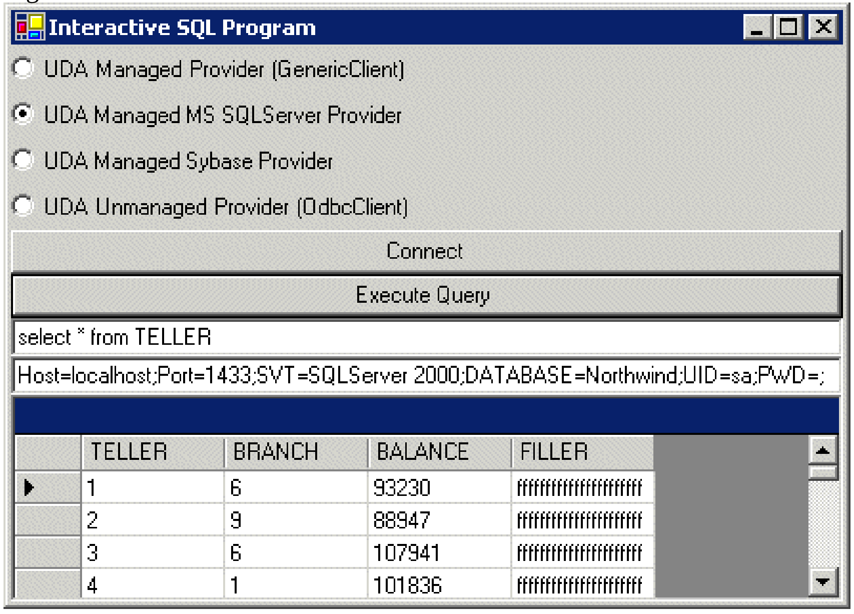
select * from TELLER

DSN=sql_lite;UID=sa;PWD=;

TELLER	BRANCH	BALANCE	FILLER
1	6	93230	ffffffffffffffffffff
2	9	88947	ffffffffffffffffffff
3	6	107941	ffffffffffffffffffff
4	1	101836	ffffffffffffffffffff

1. Select the *radio* button of the .Net Provider you want to test.
2. Enter the *connect string attributes* of the Database Server to which the connection is to be made.
3. Click the *connect* button to make the Database connection.
4. Enter the *SQL query* you wish to run in the text window provided and click the *Execute Query* button to execute.
5. Once executed the result set will be displayed in the result set pane as indicated in screen shot below:

Figure 4.7.


Interactive SQL Program
UDA Managed Provider (GenericClient)
UDA Managed MS SQLServer Provider
UDA Managed Sybase Provider
UDA Unmanaged Provider (OdbcClient)
Connect
Execute Query

( ) UDA Managed Provider (GenericClient)
[✓ UDA Managed MS SQLServer Provider]
( ) UDA Managed Sybase Provider
( ) UDA Unmanaged Provider (OdbcClient)

select * from TELLER

Host=localhost;Port=1433;SVT=SQLServer 2000;DATABASE=Northwind;UID=sa;PWD=;

TELLER	BRANCH	BALANCE	FILLER
1	6	93230	ffffffffffffffffffff
2	9	88947	ffffffffffffffffffff
3	6	107941	ffffffffffffffffffff
4	1	101836	ffffffffffffffffffff

6.8 New Features

- .Net Data Providers. A suite of Managed .Net Data Providers built with a thin generic .Net client classes for communicating with any of our supported Database agents on the Database Server, as well as a 100% Managed TDS based .Net Provider for connectivity to Microsoft and Sybase SQLServer Database eliminating the need for any software installation on the Database Server machine. An Umanaged .Net ODBC Provider is also available, which makes use of the OpenLink built in Cursor Library to provide a greater level of .Net support when connecting via an ODBC Datasource.

- Unicode Driver support. Database agents support the latest releases of all supported database engines this includes:

    1. Oracle 8i & 9i
    2. Microsoft SQL Server 2000
    3. DB/2 v7
    4. ODBC Agent
    5. JDBC Agent
    6. Informix 9
    7. Sybase Adaptive Server 12.5
    8. Progress 9 (SQL-92)
- Significant SQL Server and SYBASE Driver Enhancements. Our Drivers are now built to communicate directly with Microsoft SQL Server and SYBASE ASE using the TDS protocol (the native wire protocol for both database servers). This also implies that no additional software is required post installation in order for our ODBC Drivers to communicate with these Drivers (this applies to the Single Tier format Drivers only). The use of TDS has also enabled us to double the performance of both our Single Tier and Multi-Tier Drivers for these databases.
- Array Optimisations on Select Queries. All drivers now support the SQLSetStmtAttr(SQL_ARRAY_SIZE) call for batch select statements, providing improved performance when re-execute select statements with bound paramters.
- Deferred Fetching. The Release 4 OpenLink driver family brings marked communications layer improvements. Central to these are improved implementation of the `SQLGetData` and `SQLPutData` ODBC function calls.

    Wholesale vs. Piecemeal Data Transfer. In previous releases of the drivers, parameter data at query execution was assembled in the OpenLink driver's client component and transferred to its server component in a single network hop. (The client and server components apply to both the Single- and Multi-Tier drivers; they refer to different layers within the driver entity.) Similarly, when fetching from a "long", or large binary data column, data was transferred from the driver's server component to the client component in a single network transfer. The only way data could manipulated in a piecemeal fashion, was within the ODBC application from the driver's client component (client side only).

    The Release 4 driver family now allow transferring parameter data in parts over the network between the client and server driver components. Once transferred, the fragmented column data are re-pieced together in their entirety within the client and server portions of the driver.

    Deferred Fetching. When fetching, data from columns with "long" data are only transferred between the server and client components if one of the following applies to that column:

    It has been "bound" by the application via the SQLBindCol API call

    It has been retrieved via the SQLGetData API call

    This mechanism is referred to as deferred fetching. In this method, as "long" column data is not reassembled within the OpenLink ODBC client itself (rather, within the application), driver memory overhead incurred is dramatically reduced. Deferred fetching applies to the following "long" database column types:

    ODBC agent

        SQL_LONGVARCHAR

        SQL_LONGVARBINARY

    DB/2 agent

        SQL_BLOB

        SQL_CLOB

        SQL_DBCLOB

        If the long data compatibility option has been specified in the DB2 database

        SQL_LONGVARCHAR

SQL_LONGVARBINARY

SQL_LONGVARGRAPHIC

Oracle agent

SQLT_BLOB

SQLT_CLOB

Sybase agent

CS_IMAGE_TYPE

CS_TEXT_TYPE

However, if a table contains a column defined as one of these "long" types, but the actual data stored in the column only fills a small proportion of the available space, deferred fetching is of no benefit. In these cases, performance may be improved by switching off the deferred fetching mechanism using the control in the OpenLink Generic Client data source setup dialog.

7 Chapter 5. OpenLink OLE-DB Provider

Abstract
The OpenLink ODBC Provider for OLEDB is delivered as an OLEDB - ODBC bridge. It utilizes ODBC data sources to connect to databases, much like the Microsoft ODBC Provider; so you will require a working ODBC installation first.

The OpenLink Provider also requires that the Microsoft Data Access Components have been installed. To gain any functionality with .Net you will require MDAC 2.7 RTM (2.70.7713.4) or later. This particular version of the MDAC is included with Visual Studio .Net and with the .Net Framework SDK. Alternatively it can be downloaded from www.microsoft.com/data. Releases prior to 2.70.7713.4, specifically MDAC 2.7 included in the .Net SDK Beta 2, will not allow non-Microsoft OLE DB providers to work with the .Net Data Provider for OLE DB.

Although not completely necessary if you are using a newer OpenLink installer, before installing this provider, any previous release of the driver should be deinstalled fully. If a full deinstall using Install Shield is not performed, the old driver should be unregistered using the regsvr32 utility as follows:

```
regsvr32 /u oploleod.dll
```

Table of Contents

7.1 Testing an OLEDB Connection

The connection can be tested using the sample application, "Rowset Viewer" as follows:

1. Select *Full Connect* from the File menu
2. Select *OpenLink ODBC Provider* for the connection provider
3. Choose an ODBC *Datasource* either by typing the name directly or using the *"..."* button and selecting *Enumerator* from the submenu that appears in place of the button. In the new dialog that appears choose " *OpenLink ODBC Data Source Enumerator* " as the data source enumerator and click *Connect* . A list of ODBC data sources will appear, from which you can choose the one you want and click *OK* to confirm.
4. Enter the *UserID* and *Password* and click on *OK* to connect.
5. Once connected an "Command" subwindow will appears, into which you can submit SQL for test querying on the data source.

7.2 Call Tracing

To turn on call tracing and enable logging to a file, use the Registry Editor (regedit or regedt32) to enter a fully qualified file name in the registry entry:

```
        HKEY_LOCAL_MACHINE\SOFTWARE\OpenLink Software\OpenLink OLE DB Provider\DebugFile
```

To turn off call tracing simply leave this entry blank. A separate log file is opened for each process which uses the provider. Each file opened is named using the file base name specified in the DebugFile entry above with a three digit process ID suffix.

7.3 Provider Specific Connection Information

OLE DB predefines a number of property sets and the properties in them. The *Extended Properties* property in the Initialization property group provides a means of supplying provider-specific extended connection information. The OpenLink provider supports extended properties to control:

ODBC Cursor Library usage

whether bookmarks are enabled by default on rowsets

There are three ways to enter extended properties:

Through a prompting interface supplied by the consumer application.

Initialise a Data Source object through a connection string.

Explicitly set the DBPROP_INIT_PROVIDERSTRING property in the consumer application code.

7.3.1 Using a Consumer Supplied Prompting Interface

The OpenLink provider does not display a prompting interface of its own for entering OLE DB specific connection information. It relies instead upon the application supplying it.

OLE DB provides several core components, one of which is the Data Link component. The Data Link component provides a default prompting interface and is used by numerous Microsoft applications. It typically includes a page displayed by the All tab, which allows users to set values for all the Initialization properties supported by an OLE DB provider. The Extended Properties property can be set on this page. For the OpenLink provider, you should enter key-value pairs for any supported extended properties you wish to set, such as:

```
         Cursors=Driver
```

7.3.2 Using a Connection String

The connection string is used to contain connection specific details as semi-colon delimited key=value pairs so that the Broker/Agent facilitating the connection can chose the correct database and method. The details provided on various setup panels form the connect string automatically upon use. Applications can use hand made connection strings to avoid DSN configurations or prompting during exection. The parameters that can be used are as follows:

The connection string should contain a key-value pair for the Extended Properties keyword, such as:

PROVIDER=OpenLinkODBC

DRIVER={OpenLink Generic 32 Bit driver v4.0}

HOST=BrokerHostNameOrIP

PORT=BrokerListenPort

SVT=DBMSType/Provider

DATABASE=myDatabaseName

UID=myUsername

PWD=myPassword

PROTOCOL=TCP/IP

FBS=FetchBufferSize (Rows 1-99)

DLF=DeferLongFetch [y|n]

OPTIONS=opts

READONLY=[y|n]

DSN=myDSNname

```
        ;Extended Properties= Cursors=ODBC ;
```

Which is used for setting the cursor mode and/or bookmarking.

7.3.3 Using The DBPROP_INIT_PROVIDERSTRING Property

Applications can set this initialization property directly.

7.4 Controlling ODBC Cursor Library Usage

The OpenLink ODBC Provider requires that the underlying ODBC driver supports some form of scrollable cursor. In order to support OLE DB rowsets, the minimum scrolling functionality requirements include:

bookmark support

absolute cursor positioning

backwards scrolling

Keyset driven and static cursors both support this functionality. All OpenLink ODBC drivers support these cursor models. However, if the OpenLink ODBC provider is to be used with a third party driver, either the driver must support one of these cursor types natively, or the Microsoft ODBC Cursor Library must be used. The latter only supports static scrollable cursors, but provides sufficient functionality to allow third party drivers to be used with the OpenLink provider. When using an OpenLink ODBC driver, it is preferable not to use the Microsoft Cursor Library.

To control how the OpenLink Provider uses the ODBC Cursor Library, the provider supports the provider specific keyword Cursors in the Extended Properties property string. The keyword can be set to one of three values:

ODBC

Driver

IfNeeded

The meaning of each of these values is analogous to their ODBC counterpart SQL_CUR_USE_xxx. The default setting is Cursors=IfNeeded.

The table below lists the required Cursors setting for various ODBC drivers when used with the OpenLink ODBC Provider.

Table 5.1. Features Comparison

ODBC Driver	Cursor	Comments
All OpenLink ODBC drivers	Driver or IfNeeded (default)	The native scrollable cursor support in the OpenLink drivers provides the necessary functionality
Native Microsoft SQL Server driver	ODBC	The native scrollable cursor support is inadequate.

7.5 Controlling the Default Bookmark Setting for Rowsets

The extended property keyword BookmarkDefault controls whether rowsets are created with or without a bookmark column by default. OLE DB does not specify what the default behaviour should be. The OpenLink provider includes a bookmark column by default. Normally it should not be necessary to change this setting. To explicitly turn bookmarks on or off, use an entry of the form: BookmarkDefault=On|Off.

7.6 Known Issues

These are the known issues that apply to the OpenLink OLEDB provider as of September 2005:

7.6.1 General Issues

- The OpenLink ODBC Provider does not allow a consumer to specify the type of server-side cursors used internally. ADO consumers, for instance Visual Basic 6 and Visual Interdev 6, should use client-side cursors.
- Batched commands are not supported.

7.6.2 .Net

- Automatic generation of the DeleteCommand, InsertCommand and UpdateCommand properties of a DataAdapter by a CommandBuilder is not supported.
- Parameters must be bound in the order they occur in a command.

7.6.3 Visual Interdev 6

Visual Interdev uses the username you specify at connect time as the owner qualifier when referring to schema objects. If the target database uses case-sensitive qualifiers, the case of the username you specify when connecting must match the case of the owner qualifier of schema objects belonging to that user. For instance, if using the Oracle sample user SCOTT , connect as SCOTT not scott . If the incorrect case is used, Visual Interdev may fail to retrieve schema information. One manifestation of this problem is that the Query Builder may not list the columns belonging to a table.

7.6.4 Visual Basic 6 and ADO

When specifying the SQL for the Recordsource property in an ADO Data Control (ADO DC), you must quote the table name. If you do not, VB converts the SQL to lowercase and then quotes the table name when attempting an update. e.g. If you enter select * from emp , when attempting an update, this becomes update emp . Databases which support case-sensitive schema object names, e.g. Oracle, will then either reject this SQL, complaining that the table doesn t exist, or reference the wrong table. To avoid this problem, enter the query as select * from EMP .

7.6.5 SQL Server Data Transformation Services

The provider implements interface IRowsetFastLoad. This is an interface specific to the SQL Server OLE DB Provider and is required by SQL Server DTS, even on third party OLE DB providers. Export of tables from SQL Server to Oracle using DTS is possible.

The OpenLink OLE DB provider uses a scrollable cursor to support scrolling over rowsets. OpenLink scrollable cursors require a unique row identifer (primary key, rowid etc.) to act as a key column. If the table(s) used to populate the OLE DB rowset do not have a primary key or similar then the OpenLink OLEDB provider can still provide a rowset however it will be created as read only. With Oracle, the pseudo column rowid acts as a key - the table does not have to have a primary key to make the rowset writeable. With DB2, the table(s) must have a primary key otherwise the rowset will be read only.

If you choose to export all primary and foreign keys in the table, the DTS appears to assume it is talking to SQL Server when creating the target table. It uses a fixed SQL syntax in the CREATE TABLE command to express the primary key as a table constraint. Oracle accepts this syntax, however DB2 rejects it.

As a consequence of the above currently the only way to export a table to DB2 is to manually create the target table before launching DTS. If the target table already exists, DTS does not attempt to create it; it just copies the data from the source table. The manually created table must have a primary key.

The DTS does not handle quoting of the target table name correctly. The user must specify the target table name using the appropriate case which does not require quoting. e.g. When exporting to Oracle, give the table name in uppercase.

7.6.5.1 Provider Options in the SQL Server Enterprise manager

When setting up a linked server using the Enterprise Manager you can configure the way SQL Server uses your OLE DB provider by clicking on the Provider Options button just under the box where the provider is selected. The options in the Provider Options dialogue box apply to the provider rather than the specific server so any changes made here will effect all linked servers using that provider. A list of all the servers using the provider is given in the bottom half of the box.

Note that once a server has been created it is not possible to change these options in the property page for that server. To change them after the linked server has been created you have to create a new server and alter the options there. Making changes to these options when creating a new linked server affects all existing linked servers. Once you have set up a linked server the options you have chosen effectivley become the defaults for the provider you are using.

A document describing the Provider Options, called 'Configuring OLE DB Providers for distributed Queries', can be found in the SQL Server Books Online documentation or on the web at http://msdn.microsoft.com/library/default.asp?url=/library/en-us/adminsql/ad_1_server_8ib7.asp

By default none of the options is enabled for our driver. These options are specifically for SQLServer distributed queries using linked servers rather than being general OLE DB options.

7.6.5.2 Provider Options

Dynamic Parameters - you can set this if the provider supports the ICommandWithParameters interface. This in supported in the OpenLink provider.

NestedQueries - can be set if provider allows SELECT in a FROM clause. May cause concerns with the underlying Cursor libraries. This option should not be enabled if the underlying DSN or DBMS does not allow nested queries.

LevelZeroOnly - Should not be enabled.

Allow InProcess - This option should be checked. Checking this option stops errors like: `Error 7311: Could not obtain the schema rowset for OLE DB provider 'UNKNOWN'. The provider supports the interface, but returns a failure code when it is used. OLE DB error trace [OLE/DB Provider 'UNKNOWN' IDBSchemaRowset::GetRowset returned 0x800706c6:].`

Error 0x800706c6 is RPC_S_INVALID_BOUND. The error message identifies the OLE DB provider that the error comes from as UNKNOWN and our log files show no sign of the error. SQL Server uses a proxy/stub dll, msdaps.dll, to marshal oledb interfaces between processes/apartments. These errors from the 'UNKNOWN' provider come from msdaps.dll. If the OLE DB provider is invoked out of process or is marked as apartment model then this stub gets used. By default if the OLE DB provider is a Microsoft provider then SQL Server invokes it in process and this proxy is not used. Checking the Allow InProcess box when using our provider ensures that this stub dll is not used and so avoids this error.

NonTransactedUpdates - this is entirely up to the user to choose.

IndexAsAccessPath - should be left disabled.

DisallowAdhocAccess - Disables use of OPENROWSET and OPENDATASOURCE with the provider. Up to the user to choose

7.6.5.3 Connection Errors. Error 0x80004005

If the linked server has problems connecting then the following error is generated:

```
Error 7399: OLE DB provider 'OpenLinkODBC' reported and error. The provider
did not give any information about the error. OLE DB error trace [OLE/DB
Provider'OpenLinkODBC' IDBInitialize::Initialize returned 0x80004005: The
provider did not give any information about the error.].
```

Reasons for this error could be:

The ODBC DSN does not exist

The linked database is not started

Username or password not set up correctly on the security property page.

The first thing to check when getting this error is that the underlying ODBC DSN is connecting correctly and after that if the DSN name has been spelt correctly in the linked server definition. This error may also occur when using the OPENQUERY, OPENROWSET, and OPENDATASOURCE syntax. In addition to the connection errors listed above other causes of this error may be an incorrect table, column or owner name in the statement.

7.6.5.4 Using SQLServer Authentication

When using OPENROWSET or OPENDATASOURCE options on logging into SQL Server using SQL Server authentication, the following error might be seen:

```
Could not perform Windows NT authentication because delegation is not
available.
```

This error is avoided if the connection to the SQL Server is made using Windows NT authentication.

This problem only occurred when using OPENROWSET and OPENDATASOURCE not with OPENQUERY, i.e. only when the connection information was given in the query not when a linked server had been set up in the Enterprise Manager.

7.6.5.5 Linking Progress Servers.

Linking to Progress using a SQL-89 datasource works if the linked server is set up in the Enterprise Manager and the OPENQUERY syntax is used. Connections made using the OPENDATASOURCE or OPENROWSET syntax fail. The Progress SQL-89 engine does not handle the SQL generated by SQL Server when executing the OPENROWSET or OPENDATASOURCE query.

Linking to Progress using a SQL-92 datasource works using OPENQUERY, OPENDATASOURCE and OPENROWSET.

7.6.5.6 Inconsistent Metadata

Queries that fail with errors about inconsistent metadata usually indicate that there is a discrepancy between the information returned from SQLColumns and from SQLDescribeCol by the underlying ODBC driver.

7.7 Objects and Interfaces Implemented by the OpenLink Provider

The OpenLink provider implements four OLE DB objects.

- The data source object, which enables consumers to connect and initialize the interaction with the database.
- The session, which enables consumers to create a rowset for the data set in the database.
- The rowset, which exposes a data set to the consumer.
- The command, which enables consumers to create a SQL string for the data in the database.

These objects, along with the interfaces implemented in the OpenLink provider, are described below.

Table 5.2. OLE-DB Interfaces Implemented

Interface name	Method name
IDBInitialize	Initialize
	Uninitialize
IDBCreateSession	CreateSession
IDBProperties	GetProperties
	GetPropertyInfo
	SetProperties
IPersist	GetClassID
IGetDataSource	GetDataSource
ICommand	Cancel
	Execute
	GetDBSession
ICommandText	GetCommandText
	SetCommandText
ICommandProperties	GetProperties
	SetProperties
ICommandPrepare	Prepare

Interface name	Method name
	Unprepare
IOpenRowset	OpenRowset
IColumnsInfo	GetColumnInfo
	MapColumnIDs
IConvertType	CanConvert
IAccessor	AddRefAccessor
	CreateAccessor
	GetBindings
	ReleaseAccessor
IRowset	AddRefRows
	GetData
	GetNextRows
	ReleaseRows
	RestartPosition
IRowsetInfo	GetProperties
	GetReferencedRowset
	GetSpecification
IRowsetChange	DeleteRows
	InsertRow
	SetData
IRowsetLocate	Compare
	GetRowsAt
	GetRowsByBookmark
	Hash
ISessionProperties	GetProperties
	SetProperties
ISupportErrorInfo	InterfaceSupportsErrorInfo
IErrorInfo	GetDescription
	GetGUID
	GetHelpContext
	GetHelpFile
	GetSource
IErrorLookup	GetErrorDescription
	GetHelpInfo
	ReleaseErrors

Interface name	Method name
ISQLErrorInfo	GetSQLInfo

7.7.1 Invoking the OpenLink Provider

The OpenLink ODBC Provider (oploleod.dll) can be loaded using a variety of methods, depending on your programming environment.

7.7.1.1 Using CoCreateInstance

If you are writing an application which calls the OLE DB API directly, a consumer typically creates a provider's Data Source object by passing the provider's class identifier (CLSID) to the COM CoCreateInstance function and requesting the IDBInitialize interface. Every OLE DB provider declares a unique CLSID for itself. The CLSIDs for the OpenLink ODBC Provider, the OpenLink Error Lookup Service and the OpenLink ODBC Data Source Enumerator are given in the example below, which declares macros for each of the class identifiers.

```
#ifdef DBINITCONSTANTS
// Provider CLSID
EXTERN_C const CLSID CLSID_OPLOLEDB =
  {0X2D93A18D, 0XAC86, 0X11D1, {0X9C, 0XEA, 0XE5, 0X2A, 0X53, 0XBE, 0XA0, 0X7D}};
// OpenLink Error Lookup Service CLSID
EXTERN_C const CLSID CLSID_OPLOLEDB_ERROR =
  {0X2D93A18E, 0XAC86, 0X11D1, {0X9C, 0XEA, 0XE5, 0X2A, 0X53, 0XBE, 0XA0, 0X7D}};
// OpenLink ODBC Data Source Enumerator CLSID
EXTERN_C const CLSID CLSID_OPLOLEDB_ENUM =
  {0X2D93A190, 0XAC86, 0X11D1, {0X9C, 0XEA, 0XE5, 0X2A, 0X53, 0XBE, 0XA0, 0X7D}};
#else //DBINITCONSTANTS
EXTERN_C const CLSID CLSID_OPLOLEDB;
EXTERN_C const CLSID CLSID_OPLOLEDB_ERROR;
EXTERN_C const CLSID CLSID_OPLOLEDB_ENUM;
#endif //DBINITCONSTANTS
IDBInitialize * pIDBInitialize;
HRESULT hr;
hr = CoCreateInstance(CLSID_OPLOLEDB, NULL, CLSCTX_INPROC_SERVER,
      IID_IDBInitialize, (void**) &amp;pIDBInitialize);
if (FAILED(hr))
{
   // Display error
...
}
```

7.7.2 Connecting from ADO or .Net

The provider's name is "OpenLinkODBC". The OpenLink OLE DB Provider is invoked from ADO or .Net using a connection string which typically takes the following form:

Provider=OpenLinkODBC; Data Source=*w2ks2*;User ID=*SCOTT*;Password=*tiger*;Extended Properties="Cursors=Driver;BookmarkDefault=On";

7.7.3 Initialization Properties

When the consumer calls IDBInitialize::Initialize, the OpenLink Provider calls the UDBC/ODBC functions SQLSetConnectOption, to set various connection options, and SQLDriverConnect, to connect to an ODBC data source. The values passed to SQLSetConnectOption and SQLDriverConnect come from the initialization properties set by the consumer.

When the data source object is first created, the value of each initialization property is set to a default value.

7.7.3.1 Initialization Properties Used

The following lists the initialization properties currently used by the OpenLink Provider. Any properties not listed are not used by the OpenLink Provider.

Table 5.3. OLE-DB Initialization Properties

Property	Description
DBPROP_AUTH_PASSWORD	Passed as the value of the PWD keyword in SQLDriverConnect.
DBPROP_AUTH_USERID	Passed as the value of the UID keyword in SQLDriverConnect.
DBPROP_INIT_CATALOG	Sets the initial catalog for data sources which recognise catalogs.
DBPROP_INIT_DATASOURCE	Passed as the value of the DSN keyword in SQLDriverConnect.
DBPROP_INIT_HWND	Passed as the value of the hwnd argument in SQLDriverConnect.
DBPROP_INIT_MODE	Mapped to the ODBC connect option SQL_ACCESS_MODE.
DBPROP_INIT_PROMPT	Passed as the value of the fDriverCompletion parameter in SQLDriverConnect.
DBPROP_INIT_PROVIDERSTRING	Specifies extended properties for controlling ODBC Cursor Library usage (through the 'Cursors' keyword) and the whether bookmarks are exposed by default on rowsets (through the 'BookmarkDefault' keyword).
DBPROP_INIT_TIMEOUT	Mapped to the ODBC connect option SQL_LOGIN_TIMEOUT.

7.7.3.2 Setting and Getting Provider Properties

The data source object is the first object created when a consumer instantiates the provider by calling *CoCreateInstance*.

The data source object provides the starting point for communications between the provider and consumer. For example, a consumer can call *CoCreateInstance* and request an *IDBInitialize* interface pointer to instantiate a data source object. The provider has a CLSID (class ID) that is stored in the Windows Registry. The consumer can use this CLSID with *CoCreateInstance* to instantiate the data source object. The OpenLink provider setup program registers the OpenLink provider in the Windows Registry.

The data source object is responsible for setting and returning information about the properties supported by the provider and exposing the list of supported keywords and literals. This functionality is supported through the mandatory *IDBProperties* interface and the optional *IDBInfo* interface. The *IDBProperties* interface contains three methods:

- *GetProperties* returns the list of properties currently set on the data source object.
- *GetPropertyInfo* returns information about supported rowset and data source properties.
- *SetProperties* sets the properties on the data source object.

The *IDBInfo* interface contains two methods:

- *GetKeywords* returns a list of supported keywords.
- *GetLiteralInfo* returns information about literals used in text commands.

7.7.4 Initializing and Uninitializing the Data Source Object

The *IDBInitialize* interface contains two methods: *Initialize* and *Uninitialize*.

Initialize enables consumers to explicitly initialize a data source object. Consumers must set properties on the data source object before attempting to initialize it; and consumers must supply a valid datasource name to the database in IDBProperties::SetProperties. If the datasource is invalid, the OpenLink provider returns an E_FAIL error on initialization.

Uninitialize enables consumers to return the data source object to an uninitialized state. It is an error to call *IDBInitialize::Uninitialize* when there are open sessions or rowsets on the data source object.

7.7.5 Creating a Session

After you initialize the data source object, you must create a session object to manage the session and provide the framework needed to create a rowset with *IOpenRowset::OpenRowset*. The *IDBCreateSession::CreateSession* interface on the data source object enables you to create a new session object and returns an interface pointer to the session.

Once the session has been created, the provider must expose the interface pointer to the data source object that created the session. This interface pointer is exposed through the mandatory interface IGetDataSource.

7.7.6 Creating a Rowset

The session contains the interface that enables consumers to open a database and create a rowset object containing all rows in the database. The OpenLink provider implements both the *IOpenRowset*, and the *ICommand* (and associated *ICommandText, ICommandProperties)* interfaces on the session to create a rowset..

7.7.6.1 Instantiating and Exposing a Rowset

The *IOpenRowset* interface contains a single method: *OpenRowset*. *IOpenRowset* is a required interface on the session. *IOpenRowset::OpenRowset* can be used by consumers that do not support command objects to generate a rowset of all rows in a table or index.

The *ICommand* interface contains the method: *Execute*. *ICommand::Execute* generates a rowset from the SQL query set by *ICommandText::SetCommandText*.

7.7.6.2 Consumer and Provider Interactions with the Rowset

After receiving the rowset interface pointer, the consumer can request rowset metadata from the provider through *IColumnsInfo*. The consumer then creates bindings by requesting *IAccessor* from the provider and specifying the bindings through *IAccessor::CreateAccessor*. The provider returns a handle to the accessor to the consumer.

The consumer then requests a number of rows from the provider using *IRowset::GetNextRows*. The provider retrieves the data for these rows and stores it in the data cache. The provider then returns an array of row handles to the consumer. Each row handle returned by the provider has an initial reference count of one. The consumer is then free to get the data for any rows from the provider using *GetData*. The consumer supplies *GetData* with the row handle, the handle of an accessor, and the buffer location into which to return the data; the provider copies the data to the location specified by the consumer.

To update rows, consumers call *IRowsetChange::SetData*, which sets the data in the data cache to the values specified by the consumer. To delete rows from the rowset, the consumer calls *IRowsetChange::DeleteRows*. To insert rows into the rowset, the consumer calls *IRowsetChange::InsertRow*. Note that the OpenLink Provider is not able to fetch back a newly inserted row if the underlying datasource does not provide this functionality.

When the consumer makes any change to data in the data cache, the effects of the change are written to the data source immediately. OLE DB specifies a change-buffering model, which enables the consumer to make changes that are not realized until the consumer calls *IRowsetUpdate::Update*; this model is not supported by the OpenLink provider.

When the consumer has finished working with a row, it can release the row by calling *IRowset::ReleaseRows*. *ReleaseRows* simply decrements the reference count on the row in the data cache. If the reference count for that row reaches zero, the row data is released from the data cache.

7.7.7 Exposing Metadata

the OpenLink Provider expose information about the columns of a rowset through *IColumnsInfo*. The information for each column is returned in a DBCOLUMNINFO structure. OLE DB also has an optional metadata interface, *IColumnsRowset*; the OpenLink provider does not implement this interface.

The *GetColumnInfo* method returns metadata that is most commonly used by consumers: column ID, column name, the ordinal number of the column in the rowset, the column's data type, and so on.

The provider returns the information in an array of DBCOLUMNINFO structures, one DBCOLUMNINFO structure per column in the rowset. The order of the structures returned in the array is the order in which the columns appear in the rowset.

7.7.7.1 IColumnsInfo

Columns that have an ODBC SQL type of SQL_LONGVARCHAR or SQL_LONGVARBINARY are returned as type DBTYPE_BYTES or DBTYPE_STR, and the DBCOLUMNFLAG_ISLONG is set in the dwFlags element of the DBCOLUMNINFO structure.

7.7.7.2 Returning Column Ordinals

Columns in a rowset are identified by a column ID, which is a value of type DBID in the DBCOLUMNINFO structure.

The *MapColumnIDs* method returns column ordinals for all column IDs provided in the *rgColumnIDs* array. Column ordinals do not change during the life of the rowset, but may change between different instances of the rowset.

7.7.8 Supported Conversions

Before the consumer creates an accessor, it can call *IConvertType::CanConvert* to determine if the provider supports a particular conversion.

7.7.8.1 Default Data Type Mapping

The OpenLink Provider binds to the ODBC/UDBC data source using the types in the table below. The SQL type is queried using SQLDescribeCol. The sign of the data type (signed/unsigned) is determined using SQLColAttributes. It is used in deciding which C type to use in internal buffers and which type indicator to return through *IColumnsInfo::GetColumnInfo*.

Table 5.4. OLE-DB Data Type Mappings

SQL Type Indicator	Indicator of C Type Used For Internal Buffers	OLE DB Type Indicator
SQL_CHAR, SQL_VARCHAR, SQL_LONGVARCHAR, SQL_DECIMAL, SQL_NUMERIC	SQL_C_CHAR	DBTYPE_STR
SQL_BIT	SQL_C_BIT	DBTYPE_BOOL
SQL_TINYINT, SQL_SMALLINT	SQL_C_USHORT, SQL_C_SSHORT	DBTYPE_I2
SQL_INTEGER	SQL_C_ULONG, SQL_C_SLONG	DBTYPE_I4
SQL_BIGINT	SQL_C_STR	DBTYPE_STR
SQL_REAL	SQL_C_FLOAT	DBTYPE_R4
SQL_FLOAT, SQL_DOUBLE	SQL_C_DOUBLE	DBTYPE_R8
SQL_BINARY, SQL_VARBINARY, SQL_LONGVARBINARY	SQL_C_BINARY	DBTYPE_BYTES
SQL_DATE	SQL_C_DATE	DBTYPE_DATE
SQL_TIME	SQL_C_TIME	DBTYPE_DATE
SQL_TIMESTAMP	SQL_C_TIMESTAMP	DBTYPE_DATE

7.7.8.2 Length Binding

For types DBTYPE_UI1, DBTYPE_I2, DBTYPE_I4, DBTYPE_I8, DBTYPE_R4, DBTYPE_R8, DBTYPE_CY, DBTYPE_NUMERIC, the length binding is always set to the fixed size of the destination binding type, rather than the internal source type.

7.7.8.3 Supported Data Conversions

The follwoing table outlines the supported type conversion implemented in the OpenLink provider. An 'X' means supported, and '-' means not supported.

Table 5.5. OLE-DB Data Type Conversions

	I1	I2	I4	I8	UI1	UI2	UI4	UI8	R4	R8	CY	DEC	NUM	BOOL	DATE	DBDATE	DBTIMESTAMP	DBTIME	BY
I1	X	X	X	X	X	X	X	X	X	X	X	X	X	-	-	-	-	-	X
I2	X	X	X	X	X	X	X	X	X	X	X	X	X	-	-	-	-	-	X
I4	X	X	X	X	X	X	X	X	X	X	X	X	X	-	-	-	-	-	X
I8	X	X	X	X	X	X	X	X	X	X	X	X	X	-	-	-	-	-	X
UI1	X	X	X	X	X	X	X	X	X	X	X	X	X	-	-	-	-	-	X
UI2	X	X	X	X	X	X	X	X	X	X	X	X	X	-	-	-	-	-	X
UI4	X	X	X	X	X	X	X	X	X	X	X	X	X	-	-	-	-	-	X
UI8	X	X	X	X	X	X	X	X	X	X	X	X	X	-	-	-	-	-	X
R4	X	X	X	X	X	X	X	X	X	X	X	X	X	-	-	-	-	-	X
R8	X	X	X	X	X	X	X	X	X	X	X	X	X	-	-	-	-	-	X
CY	X	X	X	X	X	X	X	X	X	X	X	X	X	-	-	-	-	-	X
DEC	X	X	X	X	X	X	X	X	X	X	X	X	X	-	-	-	-	-	X
NUM	X	X	X	X	X	X	X	X	X	X	X	X	X	-	-	-	-	-	X
BOOL	-	-	-	-	-	-	-	-	-	-	-	-	-	X	-	-	-	-	X
DATE	-	-	-	-	-	-	-	-	-	-	-	-	-	-	X	X	X	X	X
DBDATE	-	-	-	-	-	-	-	-	-	-	-	-	-	-	X	X	-	X	X
DBTIME	-	-	-	-	-	-	-	-	-	-	-	-	-	-	X	-	X	X	X
DBTIMESTAMP	-	-	-	-	-	-	-	-	-	-	-	-	-	-	X	X	X	X	X
BYTES	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	-	X	X
BSTR	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
STR	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
WSTR	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
DISP	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	X
UNK	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	X
GUID	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	X

7.7.9 Creating and Using Accessors

Consumers describe the memory structure for their buffers through a process called binding. An accessor is a group of bindings. The OpenLink provider does not currently support reference accessors, which allow the consumer direct access to the rowset's data cache.

Accessors are implemented through *IAccessor*. Accessors are created with *IAccessor::CreateAccessor* and released with *IAccessor::ReleaseAccessor*. *IAccessor::GetBindings* can be used to determine the bindings in an existing accessor. IAccessor::AddRefAccessor enables the consumer to add a reference count to an existing accessor.

Accessors may not always be validated immediately at the time of creation. Instead, they may be validated at the time the first row is fetched. Errors will be returned at the first attempt to use such an accessor.

Creating an Accessor

CreateAccessor associates a set of bindings with an accessor handle that is used to send data to or fetch data from the rowset's data cache. The OpenLink provider supports only the DBACCESSOR_ROWDATA accessor flag, which specifies that the accessor is to be used for rowset data.

Returning Accessor Bindings

GetBindings returns the bindings in an existing accessor.

Adding a Reference Count to an Existing Accessor

AddRefAccessor adds a reference count to an existing accessor.

Releasing an Accessor

ReleaseAccessor decrements the reference count on an accessor; when the reference count reaches zero, the accessor is released.

7.7.10 Rowset MetaData

IRowsetInfo enables consumers to query the properties of a rowset through *IRowsetInfo::GetProperties*. Consumers can get an interface pointer to the object that created the rowset by calling *IRowsetInfo::GetSpecification*.

IRowset provides methods for fetching rows sequentially, exposing data from those rows to consumers, and managing the rows in the rowset. *IRowset* contains five methods: *AddRefRows*, *GetData*, *GetNextRows*, *ReleaseRows*, and *RestartPosition*.

7.7.10.1 Incrementing the Reference Count on Row Handles

AddRefRows increments the reference count on the row handles supplied by the caller. *AddRefRows* enables consumers to make multiple references to a row in the data cache.

7.7.10.2 Populating the Data Cache

IRowset::GetNextRows gets the next sequence of rows from the datasource and places them in the rowset's data cache. When *GetNextRows* is first called, it starts at the beginning of the rowset. After that, *GetNextRows* maintains information about its current position so it can proceed forward from that position. The OpenLink provider also provides support for *IRowset::RestartPosition*, which repositions the next fetch position to the beginning of the rowset.

7.7.10.3 Retrieving Data from the Data Cache

IRowset::GetData enables consumers to retrieve data from the data cache. *GetData* uses the bindings in the accessor to determine how the data should be returned and what data should be returned to the consumer's buffer. Then, *GetData* converts the data in the cache to the type specified in the binding and transfers the converted data to the consumer.

7.7.10.4 Decrementing the Reference Count on Row Handles

IRowset::ReleaseRows decrements the reference count on the rows specified. A consumer must call *ReleaseRows* once for each time a row was fetched or each time the row had its reference count incremented by *AddRefRow*. When the reference count reaches zero, the row is released if the rowset is in immediate update mode.

In providers that implement *IRowsetUpdate,* rows are released unless there are pending changes on the row. The OpenLink provider does not implement this interface; the OpenLink provider always performs rowset updates in immediate mode, which means that changes are immediately applied to the underlying data source. Therefore, the OpenLink provider does not recognize any changes as pending.

7.7.10.5 Returning to the First Row of the Rowset

IRowset::RestartPosition moves the next fetch position used by *GetNextRows* to the first row of the rowset.

7.7.10.6 Updating Rows

IRowsetChange enables consumers to change the values of columns in a row of data. If the consumer wants to change the data, it must first construct an accessor for the columns to be changed. *IRowsetChange* contains three methods: DeleteRows, InsertRow, and *SetData*.

7.7.10.7 Deleting Rows

IRowsetChange also enables consumers to delete rows from the rowset. *IRowsetChange*::*DeleteRows* deletes rows from the rowset and are applied to the data source immediately.

7.7.10.8 IRowsetChange

IRowsetChange is implemented using the UDBC/ODBC function SQLSetPos. It therefore can be exposed only when the underlying datasource supports SQLSetPos. Newly inserted rows cannot be updated.

8 Chapter 6. Server Components Installation

Abstract
Information on how to install components of the Server.

Table of Contents

8.1 OpenLink License Manager Usage Notes

8.1.1 Background

As of UDA release 6.0 and above, OpenLink have moved the handling of licenses from individual products into a specific License Manager process. This takes the form of an executable, (`oplmgr'), from which all OpenLink commercial products request licenses via network connections.

8.1.2 Single-Tier

OpenLink UDA Single-Tier is a single driver installed on the client only.

For releases 6.0, the oplmgr process was started automatically by the driver on first connection. As of release 6.1, this behaviour has changed; in order to facilitate use of License Manager process for administering licenses of all OpenLink products simultaneously on the same machine, the license-manager must be started explicitly started in advance of services that will use it. The release 6.1 installers now check if a License Manager (oplmgr) process is already running and if not start their own local instance.

8.1.3 Multi-Tier

OpenLink UDA Multi-Tier drivers comprise at least 3 components: a generic client installed on client machines, all of which contact a central request broker which spawns an RDBMS-specific database agent to connect to the specific database requested. The request-broker asks the license-manager for licenses for every connection requested.

For UDA release 6.0, the oplmgr process was started automatically by the request-broker (oplrqb). As of release 6.1, this behaviour has changed. In order that you should only need one license-manager per server, handling licenses for a variety of products (particularly combinations of Multi-Tier and OpenLink Virtuoso Universal Server), the license-manager must now be explicitly started before other services requiring it. The release 6.1 installers now check if a License Manager (oplmgr) process is already running and if not start their own local instance.

8.1.4 How to stop/start the OpenLink License Manager

The license manager takes the following commandline options:

```
      bash$ oplmgr --help
      OpenLink License Manager
      Version 1.2.2 as of Thu Feb 15 2007 (Release 6.0 cvsid 00084).
      Compiled for Linux 2.4.20-46.9.legacysmp
(i686-generic-linux-glibc23-32)
      Copyright (C) OpenLink Software.
      Usage:
      oplmgr [-shrutp] [+start] [+stop] [+reload] [+user arg] [+chroot arg]
      [+pidfile arg]
      +start     start the license manager
      +stop      stop the license manager
      +reload    force a configuration reload
      +user      run as the specified user
      +chroot    perform a chroot to the specified directory
      +pidfile   pid file to use for server operation
```

We recommend that you create an /etc/init.d/ script that runs `oplmgr +start' on boot-up.

8.1.5 Environment Variables

The OpenLink License Manager will search through directories in the OPL_LICENSE_DIR variable or failing that, the PATH environment variable, for files matching *.lic.

OpenLink recommends you use /etc/oplmgr/ to store your licenses; each product installation will include a copy of the oplmgr executable in its respective 'bin' directory, such that if this is the only OpenLink product on the system, it can be manually started and used for processing licenses with an appropriate OPL_LICENSE_DIR value. A generic system startup script is also being developed for Unix systems to enable the License Manager process to be automatically started on machine boot. If found, product installers will automatically append this to your OPL_LICENSE_DIR variable.

8.1.6 OpenLink License Manager Networking Considerations

The OpenLink License Manager sends and receives using the multicast IP address 224.0.0.24 on port 60001/udp to communicate between components and other license-managers that might be on your network. In the event that it cannot establish this multicast communication, it may cease allocating licenses, so in the event of license allocation-related errors, please check your firewall configuration permits this traffic.

8.2 Product Licensing

8.2.1 Initial License

When you obtain a Data Access Driver from the OpenLink Web Site, a license file will automatically be emailed to you. Please ensure the email address you have provided is correct. If you obtained the Data Access Driver from a CD, you may need to apply for an evaluation license through our Web site, or by contacting one of our offices. The standard license will provide 30 days to evaluate the product.

8.2.2 How to apply the License

You will be prompted for the location of your license file during the initial installation. If you have not yet received a license file, you may still install the product, and manually apply the license file when you do receive it.

To use the license file, simply place it in one of the following locations:

- Place the license file in the default location for your Operating System.
    - For Linux and Unix - $OPENLINK/bin.
    - For Windows -by default, C:\Program Files\OpenLink\bin. If you chose a different installation target directory, the bin directory beneath your choice.

  - ♦ For Mac OS X - //Library/Application Support/openlink/bin.
  - ♦ For Mac OS 9 or earlier - the Preferences folder, within the System Folder.
- Alternatively, you may define the environment variable OPL_LICENSE_DIR to specify the directory in which all OpenLink license files can be found. (Does not apply to Mac OS 9 or earlier.)
- Finally, license files may reside in any directory included in the PATH environment variable of the host machine. (Does not apply to Mac OS 9 or earlier.)

License errors are generally found in the Request Broker log file, oplrqb.log.

If your product fails to register properly, then your license may be corrupt or invalid. Please check the license details against the name and version of the Driver installed, and contact OpenLink Software Technical Support for assistance.

8.2.3 Note:

only the first license file found for a particular product is examined, and an expired license may prevent use of an unexpired license

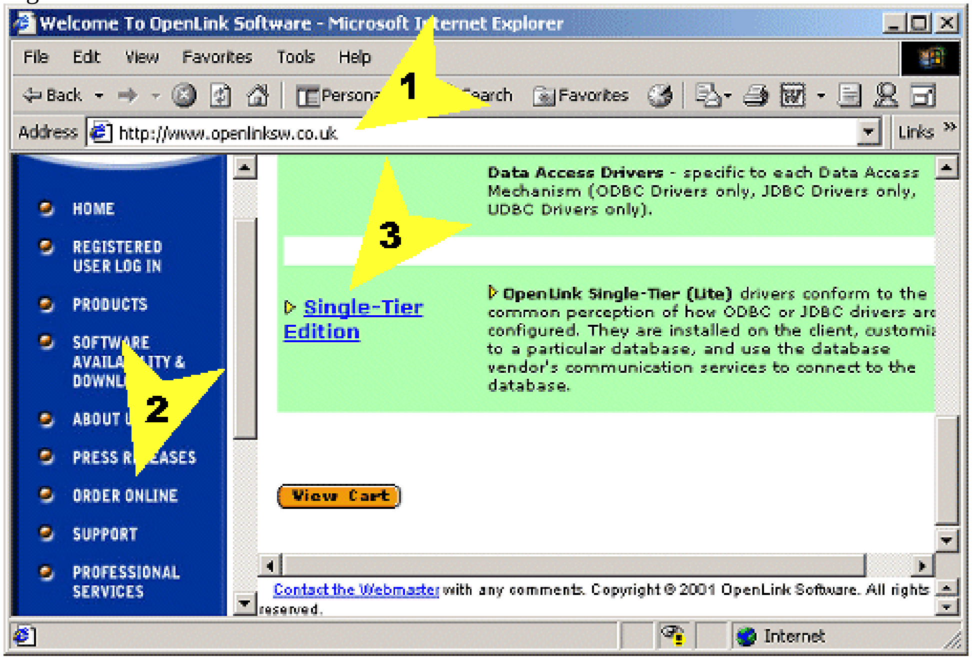
8.3 Upgrading the License

You may purchase a permanent license key online via the Internet. To purchase a key follow these instructions:

1. Use an Internet browser to connect to one of the following web sites:

   - ♦ North and South America: http://www.openlinksw.com
   - ♦ Europe, Africa: http://www.openlinksw.co.uk
   - ♦ Asia, Australia: http://www.openlinksw.com.my

   Figure 6.1. Online order


Welcome To OpenLink Software - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Personal Search Favorites
Address http://www.openlinksw.co.uk
Links
1
2
3
HOME
REGISTERED USER LOG IN
PRODUCTS
SOFTWARE AVAILABILITY & DOWNLOAD
ABOUT US
PRESS RELEASES
ORDER ONLINE
SUPPORT
PROFESSIONAL SERVICES
Single-Tier Edition
View Cart
Internet

Data Access Drivers - specific to each Data Access Mechanism (ODBC Drivers only, JDBC Drivers only, UDBC Drivers only).

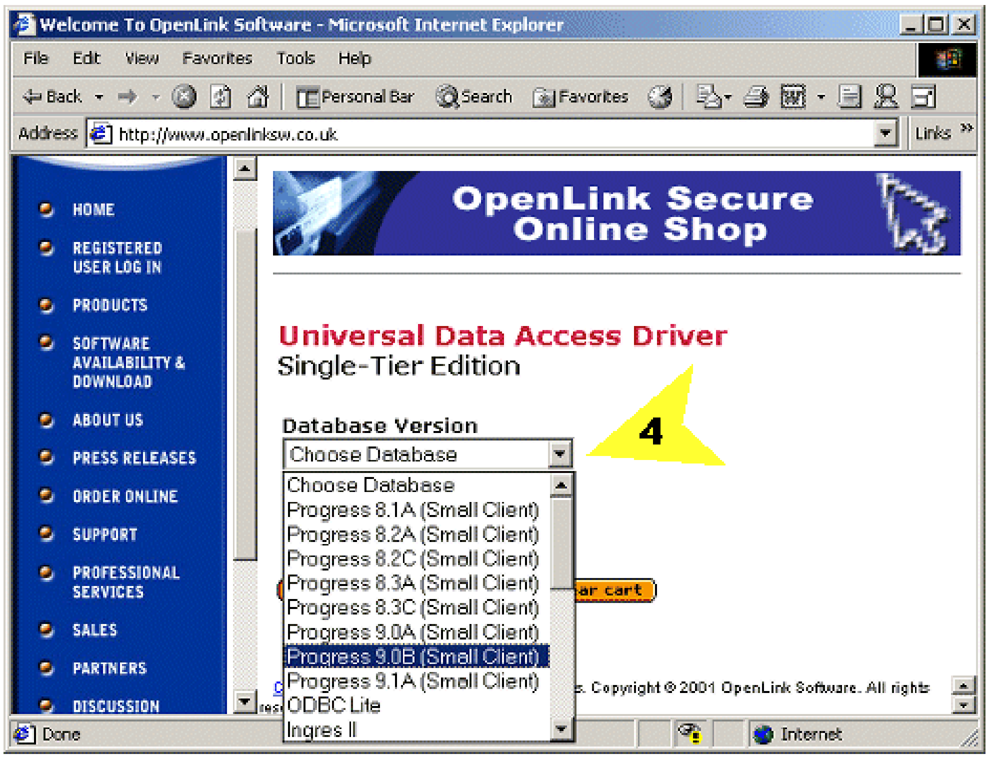
OpenLink Single-Tier (Lite) drivers conform to the common perception of how ODBC or JDBC drivers are configured. They are installed on the client, customi to a particular database, and use the database vendor's communication services to connect to the database.

Contact the Webmaster with any comments. Copyright © 2001 OpenLink Software. All rights reserved.

2. Select the ORDER ONLINE menu link from the left side of the web page.

   A new web pages appears listing links to different online forms for varying products.
3. Choose the link to the correct product.
4. Choose the Client Operating System

   Figure 6.2. License detail selection


Welcome To OpenLink Software - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://www.openlinksw.co.uk
HOME
REGISTERED USER LOG IN
PRODUCTS
SOFTWARE AVAILABILITY & DOWNLOAD
ABOUT US
PRESS RELEASES
ORDER ONLINE
SUPPORT
PROFESSIONAL SERVICES
SALES
PARTNERS
DISCUSSION
OpenLink Secure Online Shop
Universal Data Access Driver
Single-Tier Edition
Database Version
4
Choose Database
Progress 8.1A (Small Client)
Progress 8.2A (Small Client)
Progress 8.2C (Small Client)
Progress 8.3A (Small Client)
Progress 8.3C (Small Client)
Progress 9.0A (Small Client)
Progress 9.0B (Small Client)
Progress 9.1A (Small Client)
ODBC Lite
Ingres II
Copyright © 2001 OpenLink Software. All rights
Done
Internet

5. Choose the Server Operating System
6. Choose the Data Access Service
7. Choose the Licensing scheme required
8. Choose the Number of CPUs that are to be controlled by the license.
9. Add your product to the shopping cart, and follow the on screen instructions for completing the license purchase.

In the event that you are unable to use the online ordering system, then please contact sales@openlinksw.com or sales@openlinksw.co.uk to obtain the permanent license key.

8.4 Windows 95/98/NT/XP/2000 Install

1. Ensure you have downloaded all of the relevant OpenLink Server Components from OpenLink Software Download wizard.
2. From the OpenLink email that is delivered at the time of download, save the attached license file to a temporary location.
3. Ensure all existing OpenLink server and client applications are shutdown.
4. Close the Services Control Panel Applet to avoid any locking of existing OpenLink services.
5. Run setup.exe, please note that the client and server components for Windows are packaged within the same installation archive. You distinguish the actual components that you want to install as part of your interaction with the installation program.

8.4.1 Caution:

please do not choose SPX/IPX protocol support during the installation process unless you have verified that this protocol is actually installed and configured on your Windows Server machine.

6. Once the installer has completed, the broker will be running and ready for use, assuming a valid license was supplied.

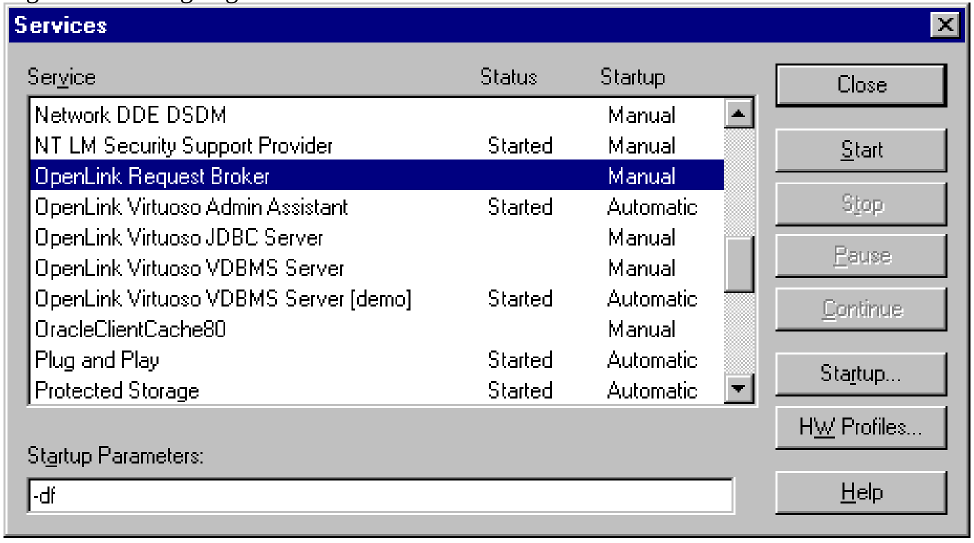
If the OpenLink Broker does not respond to client requests, or there was a Broker creation error during the install, then you can verify its operation as follows:

1. Start your OpenLink Request Broker in debug mode, from within your "services" control panel, or by opening up a DOS command window and then executing the OpenLink Request Broker startup command from within the "bin" sub-directory of the OpenLink server components installation directory:

```
        oplrqb -dv
```

Note: When doing this from the "services" control panel (Windows NT) you must stop the Broker if it is already running and then change its startup mode from "Automatic" to "Manual", then enter the required startup commands as values in the Startup Parameters field, the screen shot below demonstrates this:

Figure 6.3. 3image2.gif



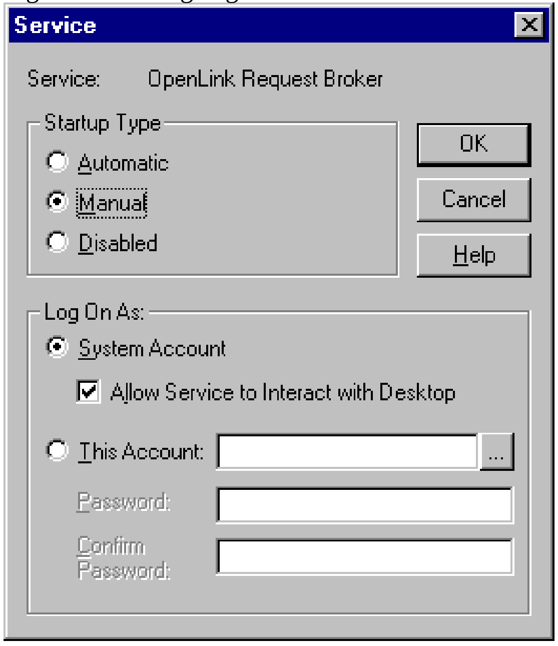
Services

Service	Status	Startup
Network DDE DSDM		Manual
NT LM Security Support Provider	Started	Manual
OpenLink Request Broker		Manual
OpenLink Virtuoso Admin Assistant	Started	Automatic
OpenLink Virtuoso JDBC Server		Manual
OpenLink Virtuoso VDBMS Server		Manual
OpenLink Virtuoso VDBMS Server [demo]	Started	Automatic
OracleClientCache80		Manual
Plug and Play	Started	Automatic
Protected Storage	Started	Automatic

Startup Parameters: -df

Selecting the service clicking the *Startup* button will give the following screen:

Figure 6.4. 3image3.gif



Service

Service: OpenLink Request Broker

Startup Type: Automatic [✓ Manual] Disabled

Log On As: [✓ System Account]

[✓] Allow Service to Interact with Desktop

This Account:

Password:

Confirm Password:

2. Start your Web Browser and then enter the following URL: http://<network alias of machine on to which you have just installed server components>:<port number you provided when prompted by OpenLink installer>. For example, if your machine has a network alias of "mainserver" and you accept the default port number at installation time, then the required URL would be constructed as follows: *http://mainserver:8000/* you can also enter the value *http://localhost:8000/* if the server in use is local rather than remote.
3. Follow the instructions provided in the "Making Your First OpenLink Connection" guide in order to verify your server components installation.
4. If the previous step is successful, shutdown the Request Broker by pressing: *<CTRL-C>* in the debug session window, or by selecting the OpenLink Request Broker entry within the "services" control panel and then clicking on the "stop" button.

   You can also shutdown the Broker by executing the following command from a separate DOS command Window:

```
oplshut -f
```

5. Revert the OpenLink Request Broker startup mode back to "Automatic" from within the "Services" control panel and then restart by clicking on the "Startup" button. Note the Broker does not have to be in "Automatic" mode for regular use, it is your choice as to the startup mode that best suits your operational needs.

   You can also start the Broker from a DOS Window by executing the command:

   ```
   oplrqb -v
   ```

8.5 Mac OS X

OpenLink Software's server components are fully supported on Mac OS X. Users of Mac OS X 10.2.x ("Jaguar") and later may use the fully migrated GUI components; users of Mac OS X 10.1.5 and earlier must use the Darwin-based components (installation covered in the UNIX and Linux section, below).

As shipped from Apple, Mac OS X did not include any ODBC support until Jaguar. The ODBC Driver Manager, Data Source Administrator, etc. - all had to be delivered with the Drivers.

When Apple introduced Darwin, an Open-Source project meant to form the kernel of Mac OS X, OpenLink determined to port its open-source iODBC Driver Manager to the new platform. With the introduction of the Preview Release of Mac OS X, the traditional set of dynamic libraries was broadened to include a system of Frameworks to encourage the development of fully ODBC compliant, native Mac OS X drivers and client applications.

That set of Frameworks, along with the OpenLink ODBC Administrator (then known as the iODBC Administrator), has been included with all OpenLink installations for Mac OS X since 10.0. The OpenLink ODBC Administrator supports all core features of ODBC, and presents driver-specific DSN configuration panels, as defined by the Driver developer, through the use of Setup Libraries.

Jaguar's release marked Apple's recognition that Data Access was an important part of an Enterprise Operating System. Already part of the standard distribution of Darwin, the basic iODBC dynamic libraries are now a part of the standard installation of Mac OS X 10.2. Apple also included their own version of an ODBC Administrator, as a proof-of-concept. Apple's Administrator permits Driver Registration, Tracing, and all other core features of ODBC; however, among other hard edges, all DSN configuration must be done by manually entering Keyword-Value pairs. Further, the user must know what Keywords to use, along with their acceptable Values.

Each OpenLink Generic Multi-Tier Driver for ODBC for Mac OS X is distributed in a single disk image (.dmg) file, which contains a Macintosh Installer mpkg. You may download an installer for a single back-end DBMS, or a "kitchen-sink" installer containing components for all OpenLink supported DBMS.

The OpenLink ODBC Server Components for Mac OS X comprise the following:

Table 6.1. OpenLink ODBC Server Components for Mac OS X

Component	Purpose
Request Broker	The oplrqb executable Broker manages all Client connections, as instructed by the Rulebook, oplrqb.ini, and as restricted by the active License file(s), *.lic. Without an active Broker, no connection will be initiated.
Session Rules	Book The oplrqb.ini file holds all non-license parameters governing behavior of the Request Broker and Database Agents. This file may contain references to other subsidiary parameter files, which are incorporated into the active Rulebook at Broker launch.
OpenLink Database Agents	Any of a number of executables, named xxxx_sv or xxxx_mv, for single- and multi-threaded agents, respectively. xxxx is replaced by the OpenLink code for a given Database. For example, virt_mv is the OpenLink Virtuoso Database Agent (multi threaded).
Broker and Database Agent license files	File(s) containing software activation and license key for the Broker and Database agents. Primary license, for Request Broker, must be named oplrqb.lic; other licenses will all end in .lic, and will typically be named similarly to a specific database agent, e.g., jdbc13_mv.lic.
OpenLink HTTP and Tcl Engine	The www_sv executable is an HTTP (Web) Server, implemented as an OpenLink Service-Providing Agent. This component provides the core upon which the OpenLink Admin Assistant has been built.

Component	Purpose
HTTP and Tcl Engine configuration file	The www_sv.ini file contains most parameters relevant to the Admin Assistant's provision, most importantly including User Authentication information, including encoded passwords, for all authorized Administrators of the OpenLink components.
HTTP and Tcl Engine configuration program	The setup executable is automatically executed as part of the Installation process. Manual activation is only necessary if you have lost your Admin Assistant's primary Username and Password.
OpenLink ODBC Client Components	As specified in the Client configuration chapter.

8.5.1 Installation

Double-click the mpkg to start the installation process.

Figure 6.5. OpenLinkUDA-5.0.dmg

You must have an Administration username and password to install the OpenLink Generic Multi-Tier Driver for ODBC. The Jaguar Installer will immediately prompt you to enter your Mac OS X Username and Password.

Figure 6.6. Authentication

The installer will display a "Welcome" message. Click "Continue".

Figure 6.7. Welcome

The next screen will display the Read-Me file, including any last-minute updates to these documents. Please read carefully, and click "Continue" when finished.

Figure 6.8. Read Me

The next screen will display the License Agreement for the OpenLink Single-Tier Driver for ODBC. Please read, and click "Continue".

Figure 6.9. Software License Agreement

You will be prompted to "Agree" to continue the installation, or "Disagree" to abort.

Figure 6.10. Agree or Disagree to Licensing

You will be asked to select a Destination Volume. Generally, this should be your Mac OS X boot volume. Click on the desired disk icon, and then click "Continue".

Figure 6.11. Choose Destination

You may now choose the Easy Install, or if you are an experienced user, you may Customize which components are installed. OpenLink generally recommends the Easy Install.

If you have installed OpenLink or iODBC components in the past, click "Upgrade" to continue; otherwise, click "Install".

Figure 6.12. Easy Install

In a custom installation, you may choose which Client, Documentation, and Server components are installed. OpenLink recommends that all Servers be equipped with all Client components, and that you install at least the HTML documentation, which will then be available through the Admin Assistant. The iODBC Runtime is necessary for all Client connections, but you may choose not to install it, if you have already installed a later revision.

Figure 6.13. Custom Install

You will be shown a graphical progress bar as Installation progresses, followed by System Optimization.

Figure 6.14. Installation Progress

During the installation process, the Installer will prompt for some parameters. Only an Administrator may configure the Server components, and this authentication is independent of the Installer itself.

Figure 6.15. Configuration Authentication

First, the Installer will configure the Request Broker `Listener` port. By default, this is 5000, but you may set it to anything appropriate to your environment. Note that all Client DSNs must target the port specified herein.

If you check off the box labeled Automatically launch the request broker during system startup, a StartupItem will be installed and enabled, to automatically start the Broker at system boot. If you do not check this box, the Broker must be manually launched to permit data access.

Figure 6.16. Initial configuration, Request Broker tab

The next pane is to configure the HTTP-based OpenLink Administrator. By default, this service will listen at port 8000, but you may set it to any port appropriate to your environment. This port will be specified in Web browsers, to access the Admin Assistant, as in the URL, <http://localhost:8000/>.

Figure 6.17. Initial configuration, Web Administrator tab

When the process is complete, you will be told that the software was successfully installed. Click "Close" and wait a few moments, while the Request Broker is launched in the background.

Figure 6.18. Installation Completes

Installation is complete, and your Broker is available for use, when a new Web browser window is opened, connecting to the newly installed Broker's Admin Assistant.

Figure 6.19. Admin Assistant Browser Window

A new Finder window will also be opened, containing launch icons for the documentation you've installed (HTML and/or PDF), as well as manual Start and Stop scripts for the Request Broker.

Figure 6.20. Finder Window

Proceed to the next chapter, to learn how to configure your new software.

8.6 UNIX & Linux Install

1. Ensure you have downloaded all of the relevant OpenLink Server Components from the page created by the OpenLink Software Download wizard, this must include the OpenLink Server Components Installation shell script file "install.sh" in addition to the compressed "tar" archives for the OpenLink Request Broker and relevant Database Agent Components.
2. From the OpenLink email that is delivered at the time of download, save the attached license file to the same location as the install.sh.
3. Log on to your Database or Application server machine using the userid and password combination that you normally use when connecting to the database in non ODBC/JDBC/UDBC mode.
4. Determine your operating system group membership by typing in the command:

   ```
   id
   ```

   The "gid" value indicates your current operating system group membership (remember this for use later on during the installation process).
5. Create an OpenLink installation directory on the server (e.g., /usr/openlink) and transfer to it all the server component files you downloaded (including compressed TAZ archives and the installation shell script "install.sh").

8.6.1 Notes:

Do not uncompress or unzip the compressed .taz archives after download; the installation script will take care of this for you.

If using FTP, be sure to transfer the compressed .taz archives in BINARY mode, and the installation shell script in TEXT mode.

You set your FTP program in TEXT or ASCII transfer mode by typing the following command at your FTP command prompt:

```
ftp> ascii
```

You set your FTP program in BINARY transfer mode by typing the following command at your FTP command prompt:

```
ftp> bin
```

6. The file "install.sh" is an "sh" shell script. It must be executed in the appropriate shell to avoid unexpected behaviour. Run the "install.sh" script on the server by typing the command:

   ```
   sh ./install.sh
   ```
7. The install script will extract the files from the ".taz" files and then create all of the relevant OpenLink server component sub-directories. The install script will also place the OpenLink server components into their appropriate sub-directories
8. Once the files have been extracted and placed in the relevant sub-directories, the install script will proceed to install and configure the OpenLink Admin Assistant. This program enables remote configuration for all OpenLink Server Components (Rule Book, Service and Database Agents) from any Web Browser.

   The dialogue below illustrates the interactions encountered during the server components installation process, accepting all the script defaults:

```
bash$ sh install.sh
$ sh install.sh
*
* Temporarily saving your original Session Rules Book
* (the file oplrqb.ini) to oplrqb.ini.save
*
* Temporarily saving your original ODBC DSN settings
* (the file odbc.ini) to odbc.ini.save
*
* Temporarily saving your original ODBC Installation settings
* (the file odbcinst.ini) to odbcinst.ini.save
*
* Temporarily saving your original UDBC DSN settings
* (the file udbc.ini) to udbc.ini.save
*
Extracting (dqadp91b.taz) ...
Extracting (dqbrzzzz.taz) ...
* Restoring your original Session Rules Book
* (the file oplrqb.ini) and creating a new oplrqb.ini.sample
*
* Restoring your original ODBC DSN settings
* (the file odbc.ini) and creating a new odbc.ini.sample
*
* Restoring your original ODBC Installation settings
* (the file odbcinst.ini) and creating a new odbcinst.ini.sample
*
* Restoring your original UDBC DSN settings
* (the file udbc.ini) and creating a new udbc.ini.sample
*
Setting up CLASSPATH for Java connectivity.
You can choose between the following Java JDK versions:
 1)     JDK 1.1.x
 2)     JDK 1.2.x
 3)     JDK 1.3.x
Please enter your choice [ENTER=2] :
Creating scripts ...
Entering default values in oplrqb.ini ...
```

```
Using Informix 7.3 Multi Threaded Agent (inf73_mv) ...
Using Ingres II Multi Threaded Agent (ingii_mv) ...
Using ODBC Multi Threaded Agent (odbc_mv) ...
Using Oracle 8.1.x Multi Threaded Agent (ora81_mv) ...
Using Virtuoso Multi Threaded Agent (virt_mv) ...
Modifying bin/odbc.ini ...
Modifying bin/odbcinst.ini ...
Creating link for ODBC ...
Enter the port number the broker will listen on for
client connections [ENTER=Default]  :
Welcome to the OpenLink Admin Assistant Setup.
This program will install the HTTP based OpenLink Admin Assistant, thereby
enabling remote configuration for all OpenLink Server Components (Rule Book,
Service and Database Agents) from any Web Browser.
TCP/IP Port to use? [ENTER=8000] :
Log File? [ENTER=www_sv.log] :
Log all requests (y/n)? [ENTER=n] :
Administrator account? [ENTER=admin] :
Administrator's password? [ENTER=admin] :
The OpenLink Admin Assistant is now ready for use.
Please (re)start the OpenLink Request Broker (using the command 'oplrqb -v')
and then point your Web Browser to the URL below:
http://digitalunix5.mydomain.com:8000
Enter the name of the user that will own the
programs [ENTER=Use Current User Settings]  :
Enter the name of the group that will own the
programs [ENTER=Use Current Group Settings] :
Changing ownership ...
End of installation
```

9. Run the OpenLink environment shell script as follows:

   ```
   . openlink.sh
   ```

   Users in tcsh or csh must use the command:

   ```
   source openlink.csh
   ```

   This will set all the required environment variables (e.g., $ODBCINI, $ODBCINSTINI, $PATH) which enable alternate directory executable launching, and proper use of the HTTP-based Admin Assistant.

   You may wish to place the startup command in your shell startup script (.profile, .login, .bashrc etc) in order for it to take effect on login.
10. Start your OpenLink Request Broker in debug mode by executing the OpenLink Request Broker startup command:

   ```
   oplrqb -dv
   ```

   Should you receive an "oplrqb: unable to create listener (Address already in use)" error, this most likely indicates that the Broker is already started. It can be shut down by running the command:

   ```
   oplshut -f
   ```

   or

   ```
   oplshut +port <alternate port number>
   ```

   Where <alternate port number>> is a different port number you specified as the Broker's TCP listen port at install-time. (If you used the default port, then use the former command.)

   If you continue to receive the error above, the port (either 5000, or the one you specified as the Listen port) is already occupied by another TCP process. Contact your systems administrator for assistance, or reinstall the OpenLink Request Broker, specifying a different TCP Listen port.
11. Start your Web Browser and then enter the following URL:

   ```
   http://host:port
   ```

Where *host* is the network alias or IP address of machine where Server Components are installed. The *port* is the number you provided when prompted by OpenLink installer.

For example, if your machine has a network alias of "opllinux", and you accept the default port number at installation time, then the required URL would be constructed as follows:

```
http://opllinux:8000
```

12. Follow the instructions provided in the "Client Components Installation" section of this guide, in order to verify connectivity to your server using the OpenLink Multi-Tier Data Access Drivers.
13. If the previous step is successful, shut down the Request Broker using the command:

```
oplshut -f
```

or

```
oplshut -fy
```

This will kill off any remaining connections to the database, if they have been established. Next restart in normal mode using the command:

```
oplrqb -v
```

Running your Request Broker in debug mode will log a significant amount of information to the bin/oplrqb.log file, so normal mode is recommended unless you are experiencing problems which you need to report to Technical Support.

For more information about other available options when starting and stopping the Request Broker, see the "Request Broker Startup & Shutdown" topic in the Server Components Administration section of this guide.

8.7 Installed Files & Locations

The core components required by the Request Broker and Database Agents are situated within the "bin" and "bin/w3config" sub-directories under your OpenLink server components base installation directory. Each of these components is described below and grouped by directory location.

On Windows, the default base installation directory is `C:\Progam Files\OpenLink\`. On Mac OS X, the non-configurable base installation directory is `//Library/Application Support/openlink/`.

8.7.1 The "bin" sub-directory:

Below is a list of the important files in the bin directory:

Table 6.2. Important Files in the bin directory

File	Description
oplrqb	The OpenLink Request Broker.
oplrqb.ini	The OpenLink "Session Rules" Book.
oplrqb.log	File that holds critical Broker & Startup and Shutdown audit information. All critical events that affect the Broker are written to this file irrespective of Broker startup options.
oplshut	Utility for showing Agent status and shutting down the Broker.
xxx_sv	OpenLink Agent (single thread). xxx will be an abbreviation for the data access or protocol handling service provided by the OpenLink Agent.
xxx_mv	OpenLink Agent (mutli thread). xxx will be an abbreviation for the data access or protocol handling service provided by the OpenLink Agent.
release.txt	Text file with the latest information regarding the Request Broker.
oplrqb.lic	File containing software activation and license key for the Broker and Database agents.

8.7.2 The "bin/w3config" sub-directory:

Table 6.3. Important Files in the bin/w3config directory

File	Description
www_sv	The OpenLink Web Service Agent, this is basically a HTTP/Web Server implemented as an OpenLink Service Providing Agent. It is this component that forms the core engine around which the OpenLink Admin Assistant has been built.
www_sv.log	File that holds critical Web Service Agent & Startup and Shutdown audit information. All critical events that affect the Web Service Agent are written to this file irrespective of Web Service Agent startup options.
www_sv.ini	Web Service Agent configuration file.
setup	Web Service Agent installation and configuration program (you only need to run this directly if you only want to re-install the Web Service Agent component as opposed to the entire pool of OpenLink Server Components.

8.8 Obtaining Component Details From The Command Line

This approach provides you with information about the actual executable file, it includes:

Version Number - this is a component identifier that indicates the version number specific of a specific OpenLink Component

Release Number - this is an identifier for a collection of OpenLink Components, numerous components with different version numbers make up an OpenLink Data Access Drivers commercial release.

Compilation Date - indicates the date component was compiled.

CVSID - this is a source code archive identifier that relates to the actual source code archive from which a particular component has been assembled.

Binary Platform - indicates what platform the component has been built to run on.

To obtain the information referred to above for any OpenLink Component simply type in the program name at your command prompt with the --help switch.

Examples of the output produced by the OpenLink Request Broker and the OpenLink Web Service Agent are depicted below:

8.8.1 OpenLink Request Broker:

```
[person@linuxbox bin]$ ./oplrqb --help
OpenLink Request Broker
Version 2.22 (Release 4.2) as of Mon Apr 08 2002 (cvsid 00060).
Compiled for Linux 2.4.18-xfs (i686-pc-linux-gnu)
Copyright (C) OpenLink Software.
Usage:
  oplrqb [-flLdcv] [+foreground] [+loglevel num] [+logfile arg] [+debug]
         [+configfile arg] [+version]
  +foreground   run in the foreground
  +loglevel     log level
  +logfile      log file
  +debug        debug mode
  +configfile   use alternate configuration file
  +version      show version number
```

8.8.2 Web Service Agent:

```
[person@linuxbox w3config]$ www_sv --help
OpenLink Web Service Agent
Version 1.5 (Release 4.2) as of Mon Apr 08 2002 (cvsid 00060).
Compiled for Linux 2.4.18-xfs (i686-pc-linux-gnu)
Copyright (C) OpenLink Software.
Usage:
```

```
         www_sv [-clLd] [+config arg] [+loglevel num] [+logfile arg] [+debug]
         +config     config file
         +loglevel   log level
         +logfile    log file
         +debug      debug mode
       Copyright (c) 1987-1994 The Regents of the University of California.
       Copyright (c) 1994-1996 Sun Microsystems, Inc.
       Copyright (c) 1995 The Apache Group.  All rights reserved.
        This product includes software developed by the Apache Group
        for use in the Apache HTTP server project (http://www.apache.org/).
```

8.8.3 See Also:

Using the Admin Assistant section for obtaining General Information using the web interface.

8.9 Configuring a Firewall for use with UDA Multi-Tier

A multi-tier installation requires the following firewall considerations:

On the server on which the Request Broker runs, you must allow your clients to connect to port 5000/tcp (or the value of the "Listen" directive in oplrqb.ini).

On the server, you must allow clients to connect to ports 5001/tcp et seq (the values between the "PortLow" and "PortHigh" directives in oplrqb.ini).

On the client, you must be able to make new outgoing connections to the server to contact the request broker (the same 5000/tcp or the value of the "Listen" directive), and to agents listening on successively higher-numbered ports (in the range between "PortLow" and "PortHigh").

If the agent makes a network connection to the database server, the respective port should also be opened on the database server (e.g. 5432/tcp for postgres, 1433 and 1434 udp and tcp for SQL Server, etc).

If you have a stateful firewall, the return packets from request-broker and database agents to the clients should be taken care of automatically. If not, you will have to cater for your OS' selection of ephemeral client-side ports as well.

9 Chapter 7. Request Broker Administration

Abstract
Guide for successfully installing and running the OpenLink server components from the OpenLink Multi Tier Data Access Suite.

Table of Contents

9.1 Request Broker Startup & Shutdown

9.1.1 Startup Options

The Request Broker can be started in a number of ways using various command line options. A single instance is described here. For multiple instances, see a subsequent section.

```
Usage:
  oplrqb [-flLdcv] [+foreground] [+loglevel num] [+logfile arg]
        [+debug][+configfile arg] [+version]
  +foreground   run Broker in the foreground mode
  +loglevel     log level where Broker session log details decrease in verbosity from 7 down to 1
  +logfile      full path and file name into which Broker session log output is to be written
  +debug        starts Broker in debug mode
  +configfile   use alternate configuration file
  +version      show version number
```

9.1.1.1 Command Line Examples:

```
oplrqb
```

Starts the Broker in background mode.

```
oplrqb -v
```

Starts the Broker in background mode and displays version information as part of startup process.

```
oplrqb -dv
```

Starts the Broker in foreground debug mode using default log level 7 (most amount of session log information). This occupies your terminal session in the foreground, and echoes the output of all OpenLink Data Access Clients to the terminal window.

```
oplrqb -dvl 1
```

Starts the Broker in foreground debug mode using default log level 1 (lowest amount of session log information). This occupies your terminal session in the foreground, and echoes the output of all OpenLink Data Access Clients to the terminal window.

```
oplrqb -dvL /tmp/openlink.log
```

Starts the Broker in foreground debug mode using default log level 7 (most amount of session log information). This occupies your terminal session in the foreground, and echoes the output of all OpenLink Data Access Clients to the file "/tmp/openlink.log" .

9.1.2 Shutdown Options

The OpenLink Request Broker can be shut down in various ways, using the available command line options.

```
Usage:
  oplshut [-hcskryfpP] [+hostname arg] [+configfile arg] [+show] [+kill]
          [+reinit] [+yes] [+force] [+protocol arg] [+port num] [pid ..]
  +hostname     contact oplrqb on given host
  +configfile   use alternate configuration file
  +show         show database agent connections
  +kill         kill a database agent
  +reinit       oplrqb must reread the rulebook
  +yes          don't ask for confirmation
  +force        force immediate shutdown (kill all)
  +protocol     specify protocol to use
  +port         specify port to use
Examples:
  oplshut            Shutdown broker on local machine (when idle)
  oplshut -f         Shutdown broker without waiting for clients to terminate
  oplshut -fy        Same as -f, but without asking for confirmation
  oplshut -s         Show the current active connections
  oplshut -k 81 93   Try to terminate the database agents with svpid 81 and 93
```

9.1.2.1 Command Line Examples:

```
oplshut
```

Shuts down the Request Broker on the local machine. When a connection is still active, and the user calls oplshut without additional arguments, the broker will enter a gracefull shutdown state which means that it will not accept any new connections, but remains active until all active connections are terminated.

The shutdown state will be reported on the foreground screen or broker log:

```
oplrqb: shutdown mode initiated by openlink@127.0.0.1
```

Any client that tries to connect to the broker will get the following error message:

```
odbctest DSN=ora8i
OpenLink ODBC Demonstration program
This program shows an interactive SQL processor
[OpenLink][ODBC]the broker is being shut down, SQLSTATE=08004
[OpenLink][ODBC]Connection rejected by data source, SQLSTATE=08004
```

```
oplshut -f
```

Shuts down the Request Broker on the local machine without waiting for clients to terminate.

If you use the -f (or force) flag, the broker will kill all active connections regardless what the client application is doing, which can lead to problems if the applications are not written to handle such events. Depending on the database, open transactions will be rolled back etc.

Many databases like Oracle and Progress implement the same concept where the gracefull shutdown option is the default, but a shutdown can be forced in case of a big calamity, or when a connection remains lingering.

```
oplshut -fy
```

Same functionality as prior command, but without asking for confirmation.

```
oplshut +port 16000
```

Shuts down the Request Broker running on the TCP Listen port of "16000". Note that the default Request Broker Listen port is 5000, thus this option is not necessary unless you specified a different Listen port at install-time.

```
oplshut -s
```

Shows the current active OpenLink Data Access Client connections.

```
oplshut -k 81 93
```

Try to terminate the database agent sessions with process IDs 81 and 93.

```
oplshut -h 12.34.56.78
```

Shuts down the Request Broker on remote machine with IP address "12.34.56.78".

9.1.3 Security

Due to the Unix security model certain kernel calls can only be used when the process is run by root or under the permission of root. One of these function calls is needed for the PingWatchDog protocol, thereby forcing you to run the request broker as root. This can be accomplished in two ways:

1. The user logs in as root and then fires up the request broker.
2. The request broker is flagged to run under root privileges and the user can start it up from other accounts.

The consequences of 1 are that the user(s) who need to startup the broker need to have knowledge of the root password, which is not advisable for system managers to give to ordinary users.

Therefore we have built a solution whereby the broker can be flagged as running under root privileges without the user(s) needing to know the root password. The system manager executing the following commands accomplishes this:

```
cd openlink/bin
 ./security -s
```

This will place the broker into a special security state that has the following impact:

1. The permissions of the oplrqb program are changed so it takes on the identity of "root" when started up.
2. The broker now knows the position of the oplrqb.ini file and will not accept certain startup flags so it cannot be tricked to read in another oplrqb.ini file from the command line.
3. The oplrqb.ini file can now only be written to by the "root" account, other accounts will only be able to read this file.

If a user now starts the broker the StartupBy clause within the [Security] section of the broker is checked to see whether the user is allowed to startup the broker (same for shutdown).

This accomplishes the wish of many system administrators that the broker process can be started up as root, without widespread knowledge of the root password.

As stated in item 3 above the "oplrqb.ini" file can now only be written to by the "root" account. This means that ordinary users cannot modify this file, which often seems like an unpleasant side effect. To overcome this some system administrators are temped to modify the mode of the oplrqb.ini file, thereby opening up to security risks. This should be avoided, the *IncludeRuleBook* parameter inside the *Security* sections should be used instead to divide the rulebook in to parts for others to manage for themselves. The system administrator can split up the rulebook into different files which than can be included from the master oplrqb.ini file. This makes specific sections available to normal users, but allowing the system administrator to decide which sections can/cannot be overruled by normal users. The included rulebooks cannot overrule any section within the oplrqb.ini file. This is a very powerful way of dealing with high security installations within large companies.

9.1.4 Multiple Request Brokers

There are now three different behaviors for running the broker under Windows that are affected by the type of Windows operating system, and the +foreground and +debug flags.

9.1.4.1 METHOD 1 - As a service

When:

- Under NT/XP/2000, when no +foreground or +debug has been specified. The oplrqb service will start, but only if it has been registered. When no +instance is specified, it will use the default instance. If this fails, it will run as method 2 or 3.
- Under NT/2000, with a +service start [+instance identifier] If this fails, it will exit with an error.

9.1.5 Note 1:

When starting oplrqb without options, it will attempt to auto start an existing service. if that fails, it will attempt methods 2 and 3 (see below)

9.1.6 Note 2:

You'll need to do +service create first, the broker doesn't do this by itself.

9.1.6.1 METHOD 2 - With debugging console

When:

- with +foreground or +debug, started from explorer/dos box
- without +foreground or +debug option, started from a dos box.

9.1.7 Note:

Start up mode is different when launched from a dos box. If there are no command line options specified, the broker will now default to +debug when started in a dos box.

the +foreground and +debug options take precedence over method 1 (starting the service)

9.1.7.1 METHOD 3 - User Specific Service (characterized by a Services Tray Icon)

When:

- started from explorer, without +foreground/+debug

  (NT/XP/2000: Method 1 is tried first)

9.1.8 Note:

It will attempt to locate oplrqb.ini in the current directory, unless +config file has been specified.

9.1.8.1 Startup Mode Summary

To summarize, here is some pseudo code:

```
if OS = NT or OS = Win2000 or OS = XP then
  if no +foreground or +debug options then
    if there is a service installed then
      silently attempt to start the service
      if this succeeds then exit
    endif
endif
if +foreground or +debug options then
  allocate a debugging console
```

```
      run in the foreground
    else if started from a dos box then
      set +foreground and +debug
      allocate a debugging console
      run in the foreground
    else
      create the splash window and assume tray behavior
    endif
    Note: it is possible to make the oplrqb service allocate a debugging
    console. To do this, specify +debug while creating the service
    (example: oplrqb +service create +debug)
```

9.1.8.2 Multiple Service Instances

Multiple instances of the broker work only for NT/XP/2000 when running as a service.

There is one default instance (unnamed) which is used when no +instance command line flag has been specified.

examples:

```
oplrqb +service create
```

Registers the default service instance. This service will use the oplrqb.ini rulebook in the current directory

```
oplrqb +service create +instance 1 +config f:\openlink
```

Registers the oplrqb_1 service instance This service will use the oplrqb.ini rulebook in f:\openlink

```
oplrqb +service create +config f:\openlink\test\debug.ini +debug
```

Registers the default service instance. This service will use the debug.ini rulebook in f:\openlink\test This service will produce debug output in a console. This service will not auto start, use oplrqb +service start

Other service commands:

```
  +service auto [+instance id]    Set start up type to Automatic
  +service manual [+instance id]  Set start up type to Manual
  +service start [+instance id]   Start up the service
  +service stop [+instance id]    Stops up the service
  +service delete [+instance id]  Unregisters up the service
```

+service list has been extended to show the rulebook that the instance uses as well as other status information.

To configure another Broker instance manually:

- create a new directory
- create a new rulebook in this directory
- set BinaryDirectory in this new rulebook to the installation directory of oplrqb.exe (need to share binaries & licensing)
- Specify a different Listen port
- Either disable www_sv in the rulebook or create a new www_sv.ini with 'HttpPort' key value set to a unique value other than the default 8000.
- create another entry using +instance <id> +config <the_new_rulebook_ini>

9.2 Request Broker Session Parameters

The Request Broker is responsible for setting up the profile of one of more OpenLink Data Access sessions, this includes a Transport Protocol selection, Keep Alive Packets notifications, Network Message Buffer Sizes. Initialization retry intervals, Agent initialization timeouts, and much more. Rather than have you manually set this options via the OpenLink Rule Book, it is now possible for you to configure this options via your Web Browser of choice using the Admin Assitant

The Admin assistant allows you to configure Request Broker session parameters in two ways, via a series of Wizard Interactions or via an HTML form, the sections that follow illustrate both approaches.

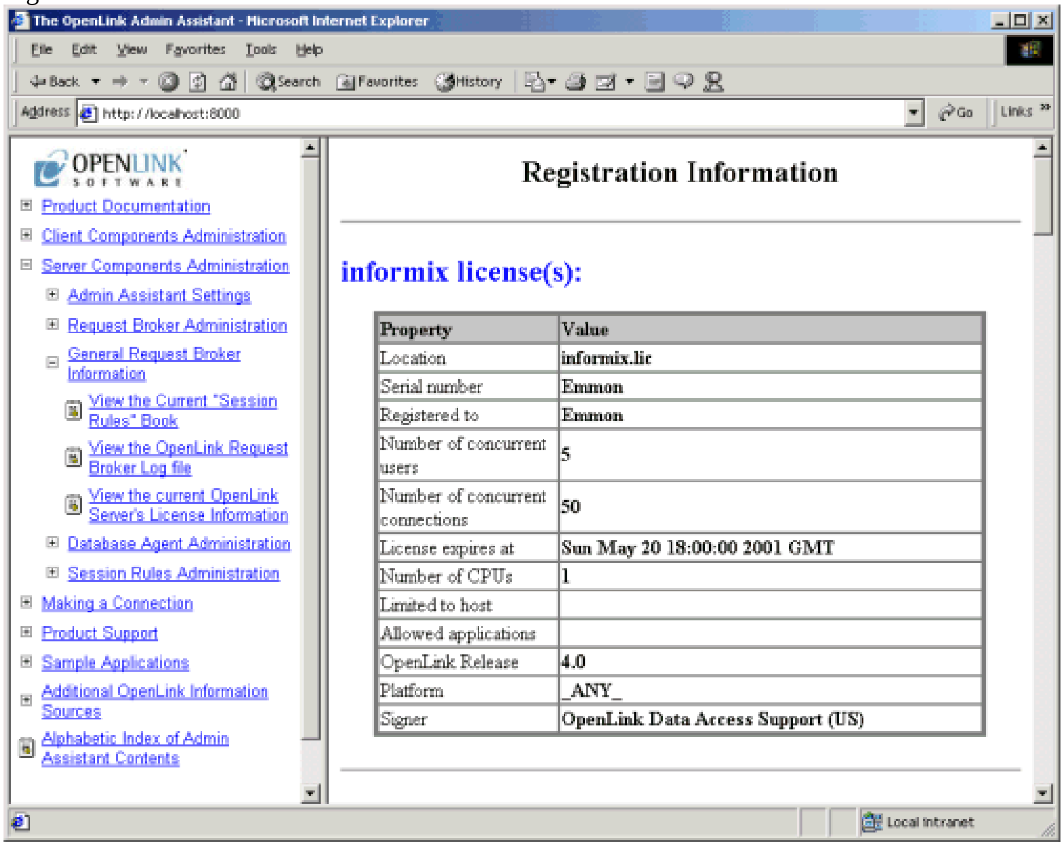
9.2.1 General Information

You obtain additional information using the Admin Assistant, expand the "Server Components Administration" menu item, then "General Request Broker Information", you are offered:

- Register File Information - enables you to determine the license(s) that you have in place.
- Session Rules Book - the current set of rules for your OpenLink Data Access sessions.
- Request Broker Log File Information - displays the contents of your Request Broker log file.

Here is an example of the License Information page:

Figure 7.1. Admin Assistant Wizard - License Information


The OpenLink Admin Assistant - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://localhost:8000
Go Links
OPENLINK SOFTWARE
Product Documentation
Client Components Administration
Server Components Administration
Admin Assistant Settings
Request Broker Administration
General Request Broker Information
View the Current "Session Rules" Book
View the OpenLink Request Broker Log file
View the current OpenLink Server's License Information
Database Agent Administration
Session Rules Administration
Making a Connection
Product Support
Sample Applications
Additional OpenLink Information Sources
Alphabetic Index of Admin Assistant Contents
Local intranet

Registration Information

informix license(s):

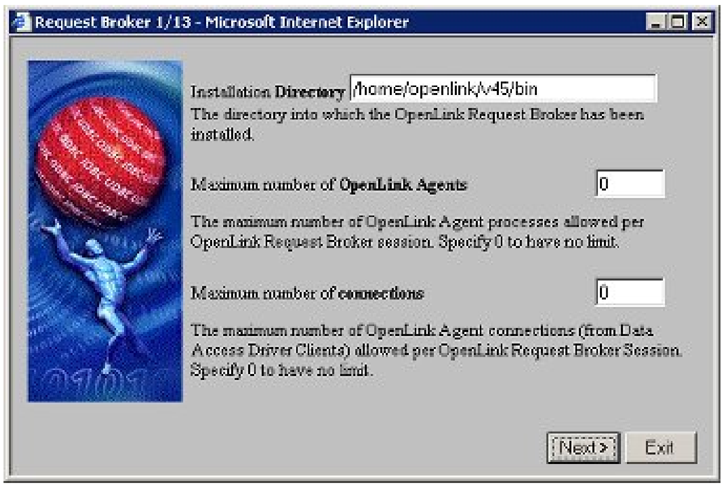
Property	Value
Location	informix.lic
Serial number	Emmon
Registered to	Emmon
Number of concurrent users	5
Number of concurrent connections	50
License expires at	Sun May 20 18:00:00 2001 GMT
Number of CPUs	1
Limited to host	
Allowed applications	
OpenLink Release	4.0
Platform	_ANY_
Signer	OpenLink Data Access Support (US)

9.2.2 Using Admin Assistant Wizards

1. Specify the location of your Request Broker and its dependent components. Also set restrictions on number of Agents, and number of connections.

   Click the "Next" button.

   Figure 7.2. Admin Assistant Wizard



Installation Directory /home/openlink/v45/bin

The directory into which the OpenLink Request Broker has been installed.

Maximum number of OpenLink Agents 0

The maximum number of OpenLink Agent processes allowed per OpenLink Request Broker session. Specify 0 to have no limit.

Maximum number of connections 0

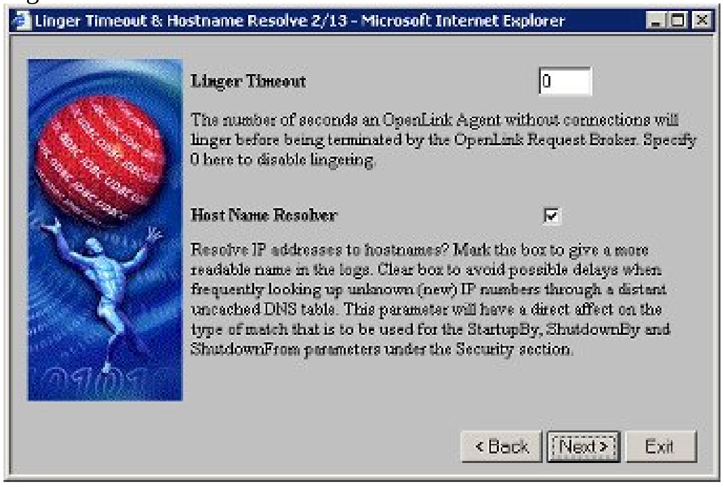
The maximum number of OpenLink Agent connections (from Data Access Driver Clients) allowed per OpenLink Request Broker Session. Specify 0 to have no limit.

2. Linger Timeout: The number of seconds an OpenLink Agent without connections will linger before being terminated by the OpenLink Request Broker, Specify 0 here to disable lingering. Lingering is a feature for avoiding repetative load/unload overhead of agents that are frequently used for short, quick concurrent open/close connections.

   This setting is global to all agents. For specific agent settings, use the LingerTimeout= value in the agent environment section.

   Set Resolving of IP addresses to hostnames. The logging will show the hostname instead of the IP number if this option is selected. Switch off hostname lookup to avoid possible delays when frequently looking up unknown (new) IP numbers through a distant uncached DNS table. This parameter will have a direct affect on the type of match that is to be used for the StartupBy, ShutdownBy and ShutdownFrom parameters under the Security section.

   Click the "Next" button.

   Figure 7.3. Admin Assistant Wizard
   
   Linger Timeout & Hostname Resolve 2/13 - Microsoft Internet Explorer

   Linger Timeout 0

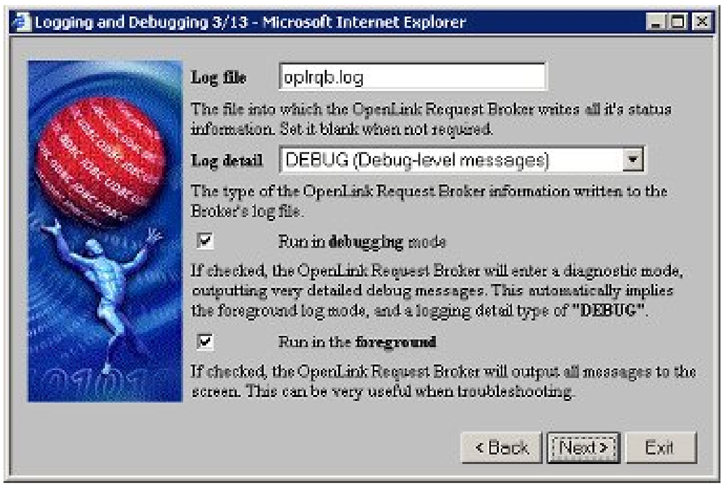
   The number of seconds an OpenLink Agent without connections will linger before being terminated by the OpenLink Request Broker. Specify 0 here to disable lingering.

   Host Name Resolver ☑

   Resolve IP addresses to hostnames? Mark the box to give a more readable name in the logs. Clear box to avoid possible delays when frequently looking up unknown (new) IP numbers through a distant uncached DNS table. This parameter will have a direct affect on the type of match that is to be used for the StartupBy, ShutdownBy and ShutdownFrom parameters under the Security section.

3. To help with troubleshooting, turn on session logging and extra information. Or run the broker in foreground mode to view the trace. Then click on the "Next" button.

   Figure 7.4. Admin Assistant


Logging and Debugging 3/13 - Microsoft Internet Explorer

Log file oplrqb.log

The file into which the OpenLink Request Broker writes all it's status information. Set it blank when not required.

Log detail DEBUG (Debug-level messages)

The type of the OpenLink Request Broker information written to the Broker's log file.

☑ Run in **debugging** mode

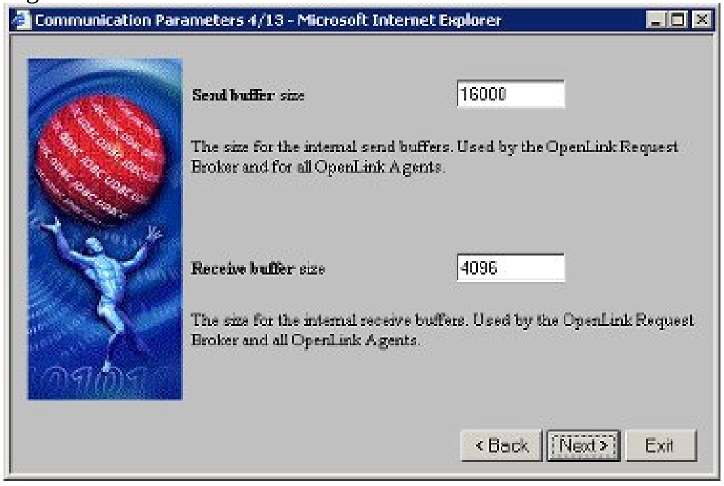
If checked, the OpenLink Request Broker will enter a diagnostic mode, outputting very detailed debug messages. This automatically implies the foreground log mode, and a logging detail type of **"DEBUG"**.

☑ Run in the **foreground**

If checked, the OpenLink Request Broker will output all messages to the screen. This can be very useful when troubleshooting.

4. Enter buffer size values for the inbound and outbound network messages for each OpenLink Agent (it is advisable to take the default values presented). Then click the "Next" button.

Figure 7.5. Admin Assistant


Communication Parameters 4/13 - Microsoft Internet Explorer

Send buffer size 16000

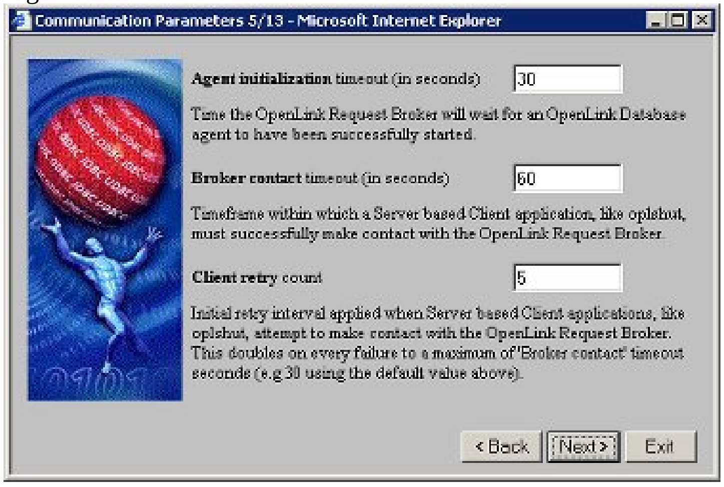
The size for the internal send buffers. Used by the OpenLink Request Broker and for all OpenLink Agents.

Receive buffer size 4096

The size for the internal receive buffers. Used by the OpenLink Request Broker and all OpenLink Agents.

5. Enter values for "Database Agent" session initialization, "Broker Contact", and initialization failure retry counts by entering values into the respective fields as depicted below (it is advisable take the defaults). Then click on the next button.

Figure 7.6. Admin Assistant


Communication Parameters 5/13 - Microsoft Internet Explorer

Agent initialization timeout (in seconds) 30

Time the OpenLink Request Broker will wait for an OpenLink Database agent to have been successfully started.

Broker contact timeout (in seconds) 60

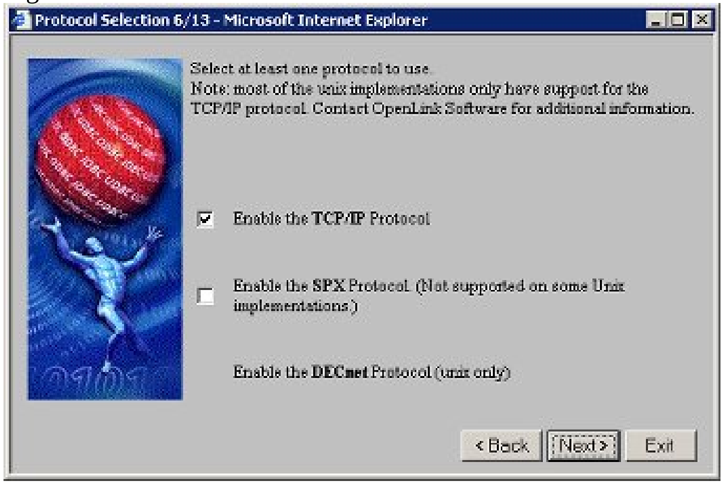
Timeframe within which a Server based Client application, like oplshut, must successfully make contact with the OpenLink Request Broker.

Client retry count 5

Initial retry interval applied when Server based Client applications, like oplshut, attempt to make contact with the OpenLink Request Broker. This doubles on every failure to a maximum of 'Broker contact' timeout seconds (e.g 30 using the default value above).

6. Check the related boxes presented in order to select the Network Transport Protocol to be used by your Request Broker and Database Agents when communicating with your OpenLink Data Access Clients. Then click on the next button.

Figure 7.7. Admin Assistant



Select at least one protocol to use.
Note: most of the unix implementations only have support for the TCP/IP protocol. Contact OpenLink Software for additional information.

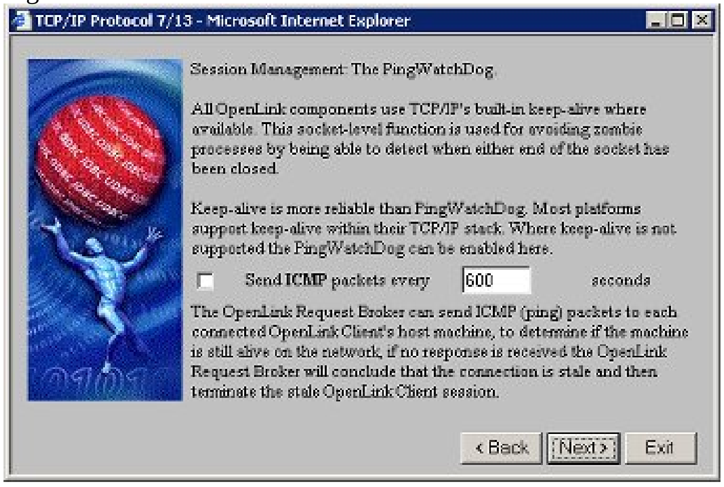
[✓] Enable the **TCP/IP** Protocol

[ ] Enable the **SPX** Protocol. (Not supported on some Unix implementations.)

Enable the **DECnet** Protocol (unix only)

7. You can choose to enable the OpenLink Ping Watchdog. All OpenLink components automatically use the keep-alive feature built in to most current implementations of TCP/IP stack. If keep-alive is not available the Ping WatchDog can be enabled in attempt to compensate for this. Click the "Next" button to continue.

Figure 7.8. Admin Assistant



Session Management: The PingWatchDog

All OpenLink components use TCP/IP's built-in keep-alive where available. This socket-level function is used for avoiding zombie processes by being able to detect when either end of the socket has been closed.

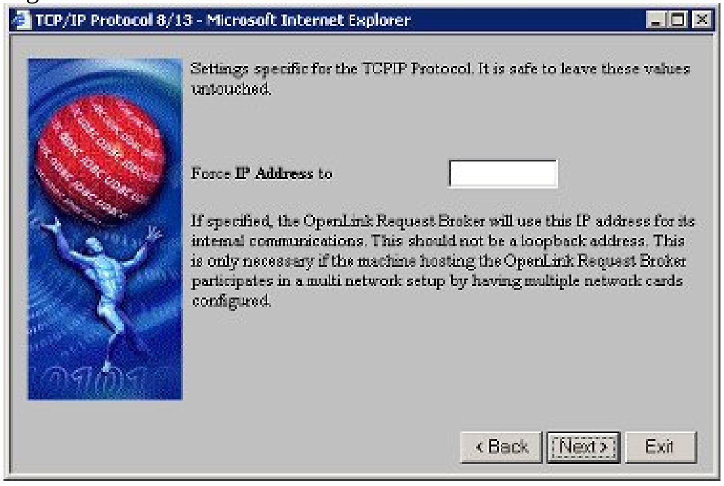
Keep-alive is more reliable than PingWatchDog. Most platforms support keep-alive within their TCP/IP stack. Where keep-alive is not supported the PingWatchDog can be enabled here.

[ ] Send **ICMP** packets every 600 seconds

The OpenLink Request Broker can send ICMP (ping) packets to each connected OpenLink Client's host machine, to determine if the machine is still alive on the network, if no response is received the OpenLink Request Broker will conclude that the connection is stale and then terminate the stale OpenLink Client session.

8. If you have more than one Network Interface Card (NIC) installed on your OpenLink Server machine, then you can enter a value in the "Force IP Address" field that represents the IP address that you want the Request Broker to listen at.

Figure 7.9. Admin Assistant



Settings specific for the TCPIP Protocol. It is safe to leave these values untouched.

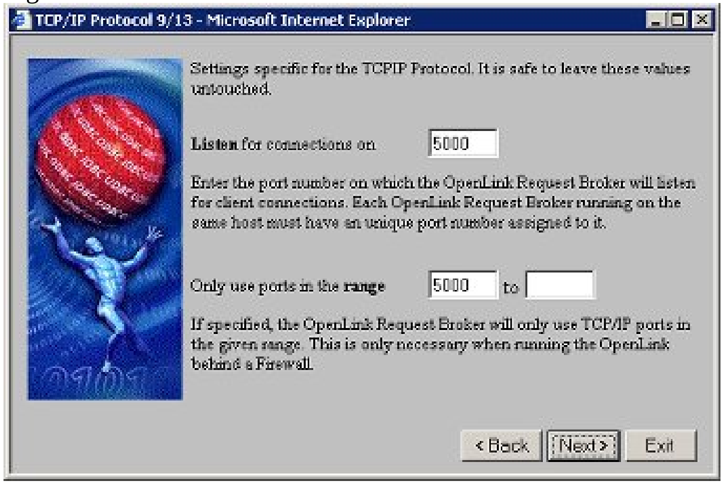
Force IP **Address** to

If specified, the OpenLink Request Broker will use this IP address for its internal communications. This should not be a loopback address. This is only necessary if the machine hosting the OpenLink Request Broker participates in a multi network setup by having multiple network cards configured.

9. Enter the port number on which the OpenLink Request Broker will listen for client connections. Each OpenLink Request Broker running on the same host must have a unique port number assigned to it.

You can also designate port number ranges to be used by the Broker to advertise other OpenLink services (e.g JDBC Agent used by OpenLink Drivers for JDBC). Once completed, click the "Next" button to continue.

Figure 7.10. Admin Assistant



Settings specific for the TCPIP Protocol. It is safe to leave these values untouched.

Listen for connections on 5000

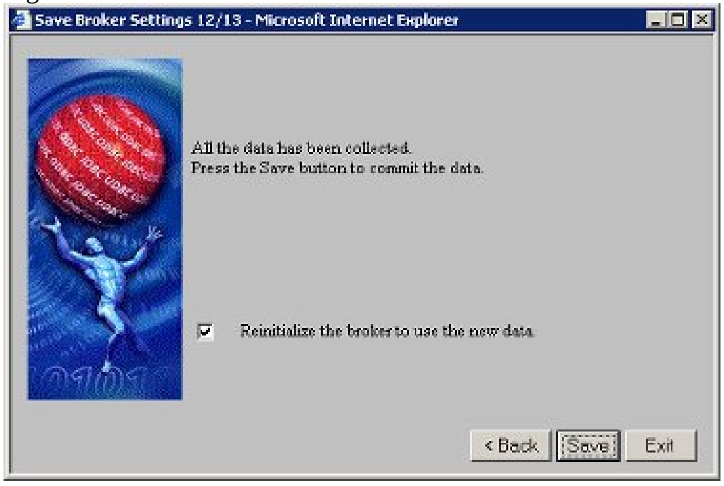
Enter the port number on which the OpenLink Request Broker will listen for client connections. Each OpenLink Request Broker running on the same host must have an unique port number assigned to it.

Only use ports in the **range** 5000 to

If specified, the OpenLink Request Broker will only use TCP/IP ports in the given range. This is only necessary when running the OpenLink behind a Firewall.

10. Click on the "Save" button to save your changes permanently.

Figure 7.11. Admin Assistant

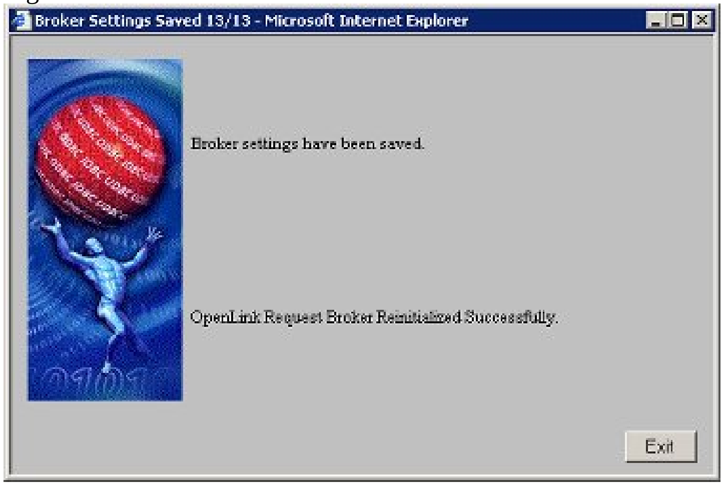


All the data has been collected.
Press the Save button to commit the data.

[✓] Reinitialize the broker to use the new data.

11. Click the "Exit" button to leave the Broker Settings Wizard.

Figure 7.12. Admin Assistant



Broker settings have been saved.

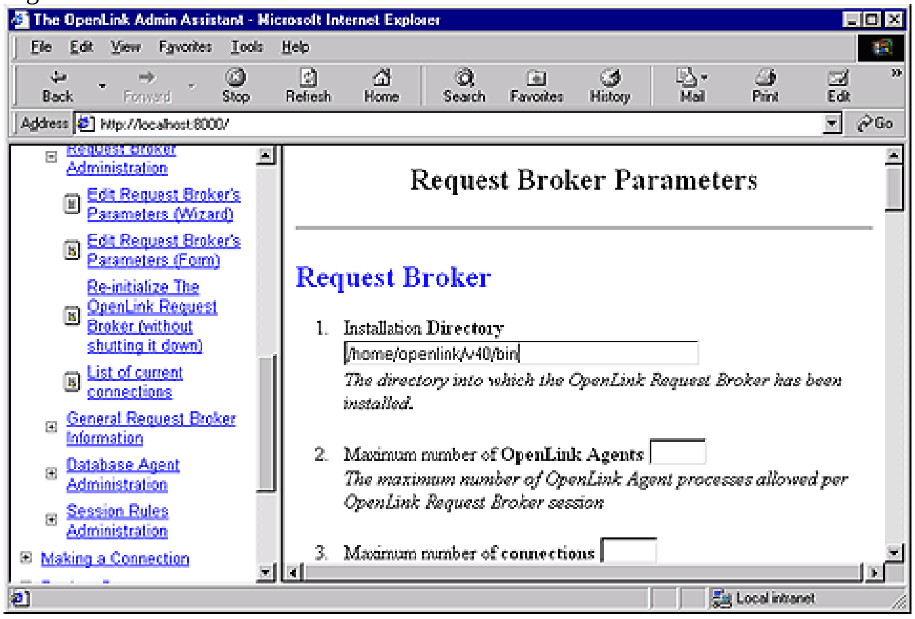
OpenLink Request Broker Reinitialized Successfully.

9.2.3 Using Admin Assistant Forms

You can also choose to configure your Request Broker's session settings via a Forms as opposed to Wizard based interface. You select this option from the Admin Assistant and you will be presented with a screen similar to the snapshot below.

Each Broker session settings field contains a description of the values that need to be entered into each of the forms fields. In addition a brief description of the use of each of these values is provided below each field.

Figure 7.13. Admin Assistant


The OpenLink Admin Assistant - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail Print Edit
Address http://localhost:8000/
Go
Request Broker Administration
Edit Request Broker's Parameters (Wizard)
Edit Request Broker's Parameters (Form)
Re-initialize The OpenLink Request Broker (without shutting it down)
List of current connections
General Request Broker Information
Database Agent Administration
Session Rules Administration
Making a Connection
Local intranet

Request Broker Parameters

Request Broker

1. Installation Directory
/home/openlink/v40/bin
The directory into which the OpenLink Request Broker has been installed.

2. Maximum number of OpenLink Agents
The maximum number of OpenLink Agent processes allowed per OpenLink Request Broker session

3. Maximum number of connections

9.3 OpenLink Session Rules Administration & Configuration Guide

One of the most important features and benefits of your OpenLink Multi Data Access Drivers, is the ability to configure and control your entire OpenLink infrastructure from a central point using "OpenLink Session Rules". These session rules are stored and maintained in a text based repository called the "Session Rules Book" (the file "oplrqb.ini").

You administer these rules from your web browser using the OpenLink Admin Assistant. The rules that you create are enforced by the OpenLink Request Broker, giving you phenomenal control over your distributed computing infrastructure.

9.3.1 OpenLink Session Rules Concepts

Session rules are declarative in nature and template driven. You build a template that determines how one or more OpenLink Client components are going to interact with a particular instance of an OpenLink Server component (Database Agent or Service Provider Agent). Session rules also determine what OpenLink Server is instantiated and how it is to be instantiated for a particular OpenLink Client.

The basic Session Rule unit is an OpenLink Connection Attribute, each representing a key aspect of an OpenLink Client's connection to an OpenLink Agent Session. There are six OpenLink Connection Attributes: Domain, User, OpSys, Machine, Application and Mode.

9.3.2 OpenLink Connection Attributes

There are two types of OpenLink Connection Attributes, these are "User Configurable" and "Non User Configurable" Connection Attributes.

9.3.2.1 User Configurable Connection Attributes

These attributes are configurable by the OpenLink Client user. When using ODBC this is handled via the ODBC Administrator or via the Admin Assistant ODBC Data Source Name Configuration Menu (this also applies to UDBC

Clients). JDBC handles this through the use of JDBC URLs and in the case of JDBC 2.0 via Data Source Names. OLE-DB handles this via a Connection String Building Wizard.

9.3.2.1.1 Domain Attribute

Identifies the OpenLink Agent to which all OpenLink Connection Attributes apply.

This attribute describes a logical reference for a database type, OpenLink agent type, or anything else you would like to use as a logical identifier for all the other OpenLink Connection Attributes.

This attribute is also referred to as the "Server Type" or "SVT" attribute of an OpenLink Client connect string. It is also referred to in older product documentation as the OpenLink "Provider Type".

User Attribute. Identifies the User making use of an OpenLink Client.

Database Attribute. Identifies a specific database name within a database environment, e.g ORCL within Oracle, "stores7" within Informix, "pubs" with Sybase or Microsoft SQL Server etc. The values associated with this Connection Attribute aren't overridden by Aliases, instead they are overridden by the "Database" attribute within an Agent Configuration template.

ServerOpts (Database Sever Environment Options) Attribute. Identifies a set of database environment initialization parameters, currently this attribute only applies to Progress environments. It is used to set self serving client initialization parameters such as: -TB, -TB, -q, -D mdy etc. The values associated with this Connection Attribute aren't overridden by Aliases, instead they are overridden by the "SeverOptions" attribute within an Agent Configuration template.

ConnectOpts (Database Server Connection Options) Attribute . Identifies a set of database server connection parameters used to initiate a connection with a backend database server process. This is how an OpenLink Database Agent makes a connection with a backend database server using a particular database vendors networking middleware (e.g. Net8 or SQL*Net for Oracle, I-Connect or I-Net for Informix, Progress Client Networking for Progress, Open Client for Sybase etc.) . The values associated with this Connection Attribute aren't overridden by Aliases, instead they are overridden by the "ConnectOptions" attribute within an Agent Configuration template.

9.3.2.2 Non User Configurable Connection Attributes

These attributes are environmental in nature and derived automatically by an OpenLink Client.

OpSys (Operating System) attribute. Identifies the Client operating system from which the OpenLink Client is being executed.

Machine attribute. Identifies the Network Alias or IP address of the machine or device from the OpenLink Client is being executed.

Application attribute. Identifies the ODBC, JDBC, UDBC, OLE-DB Client Application from which the OpenLink Client is being executed.

9.3.3 Note:

The Application name can be set within the application using the ODBC API SQLSetConnectOption(hdbc, SQL_APPLICATION_NAME, "AppName"). The JDBC equivelent can also be used.

Mode attribute. Identifies the session Mode required by an OpenLink Client, this may be Read-Only or Read-Write.

9.3.4 OpenLink Session Templates

Connection attributes are conditionally post processed during the initialization of session between an OpenLink Client and and OpenLink Agent, the rule book consists of a number of templates that play different roles during this process. The rule book is made up of the following templates: Session Aliases, Mapping Rules, and Agent Configuration

Session Aliases Templates. These are rule book templates used for post processing OpenLink Connection Attribute values prior to Mapping Rules evaluation. This is the facility used by the Request Broker for overriding Connection Attributes from OpenLink Clients with values on configured an OpenLink server.

Mapping Rules Templates. These are rule book templates used to determine which OpenLink Agents are instantiated in line with an OpenLink Client's session request.

OpenLink Agent Configuration Templates. Templates used for setting key OpenLink Agent Configuration parameters. See the OpenLink Agent Administration section for detailed information.

9.3.5 Session Rules Execution Process

When an OpenLink Client makes contact with an OpenLink Request Broker a series of events occur culminating in the identification and execution of a session rule. The step as follow guide you through this process:

1. Request Broker receives session request from one or more OpenLink Clients (Drivers for ODBC, UDBC, JDBC, OLE-DB).
2. Request Broker parses the session request data stream received from the relevant OpenLink Client isolating each OpenLink Connection Attribute type and associated attribute values.
3. Request Broker then performs a regular expression search through the rule book looking for Session Aliases that match the parsed OpenLink Connection Attributes.
4. For each OpenLink Connection Attribute that the Request Broker find a matching Session Alias it determines if the Alias has a non NULL assigned value, if this is true the OpenLink Connection Attribute values are reassigned to those of the matching Session Alias, otherwise they retain their existing values.
5. Request Broker then performs a regular expression search using a combination of all the OpenLink Connection Attributes across the session rules books "Mapping Rules" template.
6. The Request Broker scans through each "Mapping Rule" in ascending order, If it finds a "Mapping Rule" match it then applies the matching rule to the appropriate Client Session request, otherwise it reports an error condition back to the OpenLink client.
7. When a "Mapping Rules" occurs the Request Broker evaluates the "Mapping Rule". This evaluation results in the Acceptance or Rejection of an OpenLink Client request.
8. OpenLink Client request acceptance results in the OpenLink Clients session request being associated with an OpenLink Agent template, this template then applies all of its attribute Attributes to the OpenLink Agent Configuration process.
9. Session Rules rejection results in a user/administrator definable error message being relayed back to the OpenLink Client.
10. OpenLink Session is fully initialized. This means that the OpenLink Client and Server (agents) components are now linked and operating in a connected state.
11. OpenLink Agent evaluates subsequent OpenLink Client session requests to see if they are in line with the ReUse attribute of its Agent Configuration template.

9.3.6 Creating Custom Aliases For Use By OpenLink Data Access Clients

Understanding how to maintain Session Aliases is a critical part of understanding how to create session rules. You can create, modify, edit, and delete Session Aliases in two ways, you either use the Admin Assistant's GUI interface or manually edit the session rule book using a text editor (only recommended for advanced users).

The steps that follow guide you through the Session Alias management process using the Admin Assistant's GUI. Before performing any of these steps you need to start the Admin Assistant, this is done by following the steps below:

1. Start the Request Broker
2. Start a Web Browser session
3. Enter the following URL into your browser:

If you started the Request Broker on your local machine enter:

```
http://localhost:8000/
```

(assuming you accepted port 8000 as the Admin Assistant port number at installation time).

If the Request Broker in on another machine enter:

```
http://<hostname or IP address>:8000
```

(assuming you accepted port 8000 as the Admin Assistant port number at installation time).

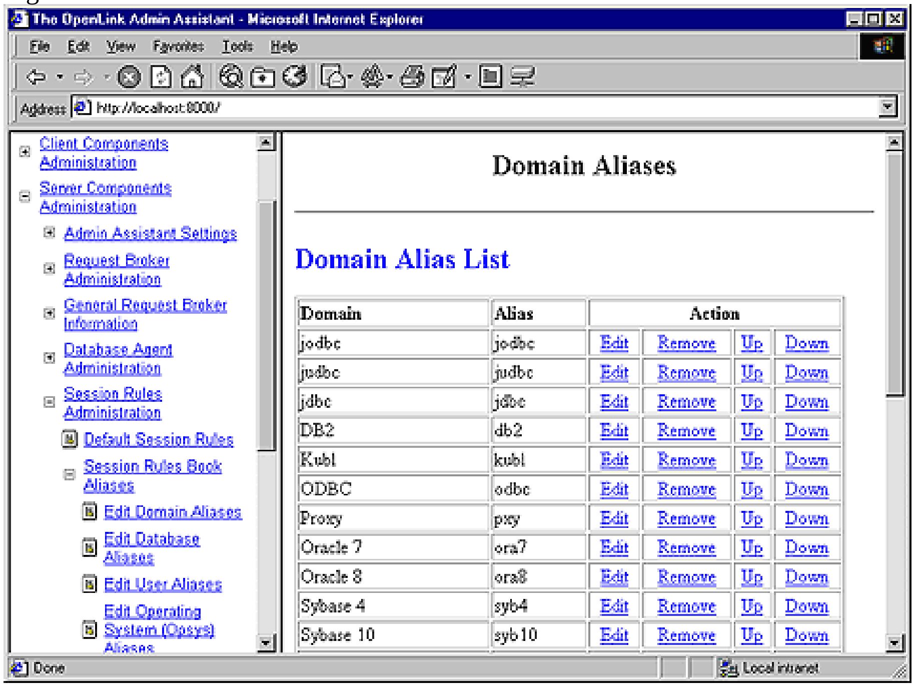
9.3.6.1 Domain Aliases

The steps that follow show you how to manage Domain Aliases:

1. Follow the "Server Components Administration"->"Session Rules Administration"->"Session Rules Book Aliases"->"Edit Domain Aliases" menu tree which brings you to a screen identical to the one depicted below.

   The Admin Assistant presents you with a list of pre-configured Domain Aliases, click on the "Edit" hyperlink to change settings, the "Remove" hyperlink to delete an Alias, and the "Add" button at the bottom of the screen to create a new Domain Alias.

   Figure 7.14. Admin Assistant


The OpenLink Admin Assistant - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://localhost:8000/
Client Components Administration
Server Components Administration
Admin Assistant Settings
Request Broker Administration
General Request Broker Information
Database Agent Administration
Session Rules Administration
Default Session Rules
Session Rules Book Aliases
Edit Domain Aliases
Edit Database Aliases
Edit User Aliases
Edit Operating System (Opsys) Aliases
Done
Local intranet

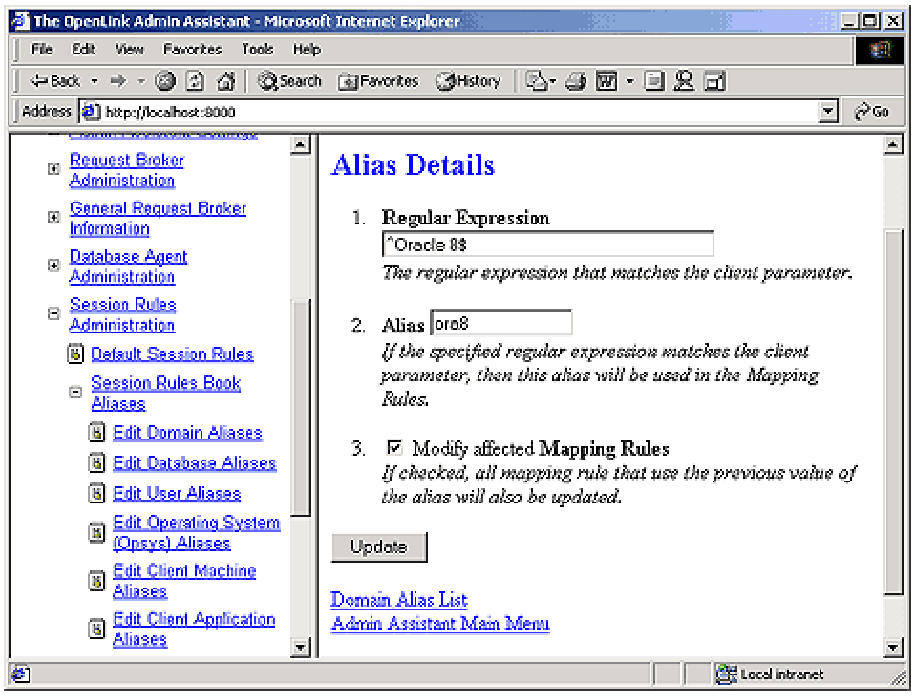
Domain Aliases

Domain Alias List

Domain	Alias	Action			
jodbc	jodbc	Edit	Remove	Up	Down
judbc	judbc	Edit	Remove	Up	Down
jdbc	jdbc	Edit	Remove	Up	Down
DB2	db2	Edit	Remove	Up	Down
Kubl	kubl	Edit	Remove	Up	Down
ODBC	odbc	Edit	Remove	Up	Down
Proxy	pxy	Edit	Remove	Up	Down
Oracle 7	ora7	Edit	Remove	Up	Down
Oracle 8	ora8	Edit	Remove	Up	Down
Sybase 4	syb4	Edit	Remove	Up	Down
Sybase 10	syb10	Edit	Remove	Up	Down

2. The example below assumes that you are modifying a Domain Alias named "ora8" with the attribute values "Oracle 8". This implies that you want to take note of Domain Connection Attributes from an OpenLink Client that start with the value "Oracle 8" evaluation against a mapping rules template. Once you have completed your input click the "Update" button.

   Figure 7.15. Admin Assistant



The OpenLink Admin Assistant - Microsoft Internet Explorer

Address http://localhost:8000

Request Broker Administration
General Request Broker Information
Database Agent Administration
Session Rules Administration
Default Session Rules
Session Rules Book Aliases
Edit Domain Aliases
Edit Database Aliases
Edit User Aliases
Edit Operating System (Opsys) Aliases
Edit Client Machine Aliases
Edit Client Application Aliases

Alias Details

1. Regular Expression ^Oracle 8$
The regular expression that matches the client parameter.

2. Alias ora8
If the specified regular expression matches the client parameter, then this alias will be used in the Mapping Rules.

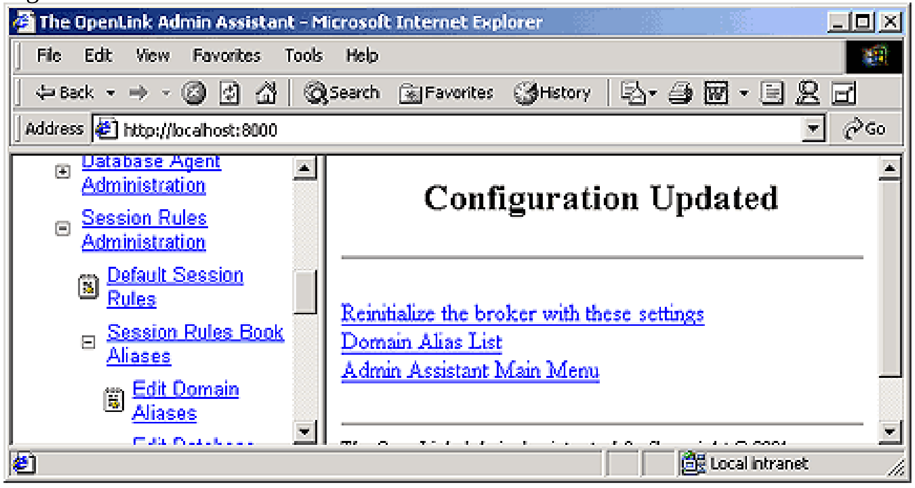
3. ☑ Modify affected Mapping Rules
If checked, all mapping rule that use the previous value of the alias will also be updated.

Update

Domain Alias List
Admin Assistant Main Menu

3. You then commit your changes to the rule book by clicking on the "Reinitialize the OpenLink Request Broker with these settings" hyperlink. See screen shot below:

Figure 7.16. Admin Assistant



The OpenLink Admin Assistant - Microsoft Internet Explorer

Address http://localhost:8000

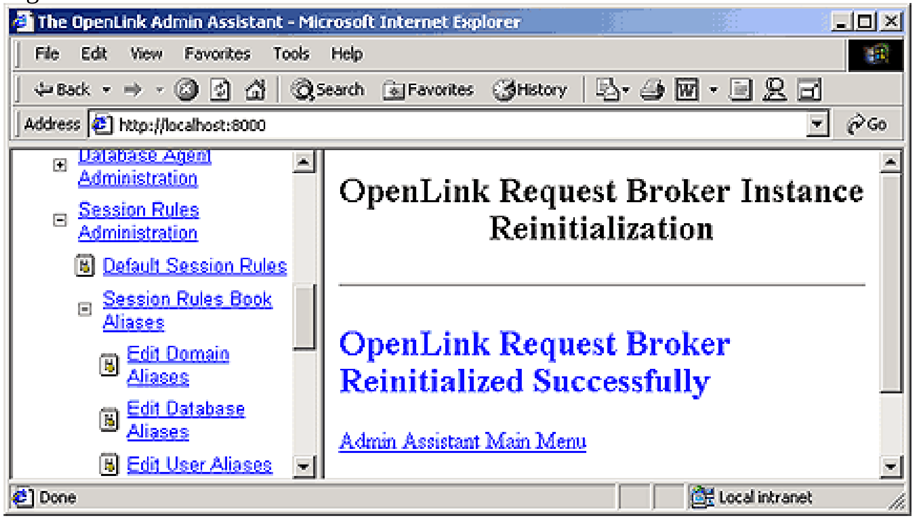
Database Agent Administration
Session Rules Administration
Default Session Rules
Session Rules Book Aliases
Edit Domain Aliases

Configuration Updated

Reinitialize the broker with these settings
Domain Alias List
Admin Assistant Main Menu

4. Admin Assistant confirms commitment of your changes to the rule book.

Figure 7.17. Admin Assistant



The OpenLink Admin Assistant - Microsoft Internet Explorer

Address http://localhost:8000

Database Agent Administration
Session Rules Administration
Default Session Rules
Session Rules Book Aliases
Edit Domain Aliases
Edit Database Aliases
Edit User Aliases

OpenLink Request Broker Instance Reinitialization

OpenLink Request Broker Reinitialized Successfully

Admin Assistant Main Menu

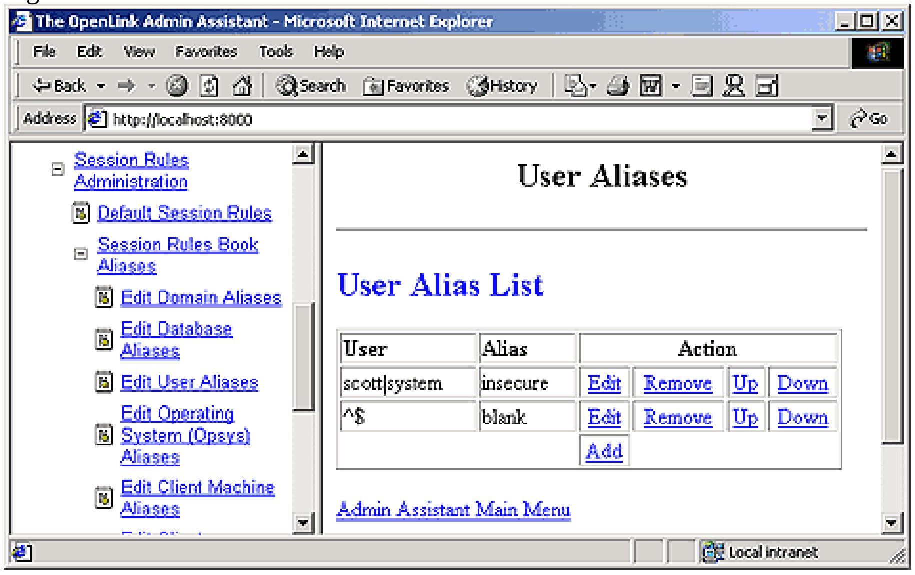
9.3.6.2 User Aliases

The steps that follow show you how to manage User Aliases using the Admin Assistant:

1. Follow the "Server Components Administration"->"Session Rules Administration"->"Session Rules Book Aliases"->"Edit User Aliases" menu tree which brings you to a screen similar to the one depicted below.

   The Admin Assistant presents you with a list of pre-configured User Aliases if any exist, if this is your first time there will be not items listed. Click on the "Add" hyperlink to create a new User Alias.

   Figure 7.18. Admin Assistant


The OpenLink Admin Assistant - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://localhost:8000
Session Rules Administration
Default Session Rules
Session Rules Book Aliases
Edit Domain Aliases
Edit Database Aliases
Edit User Aliases
Edit Operating System (Opsys) Aliases
Edit Client Machine Aliases
Local intranet

User Aliases

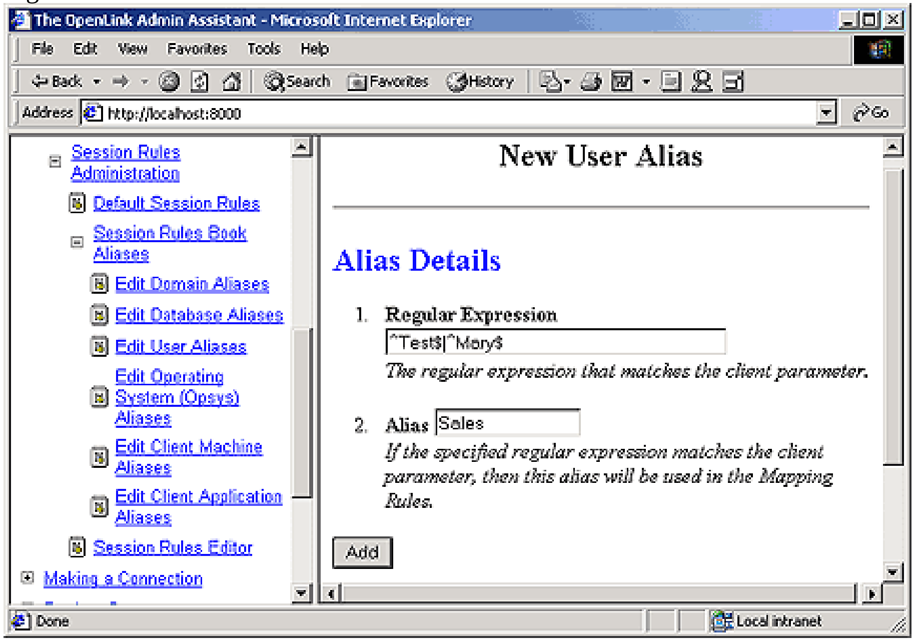
User Alias List

User	Alias	Action			
scott|system	insecure	Edit	Remove	Up	Down
^$	blank	Edit	Remove	Up	Down
		Add			

Admin Assistant Main Menu

2. The example below assumes that you are creating an User Alias named "Sales" with attribute values of "Test" or "Mary", the use of the caret symbol (character "^") and the Dollar symbol indicate the start and end of regular expression evaluation values respectively. The Pipe symbol (character "|") indicates an OR condition. Thus the entry depicted below implies that you want to take note of User Connection Attributes from an OpenLink Client that hold the values "Test" or "Mary" for evaluation against a mapping rules template. Once you have completed your input click the "Add" button.

   Figure 7.19. Admin Assistant


The OpenLink Admin Assistant - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://localhost:8000
Session Rules Administration
Default Session Rules
Session Rules Book Aliases
Edit Domain Aliases
Edit Database Aliases
Edit User Aliases
Edit Operating System (Opsys) Aliases
Edit Client Machine Aliases
Edit Client Application Aliases
Session Rules Editor
Making a Connection
Done
Local intranet

New User Alias

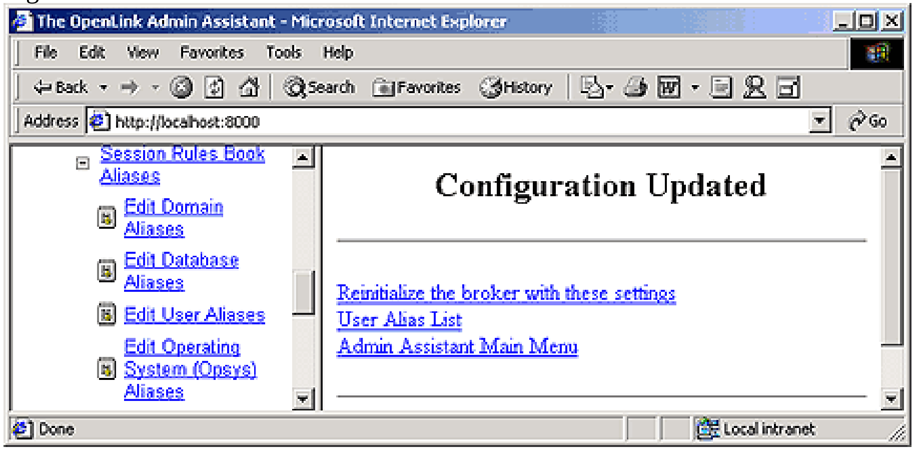
Alias Details

1. Regular Expression ^Test$|^Mary$
   The regular expression that matches the client parameter.
2. Alias Sales
   If the specified regular expression matches the client parameter, then this alias will be used in the Mapping Rules.

Add

3. 3.You then commit your changes to the rule book by clicking on the "Reinitialize the OpenLink Request Broker with these settings" hyperlink. See screen shot below:

Figure 7.20. Admin Assistant

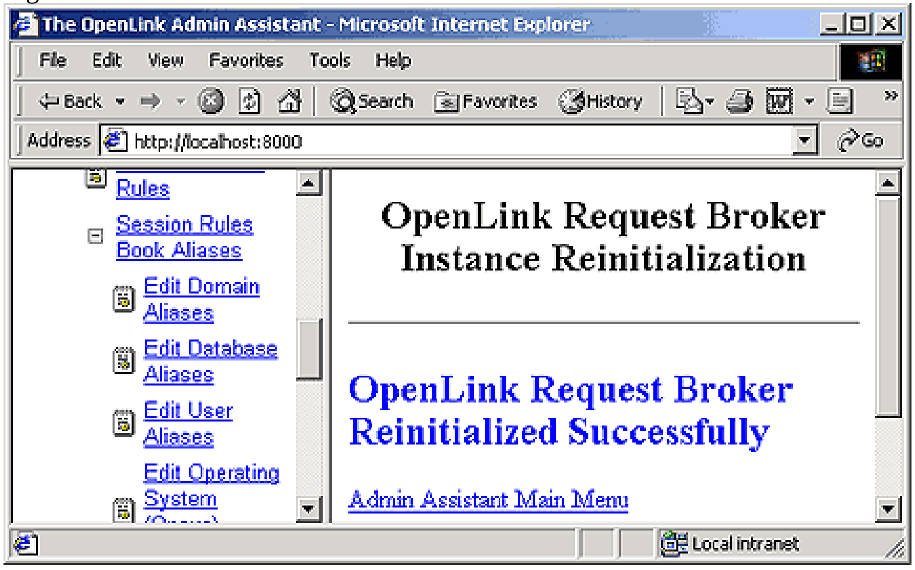

The OpenLink Admin Assistant - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://localhost:8000
Go
Session Rules Book Aliases
Edit Domain Aliases
Edit Database Aliases
Edit User Aliases
Edit Operating System (Opsys) Aliases
Done
Local intranet

Configuration Updated

Reinitialize the broker with these settings
User Alias List
Admin Assistant Main Menu

4. Admin Assistant confirms commitment of your changes to the rule book.

Figure 7.21. Admin Assistant


The OpenLink Admin Assistant - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://localhost:8000
Go
Rules
Session Rules Book Aliases
Edit Domain Aliases
Edit Database Aliases
Edit User Aliases
Edit Operating System
Local intranet

OpenLink Request Broker Instance Reinitialization

OpenLink Request Broker Reinitialized Successfully

Admin Assistant Main Menu

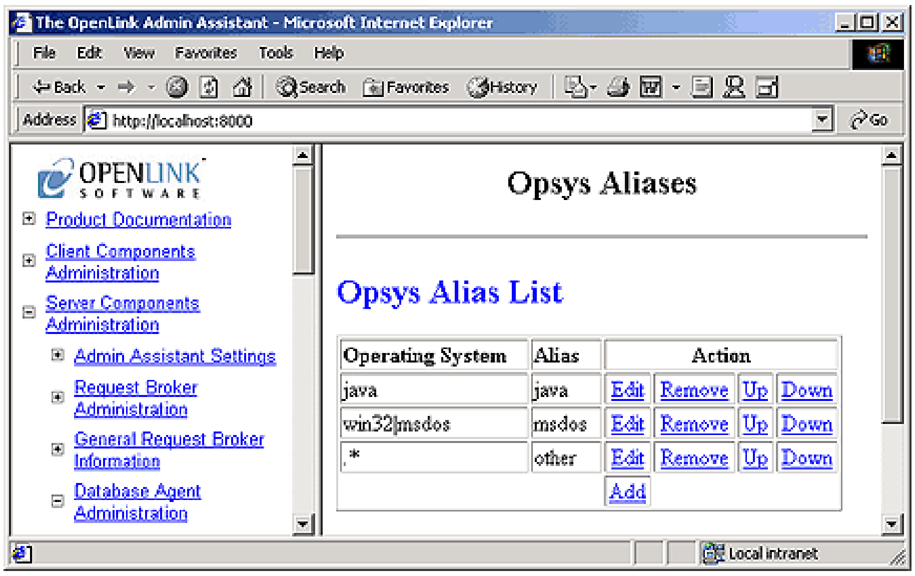
9.3.6.3 OpSys Aliases (Operating System Aliases)

1. The steps that follow show you how to manage OpSys Aliases using the Admin Assistant:

   Follow the "Server Components Administration"->"Session Rules Administration"->"Session Rules Book Aliases"->"Edit Operating System Aliases" menu tree which brings you to a screen identical to the one depicted below.

   The Admin Assistant presents you with a list of pre-configured OpSys Aliases if any exist, if this is your first time no items will be listed. Click on the "Add" hyperlink to create a new OpSys Alias.

   Figure 7.22. Admin Assistant


The OpenLink Admin Assistant - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://localhost:8000
Product Documentation
Client Components Administration
Server Components Administration
Admin Assistant Settings
Request Broker Administration
General Request Broker Information
Database Agent Administration
Local intranet

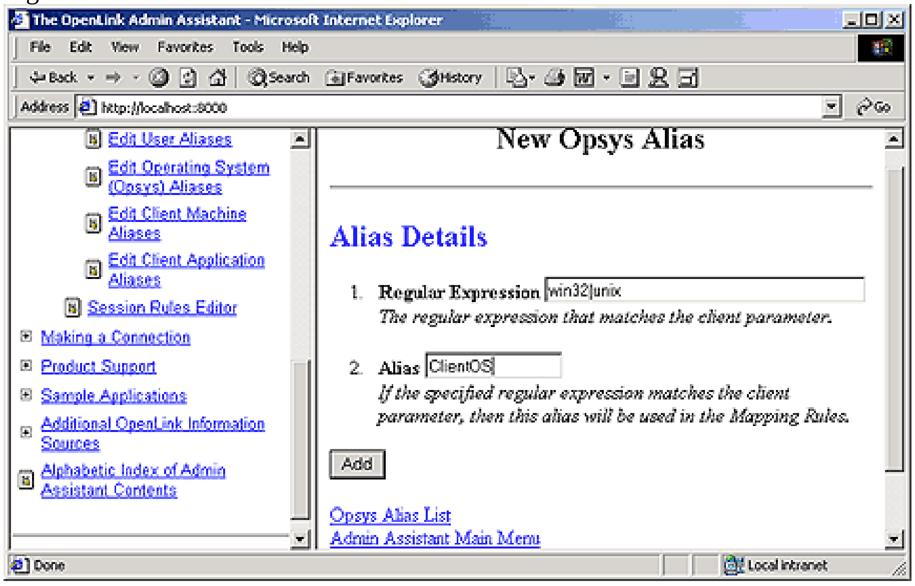
Opsys Aliases

Opsys Alias List

Operating System	Alias	Action			
java	java	Edit	Remove	Up	Down
win32|msdos	msdos	Edit	Remove	Up	Down
.*	other	Edit	Remove	Up	Down
		Add			

2. The example below assumes that you are creating an OpSys Alias named "ClientOS" with attribute values of "win32" or "unix". The Pipe symbol (character "|") indicates an OR condition. Thus the entry depicted below implies that you want to take note of OpSys Connection Attributes from an OpenLink Client that start with the values "win32" or "unix" for evaluation against a mapping rules template. Once you have completed your input click the "Add" button.

Figure 7.23. Admin Assistant


The OpenLink Admin Assistant - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://localhost:8000
Edit User Aliases
Edit Operating System (Opsys) Aliases
Edit Client Machine Aliases
Edit Client Application Aliases
Session Rules Editor
Making a Connection
Product Support
Sample Applications
Additional OpenLink Information Sources
Alphabetic Index of Admin Assistant Contents
Done
Local intranet

New Opsys Alias

Alias Details

1. Regular Expression win32|unix
   The regular expression that matches the client parameter.

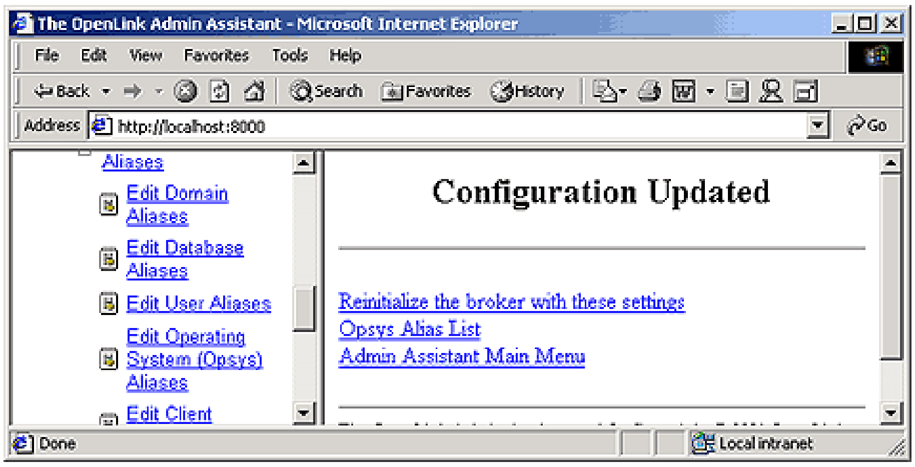
2. Alias ClientOS
   If the specified regular expression matches the client parameter, then this alias will be used in the Mapping Rules.

Add

Opsys Alias List
Admin Assistant Main Menu

3. You then commit your changes to the rule book by clicking on the "Reinitialize the OpenLink Request Broker with these settings" hyperlink. See screen shot below:

Figure 7.24. Admin Assistant

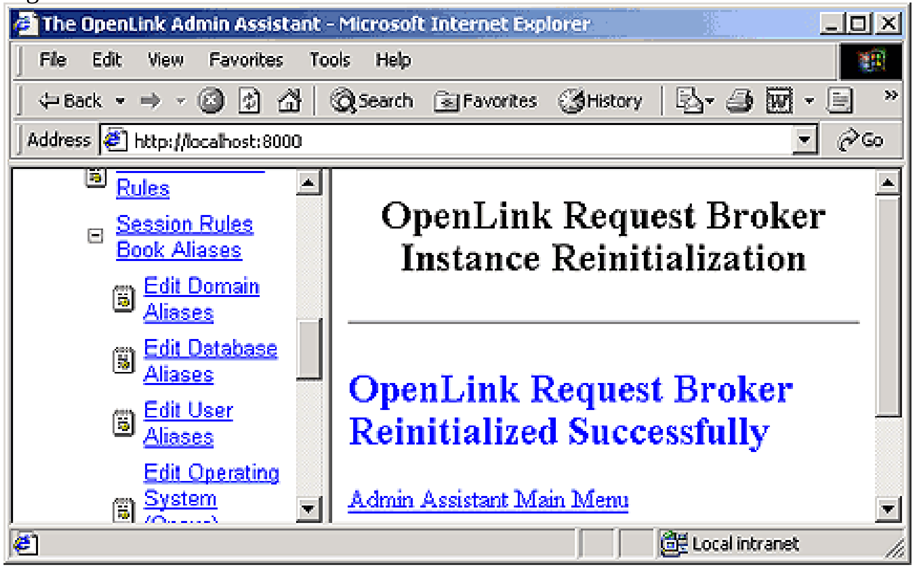

The OpenLink Admin Assistant - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://localhost:8000
Go
Aliases
Edit Domain Aliases
Edit Database Aliases
Edit User Aliases
Edit Operating System (Opsys) Aliases
Edit Client
Done
Local intranet

Configuration Updated

Reinitialize the broker with these settings
Opsys Alias List
Admin Assistant Main Menu

4. Admin Assistant confirms commitment of your changes to the rule book.

Figure 7.25. Admin Assistant


The OpenLink Admin Assistant - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://localhost:8000
Go
Rules
Session Rules Book Aliases
Edit Domain Aliases
Edit Database Aliases
Edit User Aliases
Edit Operating System
Local intranet

OpenLink Request Broker Instance Reinitialization

OpenLink Request Broker Reinitialized Successfully

Admin Assistant Main Menu

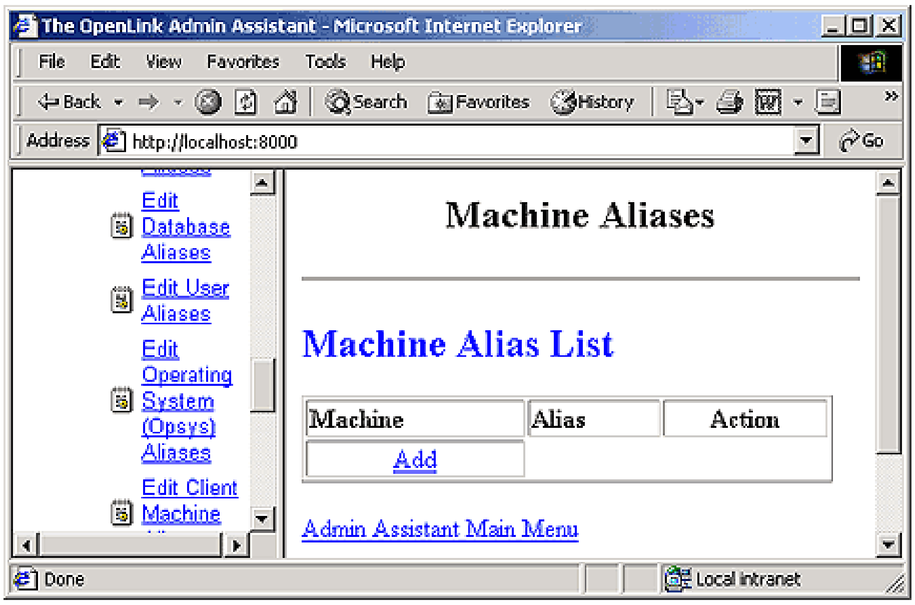
9.3.6.4 Machine Aliases

The steps that follow show you how to manage User Aliases using the Admin Assistant:

1. Follow the "Server Components Administration"->"Session Rules Administration"->"Session Rules Book Aliases"->"Edit Operating Machine Aliases" menu tree which brings you to a screen identical to the one depicted below.

   The Admin Assistant presents you with a list of pre-configured OpSys Aliases if any exist, if this is your first time no items will be listed. Click on the "Add" hyperlink to create a new Machine Alias.

   Figure 7.26. Admin Assistant


The OpenLink Admin Assistant - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://localhost:8000 Go
Edit Database Aliases
Edit User Aliases
Edit Operating System (Opsys) Aliases
Edit Client Machine
Done
Local intranet

Machine Aliases

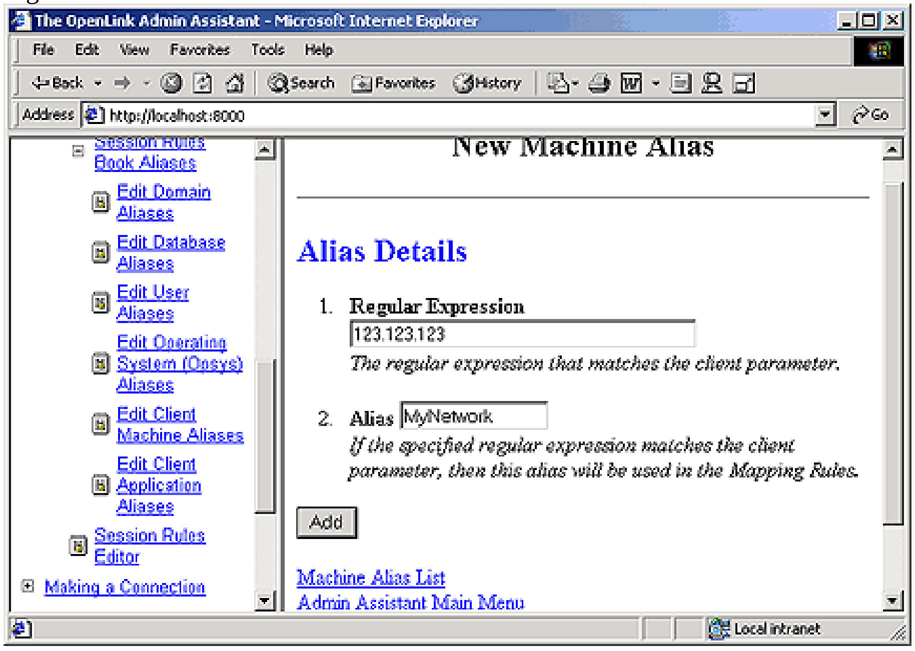
Machine Alias List

Machine	Alias	Action
Add		

Admin Assistant Main Menu

2. The example below assumes that you are creating an Machine Alias named "MyNetwork" with an attribute value of "123.123.123". This implies that you want to take note of Machine Connection Attributes from an OpenLink Client that start with the value "123.123.123" (you would do this to identify the Network portion of the client machines IP address). Once you have completed your input click the "Add" button.

Figure 7.27. Admin Assistant


The OpenLink Admin Assistant - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://localhost:8000 Go
Session Rules Book Aliases
Edit Domain Aliases
Edit Database Aliases
Edit User Aliases
Edit Operating System (Opsys) Aliases
Edit Client Machine Aliases
Edit Client Application Aliases
Session Rules Editor
Making a Connection
Local intranet

New Machine Alias

Alias Details

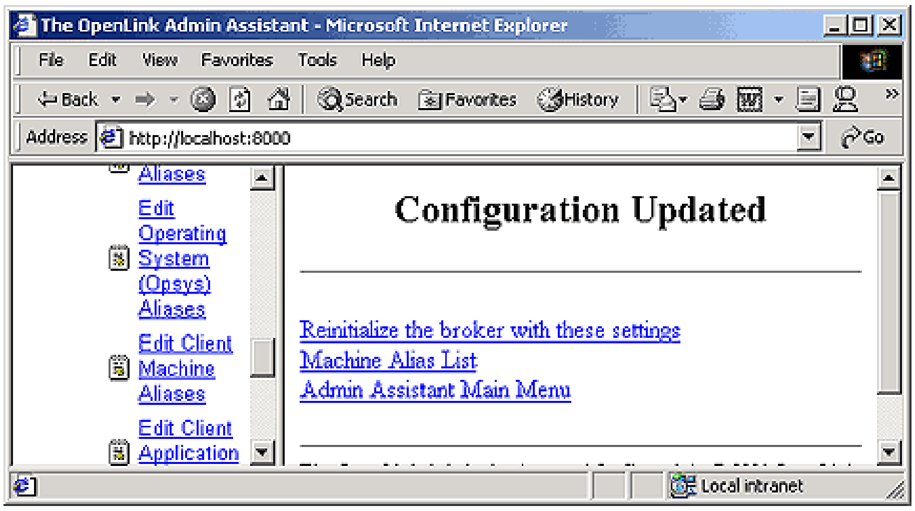
1. Regular Expression: 123.123.123
   The regular expression that matches the client parameter.
2. Alias: MyNetwork
   If the specified regular expression matches the client parameter, then this alias will be used in the Mapping Rules.

Add

Machine Alias List
Admin Assistant Main Menu

3. You then commit your changes to the rule book by clicking on the "Reinitialize the OpenLink Request Broker with these settings" hyperlink. See screen shot below:

Figure 7.28. Admin Assistant

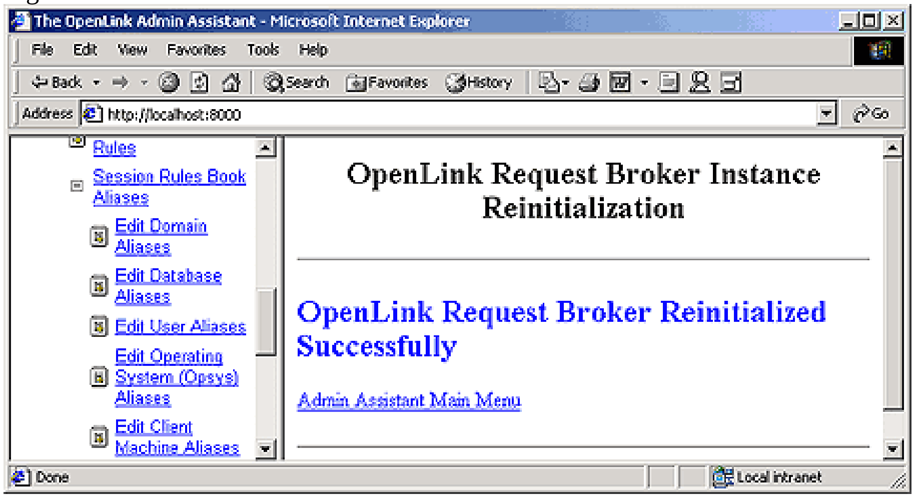

The OpenLink Admin Assistant - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://localhost:8000
Go
Aliases
Edit Operating System (Opsys) Aliases
Edit Client Machine Aliases
Edit Client Application
Local intranet

Configuration Updated

Reinitialize the broker with these settings
Machine Alias List
Admin Assistant Main Menu

4. Admin Assistant confirms commitment of your changes to the rule book.

Figure 7.29. Admin Assistant


The OpenLink Admin Assistant - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://localhost:8000
Go
Rules
Session Rules Book Aliases
Edit Domain Aliases
Edit Database Aliases
Edit User Aliases
Edit Operating System (Opsys) Aliases
Edit Client Machine Aliases
Done
Local intranet

OpenLink Request Broker Instance Reinitialization

OpenLink Request Broker Reinitialized Successfully

Admin Assistant Main Menu

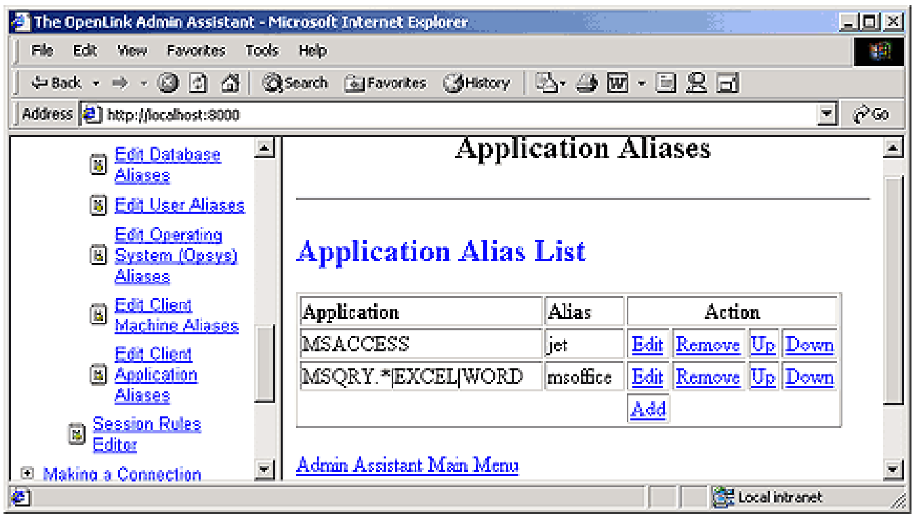
9.3.6.5 Application Aliases

The steps that follow show you how to manage User Aliases using the Admin Assistant:

1. Follow the "Server Components Administration"->"Session Rules Administration"->"Session Rules Book Aliases"->"Edit Application Aliases" menu tree which brings you to a screen identical to the one depicted below.

   The Admin Assistant presents you with a list of pre-configured Application Aliases, click on the "Edit" hyperlink to change settings, the "Remove" hyperlink to delete an Alias, and the "Add" button at the bottom of the screen to create a new Application Alias.

   Figure 7.30. Admin Assistant


The OpenLink Admin Assistant - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://localhost:8000
Edit Database Aliases
Edit User Aliases
Edit Operating System (Opsys) Aliases
Edit Client Machine Aliases
Edit Client Application Aliases
Session Rules Editor
Making a Connection
Local intranet

Application Aliases

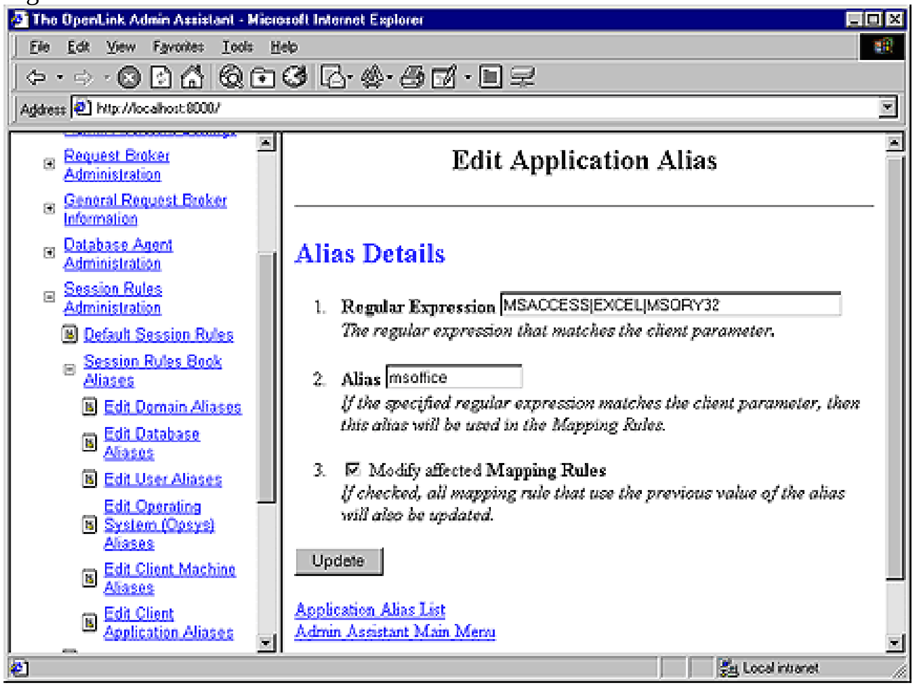
Application Alias List

Application	Alias	Action			
MSACCESS	jet	Edit	Remove	Up	Down
MSQRY.*EXCEL|WORD	msoffice	Edit	Remove	Up	Down
		Add			

Admin Assistant Main Menu

2. The example below assumes that you are modifying an Application Alias named "msoffice" with attribute values of "MSACCESS" or "EXCEL" or "MSQRY32". The Pipe symbol (character "|") indicates an OR condition. Thus, the entry depicted below implies that you want to take note of Application Connection Attributes from an OpenLink Client that hold the values "MSACCESS" or "EXCEL" or "MSQRY32" for evaluation against a mapping rules template. Once you have completed your input click the "Update" button.

Figure 7.31. Admin Assistant


The OpenLink Admin Assistant - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://localhost:8000/
Request Broker Administration
General Request Broker Information
Database Agent Administration
Session Rules Administration
Default Session Rules
Session Rules Book Aliases
Edit Domain Aliases
Edit Database Aliases
Edit User Aliases
Edit Operating System (Opsys) Aliases
Edit Client Machine Aliases
Edit Client Application Aliases
Local intranet

Edit Application Alias

Alias Details

1. Regular Expression MSACCESS|EXCEL|MSQRY32
The regular expression that matches the client parameter.

2. Alias msoffice
If the specified regular expression matches the client parameter, then this alias will be used in the Mapping Rules.

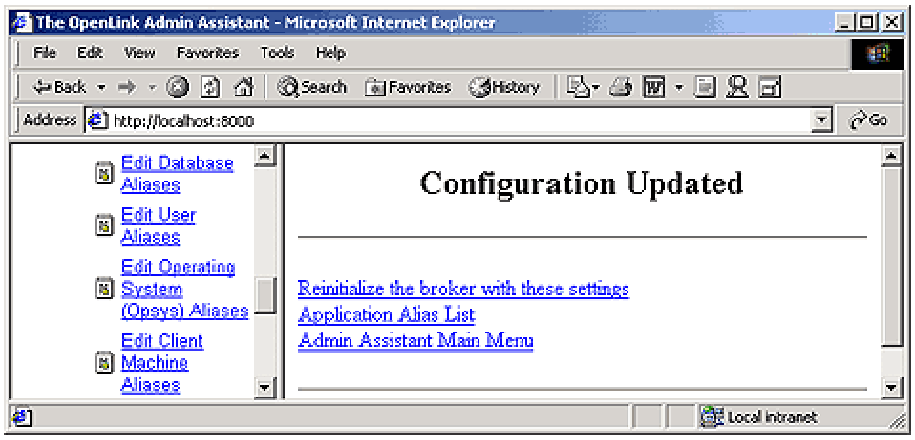
3. ☑ Modify affected Mapping Rules
If checked, all mapping rule that use the previous value of the alias will also be updated.

Update

Application Alias List
Admin Assistant Main Menu

3. You then commit your changes to the rule book by clicking on the "Reinitialize the OpenLink Request Broker with these settings" hyperlink.

Figure 7.32. Admin Assistant

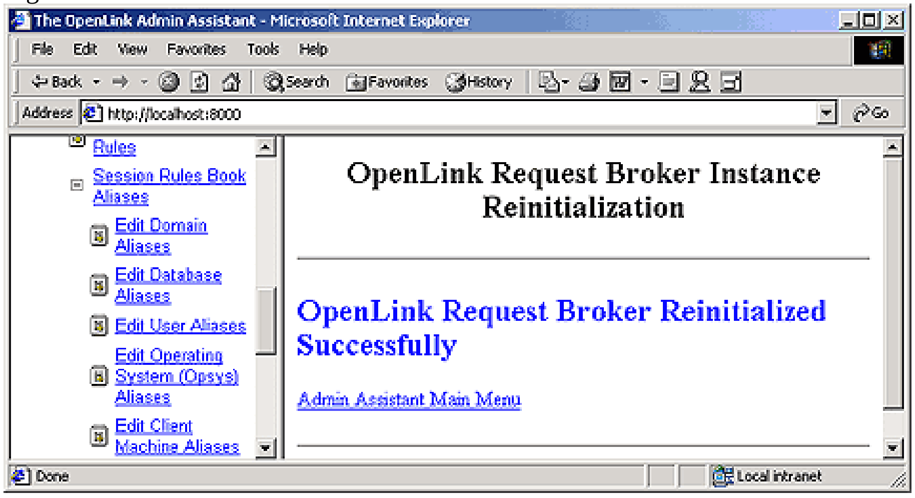

The OpenLink Admin Assistant - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://localhost:8000
Go
Edit Database Aliases
Edit User Aliases
Edit Operating System (Opsys) Aliases
Edit Client Machine Aliases
Local intranet

Configuration Updated

Reinitialize the broker with these settings
Application Alias List
Admin Assistant Main Menu

4. Admin Assistant confirms commitment of your changes to the rule book.

Figure 7.33. Admin Assistant


The OpenLink Admin Assistant - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://localhost:8000
Go
Rules
Session Rules Book Aliases
Edit Domain Aliases
Edit Database Aliases
Edit User Aliases
Edit Operating System (Opsys) Aliases
Edit Client Machine Aliases
Done
Local intranet

OpenLink Request Broker Instance Reinitialization

OpenLink Request Broker Reinitialized Successfully

Admin Assistant Main Menu

9.3.6.6 Determining The Values of OpenLink Client Generated Connection Attributes

When managing Aliases for non user configurable Connection Attributes you need to do the following in order to determine the appropriate values:

1. Start the Request Broker in Debug Mode
2. Then make a connection from your OpenLink Client
3. The debug output in your Request Brokers debug window will contain the following:

```
oplrqb: client-machine.YourDomain called (123.123.123.100.1510)
oplrqb: request: domain=Oracle 8 database= serveropts=
oplrqb: connectopts= user= opsys=win32 readonly=0
oplrqb: application=OPLSCAN processid=384
```

4. Take note of the values assigned to the following Connection Attributes: opsys, readonly, and application. Apply these values to the appropriate Aliases that your are attempting to configure.

9.3.7 Using the Admin Assistant To Create Session Rules

There are different type of session rules that you may want to create the examples that follow guide through the process of creating some very common session rules.

9.3.7.1 Security Oriented "Session Rules" Examples

A major concern with ODBC, JDBC, UDBC and OLE-DB is the fact that you could potentially lose control over the number and types of applications that have full read-write access to your organizations databases, unfortunately database engine based security is User, Group and in some case Role driven, database engines do not have the ability to decipher the type of Client application or data access mechanism being used by a client process. Thus ODBC, UDBC, JDBC, and

OLE-DB can potentially introduce serious security loopholes.

The security and configuration control attributes of the session rules book is OpenLink's single most important and distinguishing feature when compared with other data access technologies (standards or non standards based).

Session rules enable you enforce rules that protect you from many form security related exposure in a time when connectivity is increasing at an alarming rate. The examples that follow use typical security scenarios to demonstrate how session rules can be devised to address security and/or resource consumption issues introduced by the use of ODBC, JDBC, UDBC, and OLE-DB compliant solutions.

9.3.7.2 Creating A Session Rule Enforcing READ-ONLY sessions unconditionally across all OpenLink Clients

The steps that follow guide you through the process of building a session rule that enforces read-only data access across all OpenLink ODBC, JDBC, UDBC, and OLE-DB sessions:

1. Start the Request Broker
2. Start a Web Browser session
3. Enter the following URL into your browser:

   If you started the Request Broker on your local machine enter:

   ```
   http://localhost:8000/
   ```

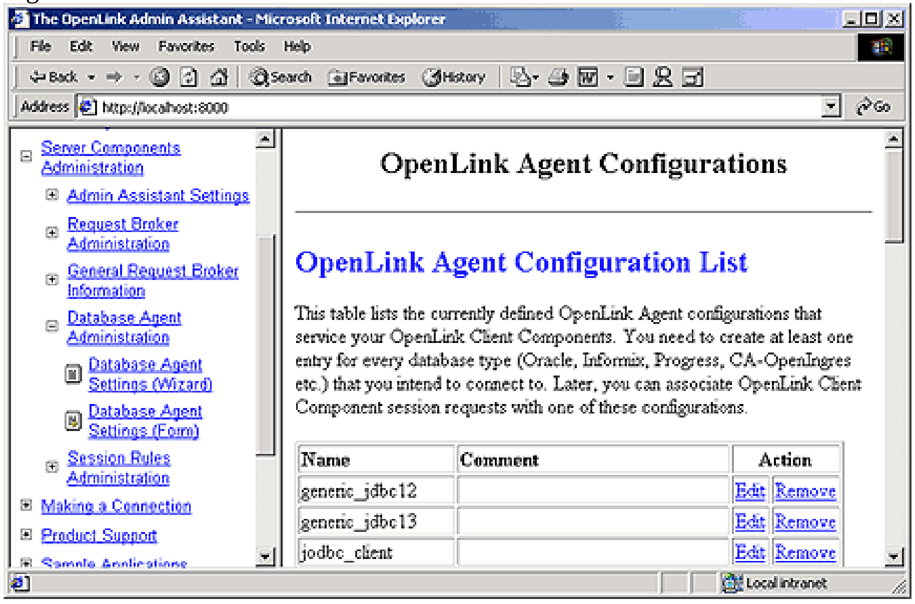
   (assuming you accepted port 8000 as the Admin Assistant port number at installation time).

   If the Request Broker in on another machine enter:

   ```
   http://<hostname or IP address>:8000
   ```

   (assuming you accepted port 8000 as the Admin Assistant port number at installation time).
4. Since you are attempting to create a read-only session rule for a particular database, you need to ensure that you actually have an OpenLink Agent template that has its ReadOnly attribute set to "Y" or "Yes". To do this you navigate the Admin Assistant menu down the following path: "Server Components Administration"->"Database Agent Administration"->"Database Agent Settings (Form)" . Locate the database agent for your database engine type from the list presented and then click on the "Edit" hyperlink.

   Figure 7.34. Admin Assistant

   
   The OpenLink Admin Assistant - Microsoft Internet Explorer
   File Edit View Favorites Tools Help
   Back Search Favorites History
   Address http://localhost:8000 Go
   Server Components Administration
   Admin Assistant Settings
   Request Broker Administration
   General Request Broker Information
   Database Agent Administration
   Database Agent Settings (Wizard)
   Database Agent Settings (Form)
   Session Rules Administration
   Making a Connection
   Product Support
   Sample Applications
   Local intranet

   OpenLink Agent Configurations

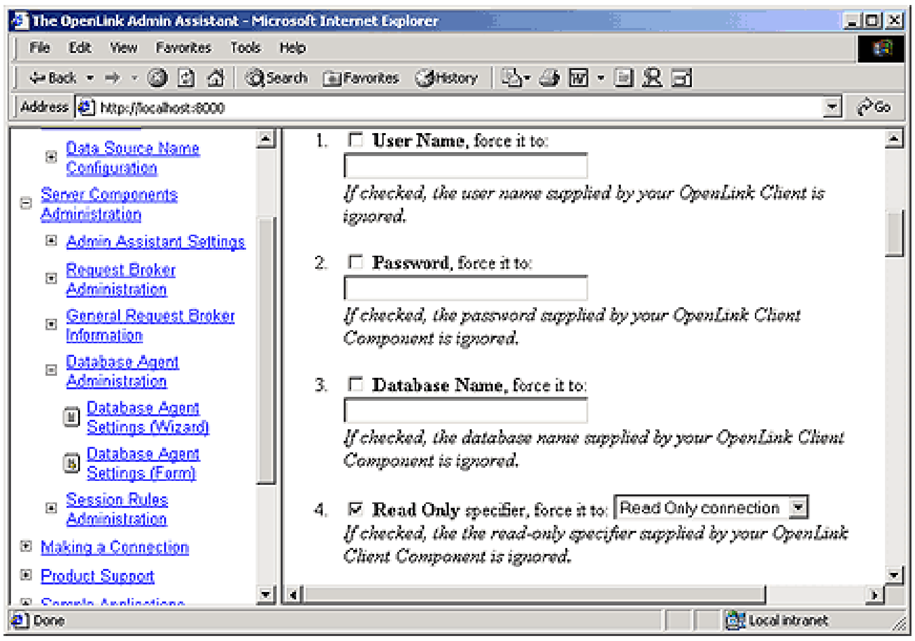
   OpenLink Agent Configuration List

   This table lists the currently defined OpenLink Agent configurations that service your OpenLink Client Components. You need to create at least one entry for every database type (Oracle, Informix, Progress, CA-OpenIngres etc.) that you intend to connect to. Later, you can associate OpenLink Client Component session requests with one of these configurations.

Name	Comment	Action
generic_jdbc12		Edit Remove
generic_jdbc13		Edit Remove
jodbc_client		Edit Remove

5. Locate the "Read Only" checkbox and then hatch the box which indicates the enforcement of Read-Only session when this agent is connected to backend database.

   Figure 7.35. Admin Assistant


The OpenLink Admin Assistant - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://localhost:8000
Data Source Name Configuration
Server Components Administration
Admin Assistant Settings
Request Broker Administration
General Request Broker Information
Database Agent Administration
Database Agent Settings (Wizard)
Database Agent Settings (Form)
Session Rules Administration
Making a Connection
Product Support
Done
Local intranet

1. ☐ **User Name**, force it to:

If checked, the user name supplied by your OpenLink Client is ignored.

2. ☐ **Password**, force it to:

If checked, the password supplied by your OpenLink Client Component is ignored.

3. ☐ **Database Name**, force it to:

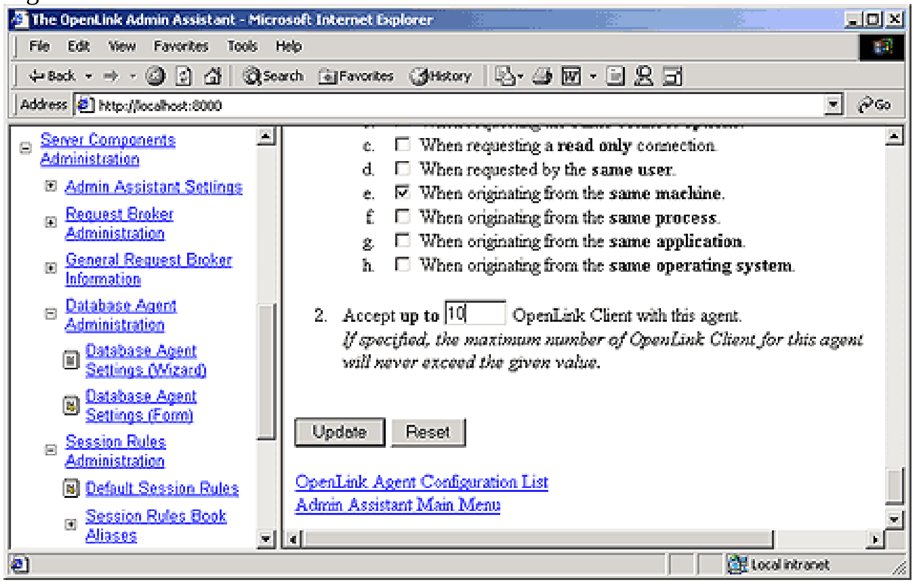
If checked, the database name supplied by your OpenLink Client Component is ignored.

4. ☑ **Read Only** specifier, force it to: Read Only connection

If checked, the the read-only specifer supplied by your OpenLink Client Component is ignored.

6. Proceed to the end of this page and then click on the "Update" button.

Figure 7.36. Admin Assistant


The OpenLink Admin Assistant - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://localhost:8000
Server Components Administration
Admin Assistant Settings
Request Broker Administration
General Request Broker Information
Database Agent Administration
Database Agent Settings (Wizard)
Database Agent Settings (Form)
Session Rules Administration
Default Session Rules
Session Rules Book Aliases
Update
Reset
Local intranet

c. ☐ When requesting a **read only** connection.
d. ☐ When requested by the **same user**.
e. ☑ When originating from the **same machine**.
f. ☐ When originating from the **same process**.
g. ☐ When originating from the **same application**.
h. ☐ When originating from the **same operating system**.

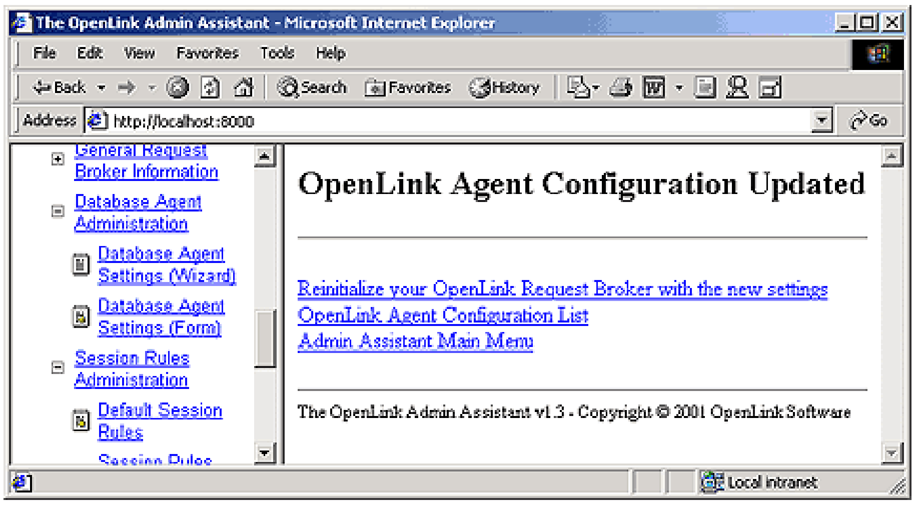
2. Accept **up to** 10 OpenLink Client with this agent.

If specified, the maximum number of OpenLink Client for this agent will never exceed the given value.

OpenLink Agent Configuration List
Admin Assistant Main Menu

7. You then commit your changes to the rule book by clicking on the "Reinitialize the OpenLink Request Broker with these settings" hyperlink.

Figure 7.37. Admin Assistant


The OpenLink Admin Assistant - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://localhost:8000
General Request Broker Information
Database Agent Administration
Database Agent Settings (Wizard)
Database Agent Settings (Form)
Session Rules Administration
Default Session Rules
Local intranet

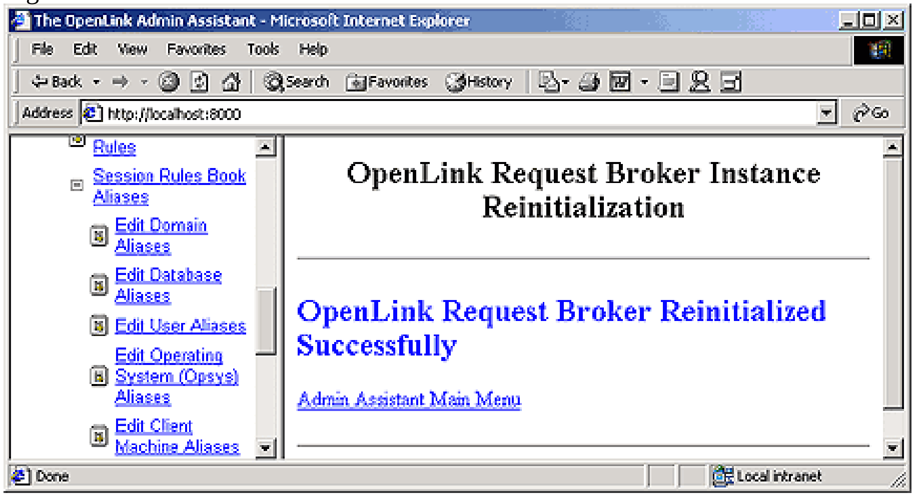
OpenLink Agent Configuration Updated

Reinitialize your OpenLink Request Broker with the new settings
OpenLink Agent Configuration List
Admin Assistant Main Menu

The OpenLink Admin Assistant v1.3 - Copyright © 2001 OpenLink Software

8. Admin Assistant confirms commitment of your changes to the rule book.

Figure 7.38. Admin Assistant


The OpenLink Admin Assistant - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://localhost:8000
Rules
Session Rules Book Aliases
Edit Domain Aliases
Edit Database Aliases
Edit User Aliases
Edit Operating System (Opsys) Aliases
Edit Client Machine Aliases
Done
Local intranet

OpenLink Request Broker Instance Reinitialization

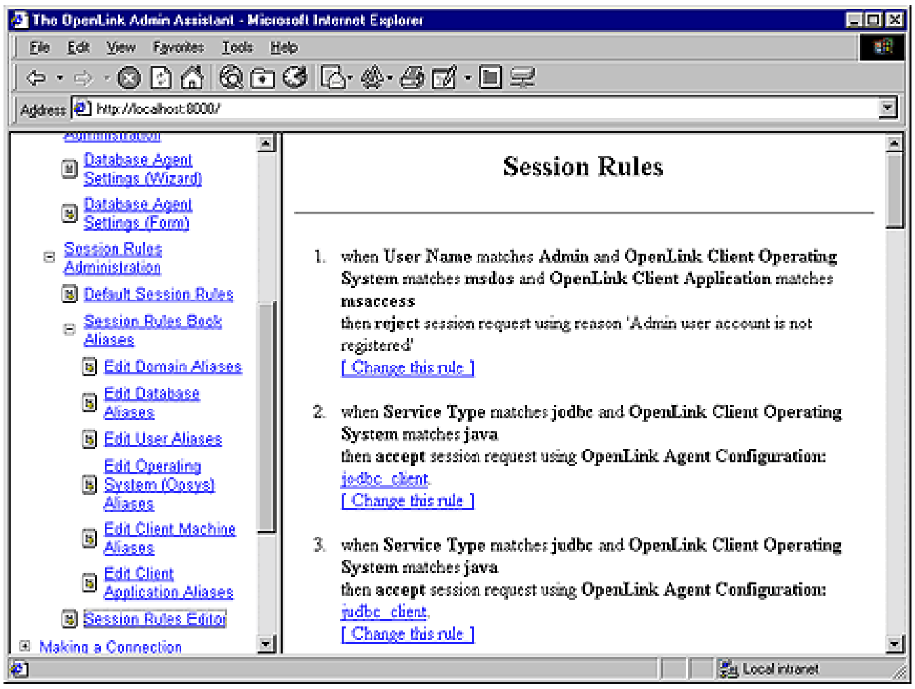
OpenLink Request Broker Reinitialized Successfully

Admin Assistant Main Menu

9. Now that you have created a OpenLink Agent Template for Read-Only sessions against your backend database you can now proceed to the creation of your session rule.

   Follow the "Server Components Administration"->"Session Rules Administration"->"Session Rules Editor" menu path which brings you to a screen identical to the one depicted below, this presents you with a list of existing session rules (all OpenLink installations come with a set of pre-configured session rules).

   Figure 7.39. Admin Assistant


The OpenLink Admin Assistant - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://localhost:8000/
Database Agent Settings (Wizard)
Database Agent Settings (Form)
Session Rules Administration
Default Session Rules
Session Rules Book Aliases
Edit Domain Aliases
Edit Database Aliases
Edit User Aliases
Edit Operating System (Opsys) Aliases
Edit Client Machine Aliases
Edit Client Application Aliases
Session Rules Editor
Making a Connection
Local intranet

Session Rules

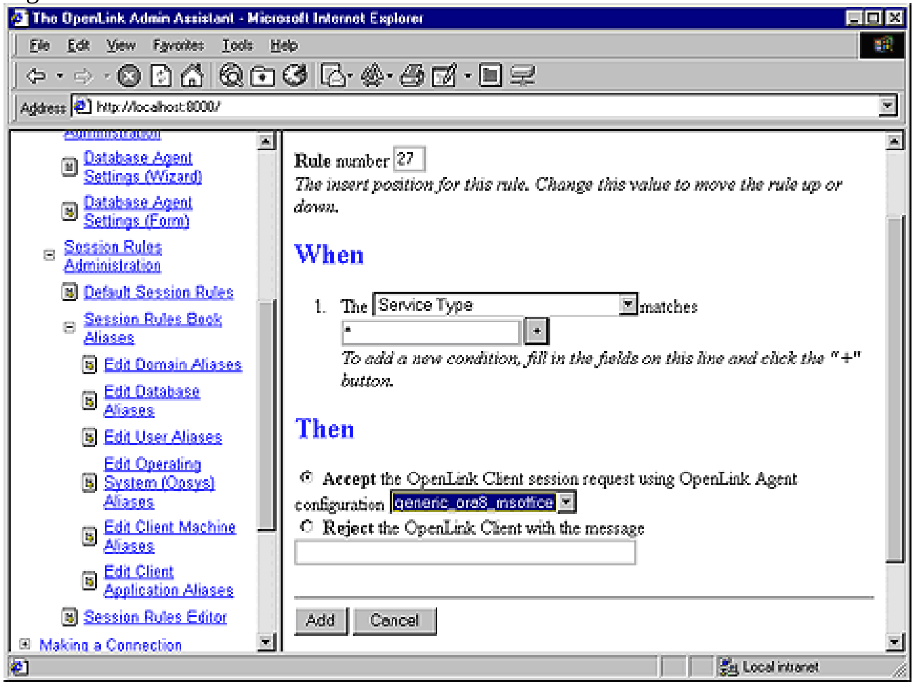
1. when User Name matches Admin and OpenLink Client Operating System matches msdos and OpenLink Client Application matches msaccess
then reject session request using reason 'Admin user account is not registered'
[Change this rule]

2. when Service Type matches jodbc and OpenLink Client Operating System matches java
then accept session request using OpenLink Agent Configuration: jodbc_client
[Change this rule]

3. when Service Type matches judbc and OpenLink Client Operating System matches java
then accept session request using OpenLink Agent Configuration: judbc_client
[Change this rule]

10. Scroll to the bottom of the session rules editor page and then click on the "Add a new rule" hyperlink, this opens up a new session rules page.

Figure 7.40. Admin Assistant


The OpenLink Admin Assistant - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://localhost:8000/
Database Agent Settings (Wizard)
Database Agent Settings (Form)
Session Rules Administration
Default Session Rules
Session Rules Book Aliases
Edit Domain Aliases
Edit Database Aliases
Edit User Aliases
Edit Operating System (Opsys) Aliases
Edit Client Machine Aliases
Edit Client Application Aliases
Session Rules Editor
Making a Connection
Add
Cancel
Local intranet

Rule number 27
The insert position for this rule. Change this value to move the rule up or down.

When

1. The Service Type matches
*
To add a new condition, fill in the fields on this line and click the "+" button.

Then

(●) Accept the OpenLink Client session request using OpenLink Agent configuration generic_ora8_msoffice
( ) Reject the OpenLink Client with the message

11. Create your new session rule.

A session rule is broken down into three parts a "Rule Number", a "When" predicate and a "Then" processing instruction.

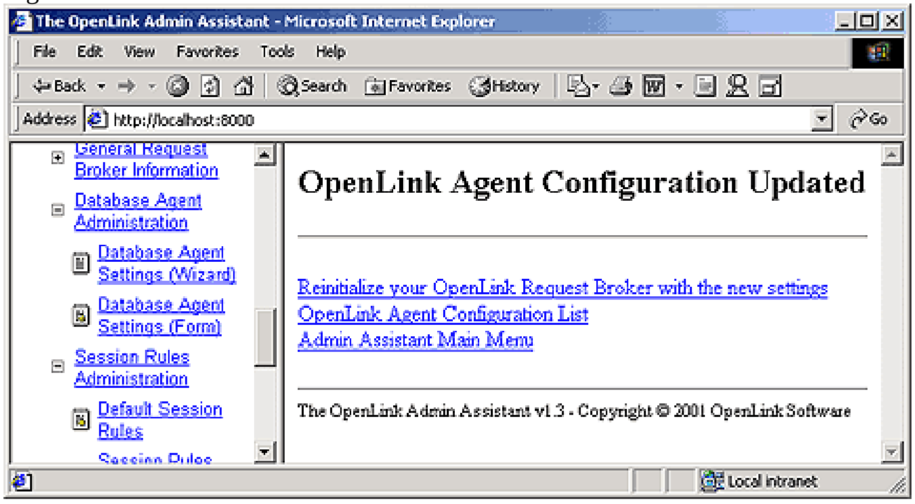
Rule Number: This sets the order in which the session rules are evaluated. The Request Broker reads scans session rules in ascending "Rule Number".

"When" Predicate: Enter a regular expression value into the field beside the "+" button, this identifies to the Request Broker an OpenLink Service Type (a Domain attribute) value that forms one of the Connection Attributes that make up the session rule that you are constructing. Simply click on the "+" button to Add other Connection Attributes that will make up the session rule that you are constructing. In this example we want our session rule to apply to all OpenLink clients thus entering an "*" (which implies any Domain attribute and sub attributes) serves our needs adequately.

"Then" Processing Instruction: Check the "Accept" radio button and then select an OpenLink Agent template to associate with your rule by clicking on the "configuration" Listbox. The template that you choose needs to match the one that you placed in Read-Only mode at the start of this exercise. Then click on the "Add" button.

12. You then commit your changes to the rule book by clicking on the "Reinitialize the OpenLink Request Broker with these settings" hyperlink.

Figure 7.41. Admin Assistant


The OpenLink Admin Assistant - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://localhost:8000 Go
General Request Broker Information
Database Agent Administration
Database Agent Settings (Wizard)
Database Agent Settings (Form)
Session Rules Administration
Default Session Rules
Local intranet

OpenLink Agent Configuration Updated

Reinitialize your OpenLink Request Broker with the new settings
OpenLink Agent Configuration List
Admin Assistant Main Menu

The OpenLink Admin Assistant v1.3 - Copyright © 2001 OpenLink Software

13. Attempt a new OpenLink Client connection to you backend database and then attempt a record update, you will observe that your update attempt will be rejected and an error condition communicated to you via your ODBC, JDBC, UDBC,or OLE-DB based application.

9.3.7.3 Creating A Session Rule That Enforces READ-ONLY sessions for a specific OpenLink Client User

The steps that follow guide you through the process of building a session rule that enforces read-only data access for a specific User account across all OpenLink ODBC, JDBC, UDBC, and OLE-DB sessions.

1. Follow Steps 1 - 10 of the "Read-Only" session rules exercise above.
2. Create your new session rule.

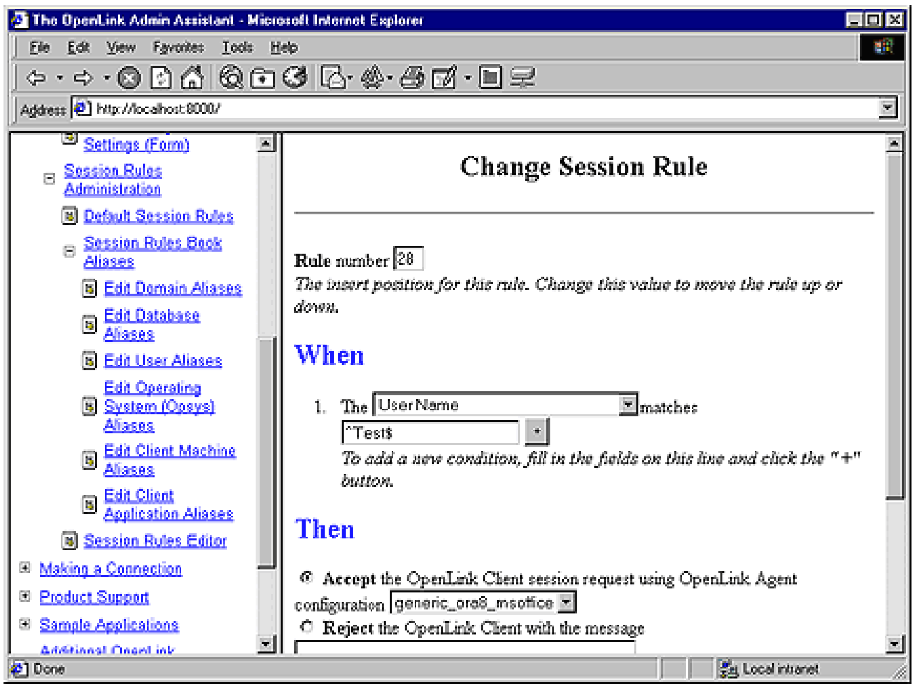
   A session rule is broken down into three parts a 'Rule Number', a "When" predicate and a "Then" processing instruction.

   Rule Number: This sets the order in which the session rules are evaluated. The Request Broker reads scans session rules in ascending "Rule Number".

   "When" Predicate: Enter a regular expression value into the field beside the "+" button, this identifies to the Request Broker an OpenLink Client User (the User attribute) value that forms one of the Connection Attributes that make up the session rule that you are constructing. Simply click on the "+" button to Add other Connection Attributes that will make up the session rule that you are constructing. In this example we want our session rule to apply to a specific OpenLink Client User irrespective of anything else. Thus, entering the value "^Test$" implies that anyone that makes an ODBC, JDBC, UDBC, or OLE-DB connection using the user name "Test" operates against your backend database in Read-Only mode.

   "Then" Processing Instruction: Check the "Accept" radio button and then select an OpenLink Agent template to associate with your rule by clicking on the "configuration" Listbox. The template that you choose needs to match the one that you placed in Read-Only mode at the start of the "Read-Only" session rules exercise. Then click on the "Add" button.

   Figure 7.42. Admin Assistant


The OpenLink Admin Assistant - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://localhost:8000/
Settings (Form)
Session Rules Administration
Default Session Rules
Session Rules Book Aliases
Edit Domain Aliases
Edit Database Aliases
Edit User Aliases
Edit Operating System (Opsys) Aliases
Edit Client Machine Aliases
Edit Client Application Aliases
Session Rules Editor
Making a Connection
Product Support
Sample Applications
Done
Local intranet

Change Session Rule

Rule number 28

The insert position for this rule. Change this value to move the rule up or down.

When

1. The User Name matches ^Test$ +

To add a new condition, fill in the fields on this line and click the "+" button.

Then

[✓ Accept] the OpenLink Client session request using OpenLink Agent configuration generic_ora8_msoffice

Reject the OpenLink Client with the message

3. Click on the "Update" button to complete the rule creation exercise and then reinitialize the Request Broker when prompted.
4. Attempt a connection under the user name "Test" and attempt a database record update. Note that "Test" must be a valid user account in your database environment.

9.3.7.4 Creating A Session Rule That Enforces READ-ONLY sessions for a Group of Users

The steps that follow guide you through the process of building a session rule that enforces read-only data access for a specific group of users across all OpenLink ODBC, JDBC, UDBC, and OLE-DB sessions.

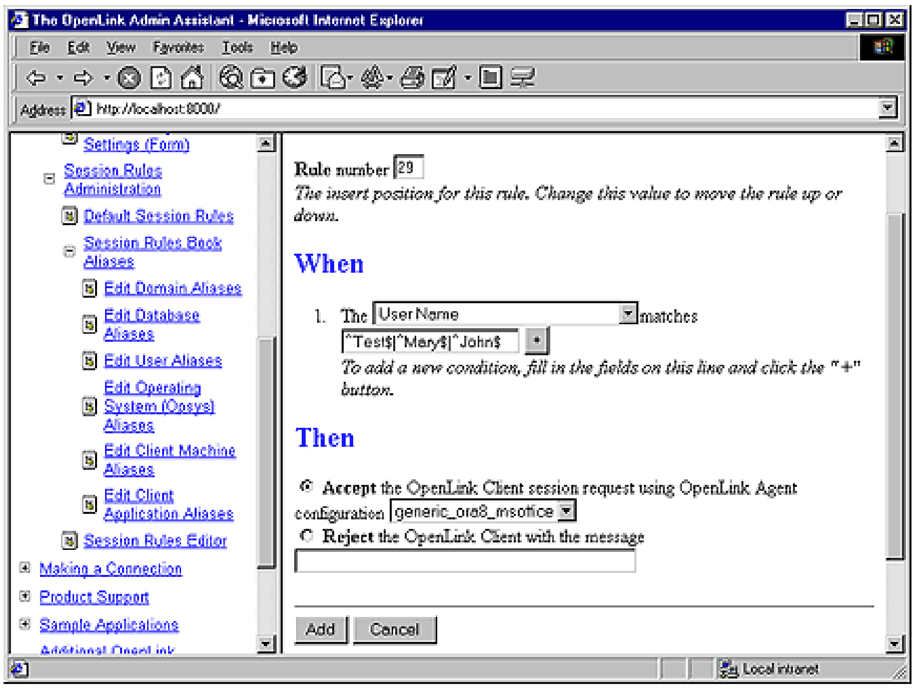
1. Follow Steps 1 - 10 of the "Read-Only" session rules exercise above.
2. A session rule is broken down into three parts a "Rule Number", a "When" predicate and a "Then" processing instruction.

   Rule Number: This sets the order in which the session rules are evaluated. The Request Broker reads scans session rules in ascending "Rule Number".

   "When" Predicate: Enter a regular expression value into the field beside the "+" button, this identifies to the Request Broker an OpenLink Client User (the User attribute) value that forms one of the Connection Attributes that make up the session rule that you are constructing. Simply click on the "+" button to Add other Connection Attributes that will make up the session rule that you are constructing. In this example we want our session rule to apply to a group of OpenLink Client Users irrespective of anything else. Thus, entering the value: "^Test$|^Mary$|^John$" implies that anyone that makes an ODBC, JDBC, UDBC, or OLE-DB connection using either the user name "Test" or "Mary" or "John" operates against your backend database in Read-Only mode.

   "Then" Processing Instruction: Check the "Accept" radio button and then select an OpenLink Agent template to associate with your rule by clicking on the "configuration" Listbox. The template that you choose needs to match the one that you placed in Read-Only mode at the start of the "Read-Only" session rules exercise. Then click on the "Add" button.

   Figure 7.43. Admin Assistant


The OpenLink Admin Assistant - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://localhost:8000/
Settings (Form)
Session Rules Administration
Default Session Rules
Session Rules Book Aliases
Edit Domain Aliases
Edit Database Aliases
Edit User Aliases
Edit Operating System (Opsys) Aliases
Edit Client Machine Aliases
Edit Client Application Aliases
Session Rules Editor
Making a Connection
Product Support
Sample Applications
Add
Cancel
Local intranet

Rule number 29

The insert position for this rule. Change this value to move the rule up or down.

When

1. The UserName matches ^Test$|^Mary$|^John$ +

To add a new condition, fill in the fields on this line and click the "+" button.

Then

◉ **Accept** the OpenLink Client session request using OpenLink Agent configuration generic_ora8_msoffice

○ **Reject** the OpenLink Client with the message

3. Click on the "Update" button to complete the rule creation exercise and then reinitialize the Request Broker when prompted.
4. Attempt a connection under the user name "Test" and attempt a database record update. Note that "Test" must be a valid user account in your database environment.

9.3.7.5 Creating A Session Rule That Enforces READ-ONLY sessions for a specific OpenLink Client Application

The steps that follow guide you through the process of building a session rule that enforces read-only data access for a specific Application.

1. Follow Steps 1 - 10 of the "Read-Only" session rules exercise previously.
2. Create your new session rule.

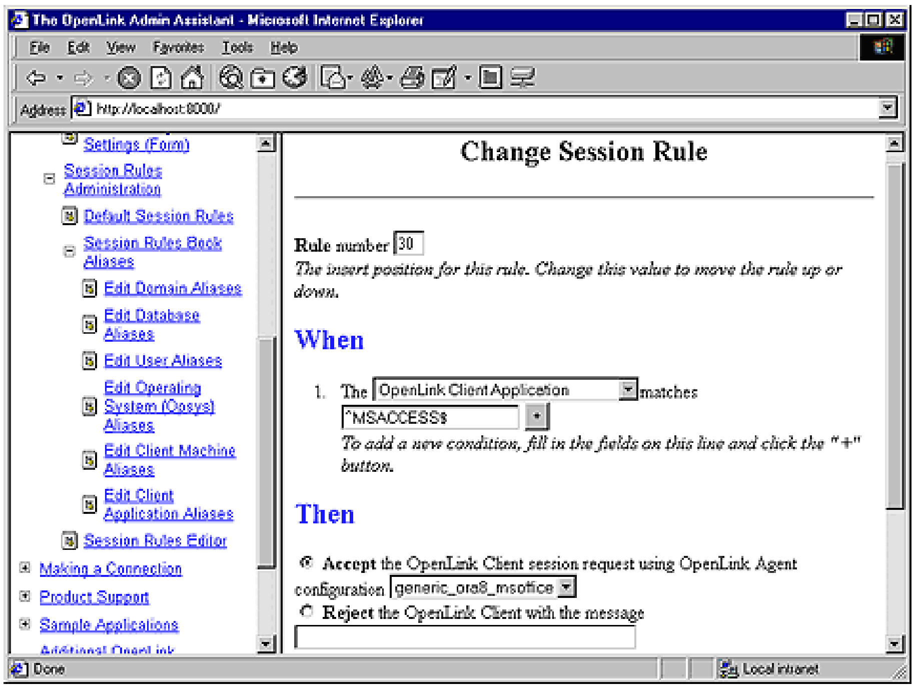
   A session rule is broken down into three parts a "Rule Number", a "When" predicate and a "Then" processing instruction.

   Rule Number: This sets the order in which the session rules are evaluated. The Request Broker reads scans session rules in ascending "Rule Number".

   "When" Predicate: Enter a regular expression value into the field beside the "+" button, this identifies to the Request Broker an OpenLink Client Application (the Application attribute) value that forms one of the Connection Attributes that make up the session rule that you are constructing. Simply click on the "+" button to Add other Connection Attributes that will make up the session rule that you are constructing. In this example we want our session rule to apply to a specific OpenLink Client Application irrespective of anything else. Thus, entering the value "^MSACCESS$" implies that anyone that makes an ODBC, JDBC, UDBC, or OLE-DB connection using the Application presented by an OpenLink Client as "MSACCESS" operates against your backend database in Read-Only mode.

   "Then" Processing Instruction: Check the "Accept" radio button and then select an OpenLink Agent template to associate with your rule by clicking on the "configuration" Listbox. The template that you choose needs to match the one that you placed in Read-Only mode at the start of the "Read-Only" session rules exercise. Then click on the "Add" button.

   Figure 7.44. Admin Assistant


The OpenLink Admin Assistant - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://localhost:8000/
Settings (Form)
Session Rules Administration
Default Session Rules
Session Rules Book Aliases
Edit Domain Aliases
Edit Database Aliases
Edit User Aliases
Edit Operating System (Oosys) Aliases
Edit Client Machine Aliases
Edit Client Application Aliases
Session Rules Editor
Making a Connection
Product Support
Sample Applications
Done
Local intranet

Change Session Rule

Rule number 30
The insert position for this rule. Change this value to move the rule up or down.

When

1. The OpenLink Client Application matches ^MSACCESS$ +
To add a new condition, fill in the fields on this line and click the "+" button.

Then

● Accept the OpenLink Client session request using OpenLink Agent configuration generic_ora8_msoffice
○ Reject the OpenLink Client with the message

3. Click on the "Update" button to complete the rule creation exercise and then reinitialize the Request Broker when prompted.
4. Attempt a connection under the user name "Test" and attempt a database record update. Note that "Test" must be a valid user account in your database environment.

9.3.7.6 Creating A Session Rule That Enforces READ-ONLY sessions for OpenLink Client connections outside of your Local Area Network (LAN)

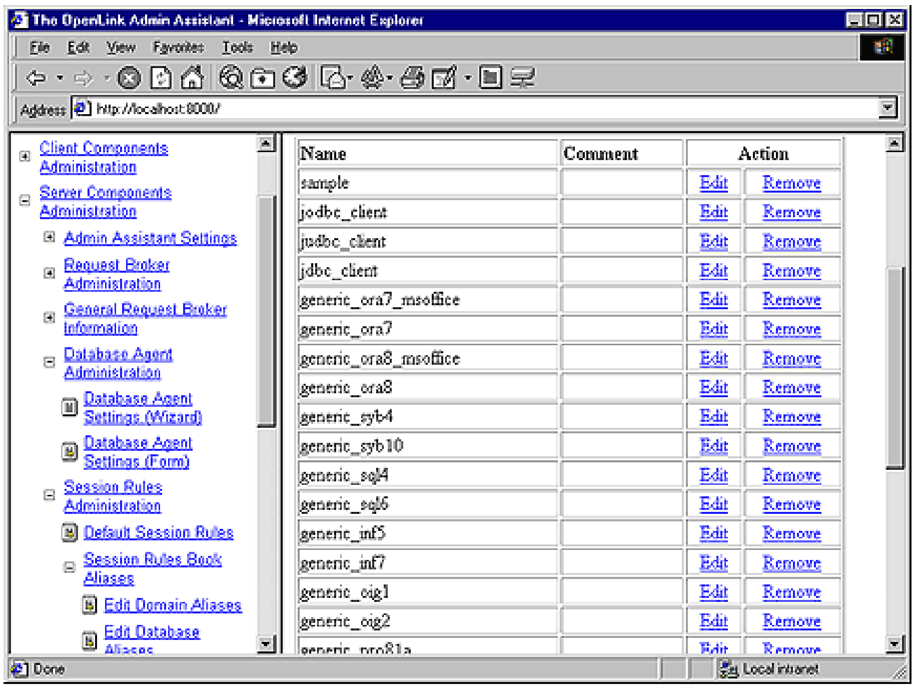
Imagine a scenario in which you staff have access to your corporate databases over the internet due to the fact that you permit employees to work remotely. It may so happen that the work that your require them to do of site requires only a read-only session. The steps that follow guide you through the process of creating a session rule that only gives read-write database access to employees within your network domain, once they connect outside of this domain the sessions are automatically read-only.

This exercise assumes that your LAN is a class C type TCP/IP network with 123.123.123 identifying your LAN.

1. Follow Steps 1 - 3 of the "Read-Only" session rules exercise previously.
2. Since you are attempting to create a read-only session rule for a particular type of connection, you need to ensure that you actually have an OpenLink Agent template for your Read-Only database sessions and another Agent template for your Read-Write database sessions. The ReadOnly attribute of the Read-Only Agent template should be set to "Y" or "Yes", while that of the Read-Write session left unchanged.

   To set things up navigate the Admin Assistant menu down the following path: "Server Components Administration"->"Database Agent Administration"->"Database Agent Settings (Form)" . Locate the database agent for your database engine type from the list presented and then click on the "Edit" hyperlink.

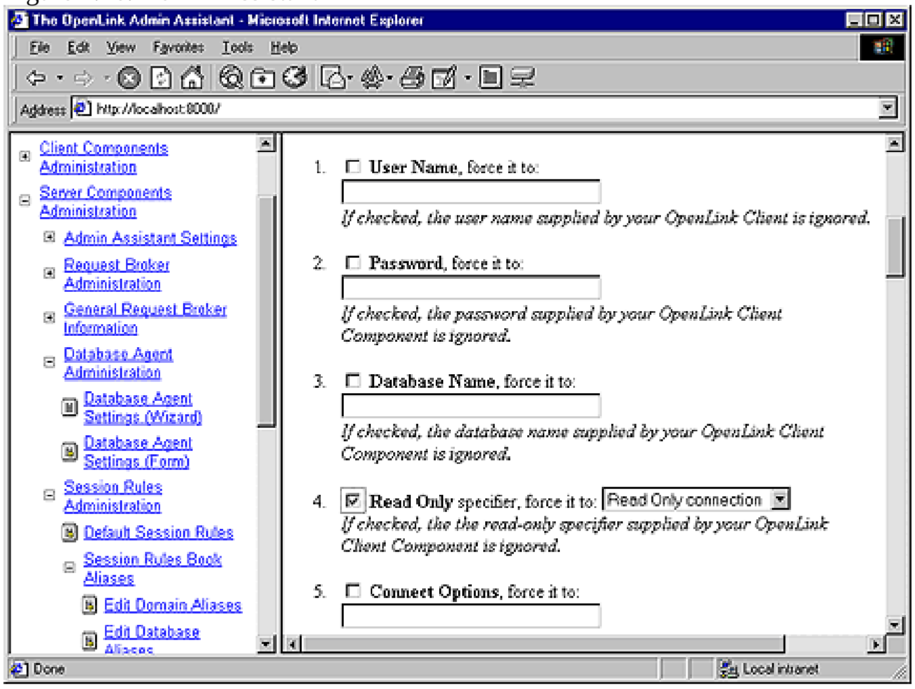
   Figure 7.45. Admin Assistant


The OpenLink Admin Assistant - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://localhost:8000/
Client Components Administration
Server Components Administration
Admin Assistant Settings
Request Broker Administration
General Request Broker Information
Database Agent Administration
Database Agent Settings (Wizard)
Database Agent Settings (Form)
Session Rules Administration
Default Session Rules
Session Rules Book Aliases
Edit Domain Aliases
Edit Database Aliases
Done
Local intranet

Name	Comment	Action	
sample		Edit	Remove
jodbc_client		Edit	Remove
judbc_client		Edit	Remove
jdbc_client		Edit	Remove
generic_ora7_msoffice		Edit	Remove
generic_ora7		Edit	Remove
generic_ora8_msoffice		Edit	Remove
generic_ora8		Edit	Remove
generic_syb4		Edit	Remove
generic_syb10		Edit	Remove
generic_sql4		Edit	Remove
generic_sql6		Edit	Remove
generic_inf5		Edit	Remove
generic_inf7		Edit	Remove
generic_oig1		Edit	Remove
generic_oig2		Edit	Remove
generic_pro81a		Edit	Remove

3. Locate the "Read Only" checkbox and then hatch the box which indicates the enforcement of Read-Only session when this agent is connected to backend database.

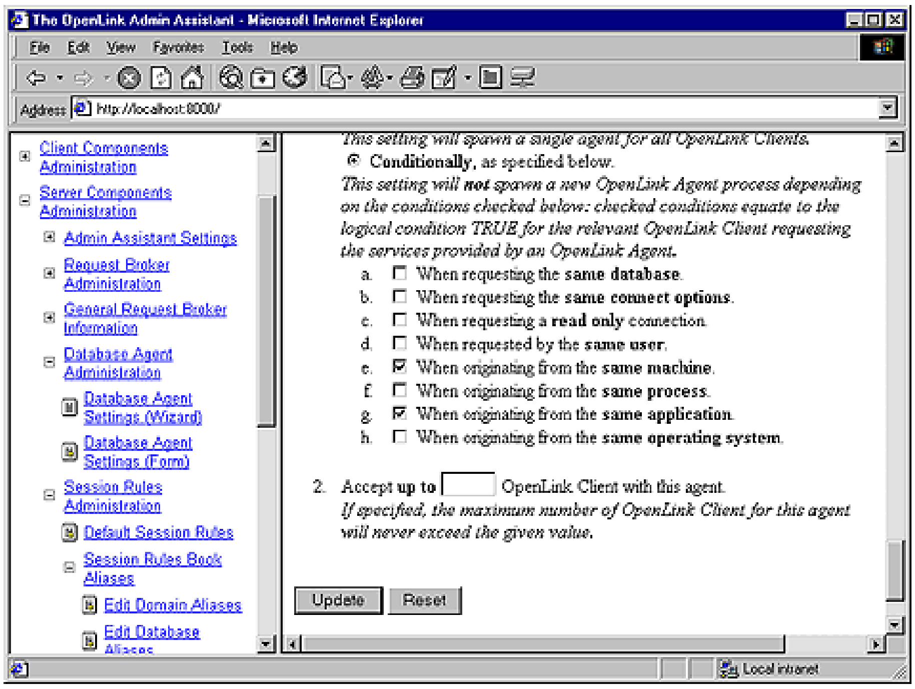
Figure 7.46. Admin Assistant


The OpenLink Admin Assistant - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://localhost:8000/
Client Components Administration
Server Components Administration
Admin Assistant Settings
Request Broker Administration
General Request Broker Information
Database Agent Administration
Database Agent Settings (Wizard)
Database Agent Settings (Form)
Session Rules Administration
Default Session Rules
Session Rules Book Aliases
Edit Domain Aliases
Edit Database Aliases
Done
Local intranet

1. ☐ **User Name**, force it to:
   If checked, the user name supplied by your OpenLink Client is ignored.
2. ☐ **Password**, force it to:
   If checked, the password supplied by your OpenLink Client Component is ignored.
3. ☐ **Database Name**, force it to:
   If checked, the database name supplied by your OpenLink Client Component is ignored.
4. ☑ **Read Only** specifier, force it to: Read Only connection
   If checked, the the read-only specifier supplied by your OpenLink Client Component is ignored.
5. ☐ **Connect Options**, force it to:

4. Proceed to the end of this page and then click on the "Update" button.

Figure 7.47. Admin Assistant


This setting will spawn a single agent for all OpenLink Clients.

[✓ Conditionally], as specified below.

This setting will not spawn a new OpenLink Agent process depending on the conditions checked below: checked conditions equate to the logical condition TRUE for the relevant OpenLink Client requesting the services provided by an OpenLink Agent.

a. ☐ When requesting the same **database**.
b. ☐ When requesting the same **connect options**.
c. ☐ When requesting a **read only** connection.
d. ☐ When requested by the **same user**.
e. ☑ When originating from the **same machine**.
f. ☐ When originating from the **same process**.
g. ☑ When originating from the **same application**.
h. ☐ When originating from the **same operating system**.

2. **Accept up to** [ ] OpenLink Client with this agent.
If specified, the maximum number of OpenLink Client for this agent will never exceed the given value.

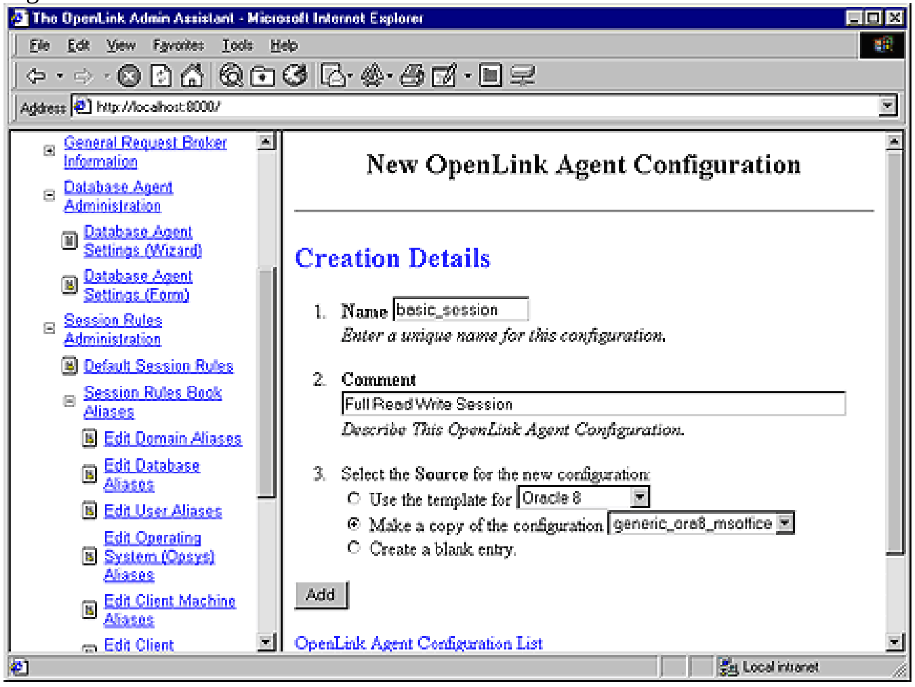
The OpenLink Admin Assistant - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://localhost:8000/
Client Components Administration
Server Components Administration
Admin Assistant Settings
Request Broker Administration
General Request Broker Information
Database Agent Administration
Database Agent Settings (Wizard)
Database Agent Settings (Form)
Session Rules Administration
Default Session Rules
Session Rules Book Aliases
Edit Domain Aliases
Edit Database Aliases
Update
Reset
Local intranet

5. Create a new OpenLink Agent template for the same database, but this time around we will not alter it Read-Only setting since this is disabled by default.

   To set things up by navigate the Admin Assistant menu down the following path: "Server Components Administration"->"Database Agent Administration"->"Database Agent Settings (Form)" . Scroll to the bottom of the page and then click on the "Add" hyperlink.

   Enter a name for your Agent template in the "Name" field, a descriptive comment in the "Comment" field, then click on the "Make a copy of the configuration" and select the template name that matches your Read-Only template.

Figure 7.48. Admin Assistant


New OpenLink Agent Configuration

Creation Details

1. **Name** basic_session
Enter a unique name for this configuration.

2. **Comment**
Full Read Write Session
Describe This OpenLink Agent Configuration.

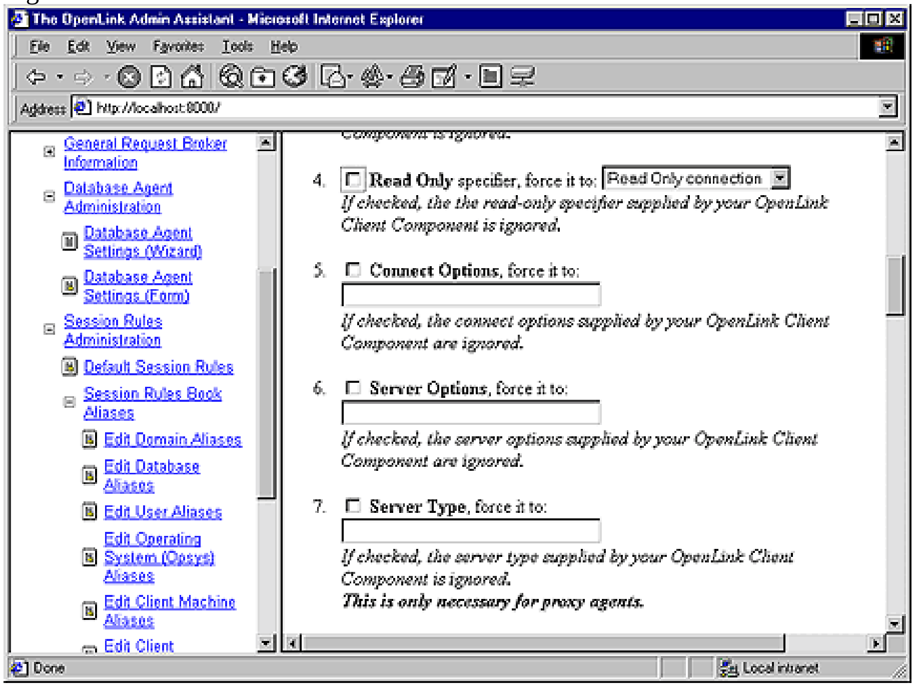
3. Select the **Source** for the new configuration:
○ Use the template for Oracle 8
[✓ Make a copy of the configuration] generic_ora8_msoffice
○ Create a blank entry.

The OpenLink Admin Assistant - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://localhost:8000/
General Request Broker Information
Database Agent Administration
Database Agent Settings (Wizard)
Database Agent Settings (Form)
Session Rules Administration
Default Session Rules
Session Rules Book Aliases
Edit Domain Aliases
Edit Database Aliases
Edit User Aliases
Edit Operating System (Opsys) Aliases
Edit Client Machine Aliases
Edit Client
Add
OpenLink Agent Configuration List
Local intranet

6. You now have a new Agent template named "basic_session", but since this is based on your Read-Only template (which was created earlier) you need to change its Read-Only attributes to that Read-Only session aren't enforced when OpenLink client connections are associated with this template after the "Mapping Rules" evaluation process. To do this navigate the Admin Assistant menu down the following path: "Server Components Administration"->"Database Agent Administration"->"Database Agent Settings (Form)" . Locate

the Agent template named "basic_session" and then click on the "Edit" button. Uncheck the "Read Only" checkbox.

Figure 7.49. Admin Assistant


The OpenLink Admin Assistant - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Address http://localhost:8000/

General Request Broker Information
Database Agent Administration
Database Agent Settings (Wizard)
Database Agent Settings (Form)
Session Rules Administration
Default Session Rules
Session Rules Book Aliases
Edit Domain Aliases
Edit Database Aliases
Edit User Aliases
Edit Operating System (Opsys) Aliases
Edit Client Machine Aliases
Edit Client

Component is ignored.

4. ☐ **Read Only** specifier, force it to: Read Only connection
If checked, the the read-only specifier supplied by your OpenLink Client Component is ignored.

5. ☐ **Connect Options**, force it to:
If checked, the connect options supplied by your OpenLink Client Component are ignored.

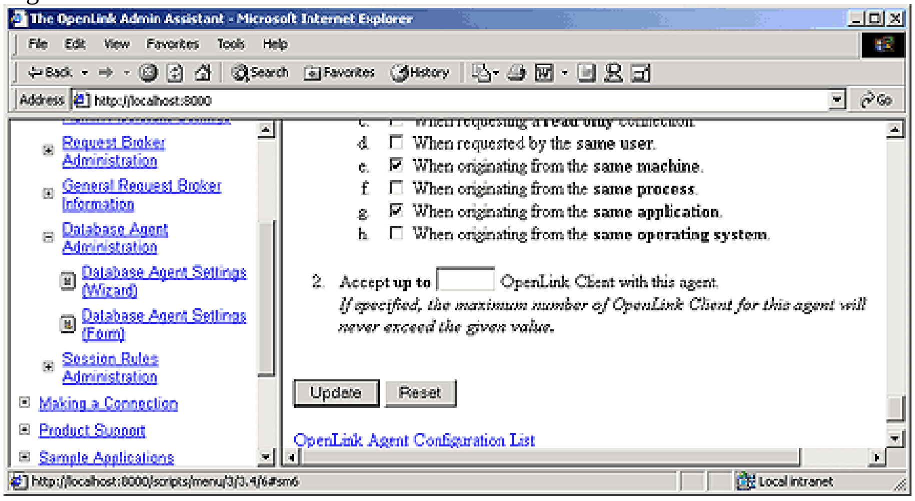
6. ☐ **Server Options**, force it to:
If checked, the server options supplied by your OpenLink Client Component are ignored.

7. ☐ **Server Type**, force it to:
If checked, the server type supplied by your OpenLink Client Component is ignored.
This is only necessary for proxy agents.

Done — Local intranet

7. Proceed to the end of this page and then click on the "Update" button.

Figure 7.50. Admin Assistant


The OpenLink Admin Assistant - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Back Search Favorites History

Address http://localhost:8000 Go

Request Broker Administration
General Request Broker Information
Database Agent Administration
Database Agent Settings (Wizard)
Database Agent Settings (Form)
Session Rules Administration
Making a Connection
Product Support
Sample Applications

c. ☐ When requesting a read only connection.
d. ☐ When requested by the **same user**.
e. ☑ When originating from the **same machine**.
f. ☐ When originating from the **same process**.
g. ☑ When originating from the **same application**.
h. ☐ When originating from the **same operating system**.

2. Accept **up to** ____ OpenLink Client with this agent.
If specified, the maximum number of OpenLink Client for this agent will never exceed the given value.

Update Reset

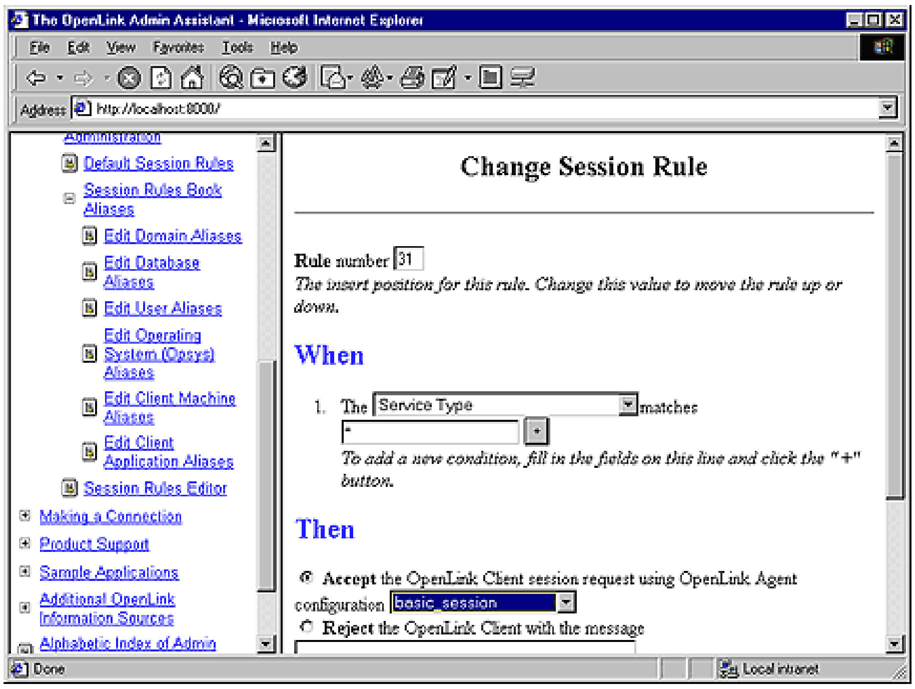
OpenLink Agent Configuration List

http://localhost:8000/scripts/menu/3/3.4/6#sm6 — Local intranet

8. Proceed to the creation of your session rule.

Follow the "Server Components Administration"->"Session Rules Administration"->"Session Rules Editor" menu path which brings you to the session rules editor, click on the "Add new rule" hyperlink so that the new session rule page is presented to you.

Figure 7.51. Admin Assistant


The OpenLink Admin Assistant - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://localhost:8000/
Administration
Default Session Rules
Session Rules Book Aliases
Edit Domain Aliases
Edit Database Aliases
Edit User Aliases
Edit Operating System (Opsys) Aliases
Edit Client Machine Aliases
Edit Client Application Aliases
Session Rules Editor
Making a Connection
Product Support
Sample Applications
Additional OpenLink Information Sources
Alphabetic Index of Admin
Done
Local intranet

Change Session Rule

Rule number 31

The insert position for this rule. Change this value to move the rule up or down.

When

1. The Service Type matches *

To add a new condition, fill in the fields on this line and click the "+" button.

Then

◉ Accept the OpenLink Client session request using OpenLink Agent configuration basic_session

○ Reject the OpenLink Client with the message

9. Create your new session rule.

A session rule is broken down into three parts a "Rule Number", a "When" predicate and a "Then" processing instruction.

Rule Number: This sets the order in which the session rules are evaluated. The Request Broker reads scans session rules in ascending "Rule Number". Since you are going to be creating two rules (one for read-write sessions and another for read-only sessions), you need to order your rules appropriately, remember Request Broker reads these rules for evaluation in ascending order. In this case the rule for read-write sessions is the exception to the norm so we change the number of this rule to 1.

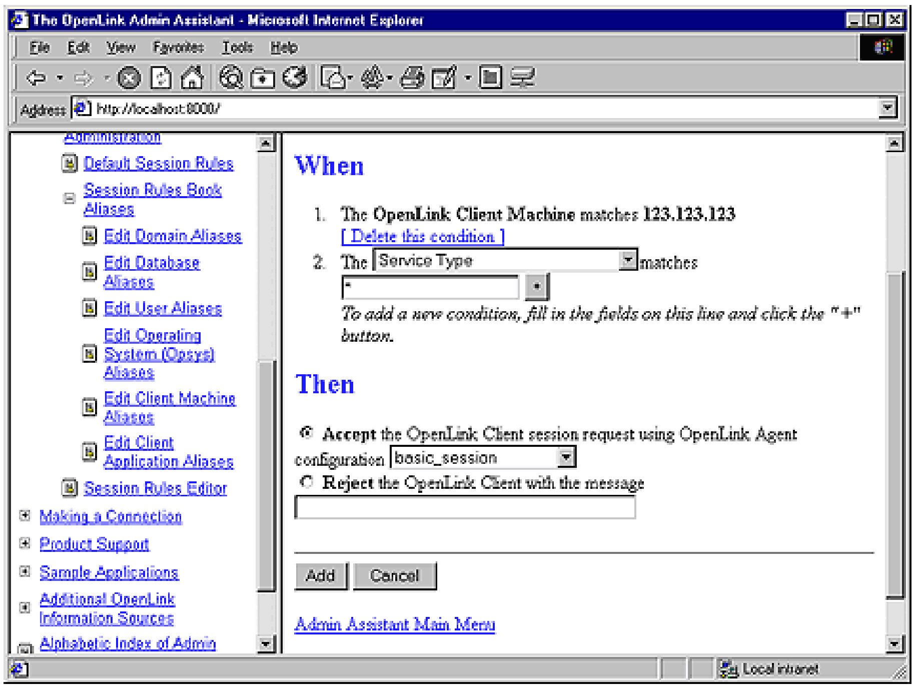
"When" Predicate: Enter a regular expression value into the field beside the "+" button, this identifies to the Request Broker an OpenLink Client Machine (the Machine attribute) value that forms one of the Connection Attributes that make up the session rule that you are constructing. Simply click on the "+" button to Add other Connection Attributes that will make up the session rule that you are constructing. In this example we want our session rule to apply to a specific OpenLink Client Machines irrespective of anything else. Thus, entering the value "123.123.123" implies that any machine that makes an ODBC, JDBC, UDBC, or OLE-DB connection with an from an IP Address that starts with "123.123.123" operates against your backend database in Read-Write mode.

9.3.7.6.1 "Then" Processing Instruction:

Check the "Accept" radio button and then select an OpenLink Agent template to associate with your rule by clicking on the "configuration" Listbox. The template that you choose needs to match the one that you placed in Read-Only mode at the start of the Read-Only" session rules exercise. Then click on the "Add" button.

Logically all other connections will end up being in Read-Only mode since they will not be resolved to having an IP Address that starts with "123.123.123". This also means that they will not be associated with the "basic_agent" Agent Template which is the only means of establishing a Read-Write session with your backend database.

Figure 7.52. Admin Assistant


The OpenLink Admin Assistant - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://localhost:8000/
Administration
Default Session Rules
Session Rules Book Aliases
Edit Domain Aliases
Edit Database Aliases
Edit User Aliases
Edit Operating System (Opsys) Aliases
Edit Client Machine Aliases
Edit Client Application Aliases
Session Rules Editor
Making a Connection
Product Support
Sample Applications
Additional OpenLink Information Sources
Alphabetic Index of Admin
Add
Cancel
Admin Assistant Main Menu
Local intranet

When

1. The OpenLink Client Machine matches 123.123.123
   [Delete this condition]
2. The Service Type matches
   *
   To add a new condition, fill in the fields on this line and click the "+" button.

Then

[✓ Accept the OpenLink Client session request using OpenLink Agent configuration basic_session]

Reject the OpenLink Client with the message

10. Click on the "Update" button to complete the rule creation exercise and then reinitialize the Request Broker when prompted.
11. Attempt a connection from outside your LAN and attempt a database record update.

9.3.7.7 Resource Management Oriented "Session Rules" Examples

The examples addressed in this section focus on the the management of OpenLink Agent processes with the view to providing systems administrators with the ability to tune OpenLink Agents in line with operating system resource constraints.

9.3.7.7.1 Unconditionally Sharing One Database Agent Instance Across Numerous OpenLink Clients

You might want to reduce the number of OpenLink Agents in use at any given time, a good reason for doing this stems from the fact that OpenLink Agent instances are processes which like other processes on your machine will consume a chunk of resources per instance. The more Agents you have running the more operating system kernel resources are consumed. Thus, by setting a ReUse attribute value of "always" you could ensure that under no condition is more than one Agent instance running.

Note: An exception to this rule only occurs if the Request Broker attempts to associate a new OpenLink Client session with an existing agent that hasn't completed its processing cycle, under these circumstance the Request Broker will spawn a new Agent instance.

The steps that follow guide you through the process of creating the appropriate session rule using the Admin Assistant:

1. Start the Request Broker
2. Start a Web Browser session
3. Enter the following URL into your browser:

   If you started the Request Broker on your local machine enter:

   ```
   http://localhost:8000/
   ```

   (assuming you accepted port 8000 as the Admin Assistant port number at installation time).

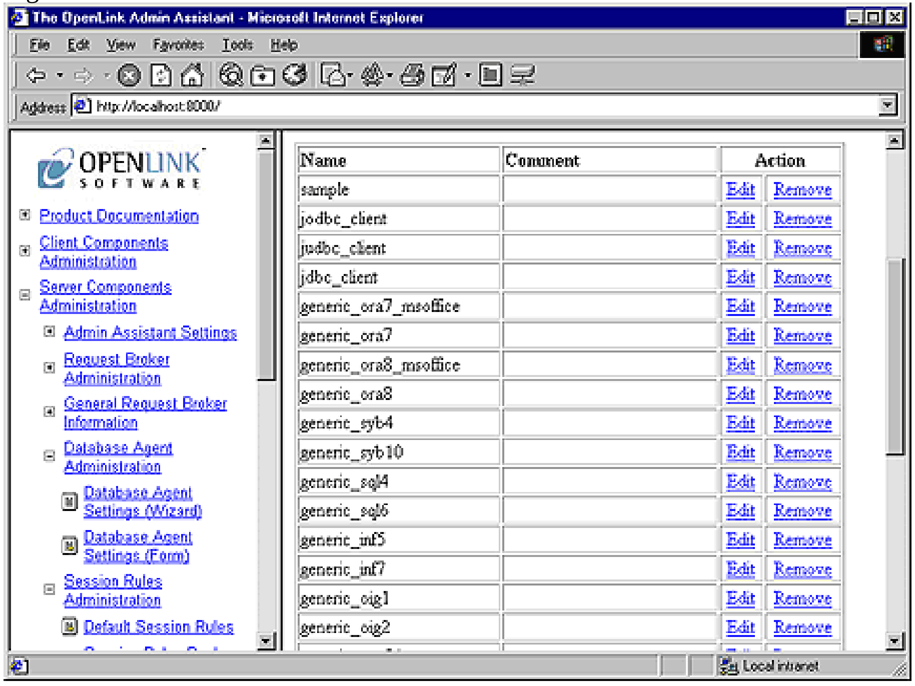
   If the Request Broker in on another machine enter:

   ```
   http://<hostname or IP address>:8000/
   ```

   (assuming you accepted port 8000 as the Admin Assistant port number at installation time).

4. Since you are attempting to modify the ReUse attribute of an existing Agent Template simply navigate the Admin Assistant menu down the following path: "Server Components Administration"->"Database Agent Administration"->"Database Agent Settings (Form)". Locate the database agent for your database engine type from the list presented and then click on the "Edit" hyperlink.

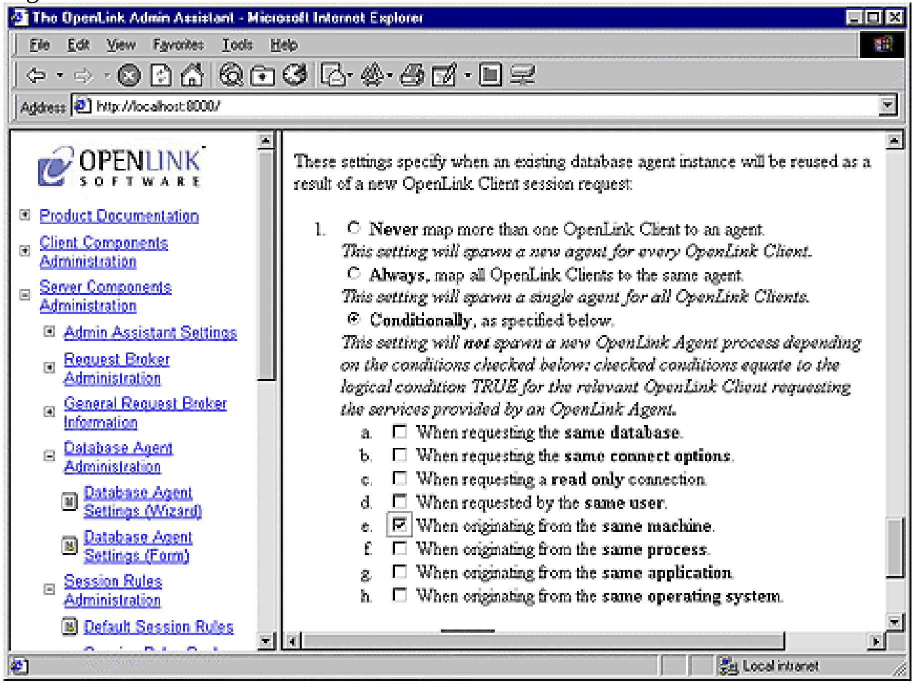
Figure 7.53. Admin Assistant


The OpenLink Admin Assistant - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://localhost:8000/
OPENLINK SOFTWARE
Product Documentation
Client Components Administration
Server Components Administration
Admin Assistant Settings
Request Broker Administration
General Request Broker Information
Database Agent Administration
Database Agent Settings (Wizard)
Database Agent Settings (Form)
Session Rules Administration
Default Session Rules
Local intranet

Name	Comment	Action	
sample		Edit	Remove
iodbc_client		Edit	Remove
iudbc_client		Edit	Remove
jdbc_client		Edit	Remove
generic_ora7_msoffice		Edit	Remove
generic_ora7		Edit	Remove
generic_ora8_msoffice		Edit	Remove
generic_ora8		Edit	Remove
generic_syb4		Edit	Remove
generic_syb10		Edit	Remove
generic_sql4		Edit	Remove
generic_sql6		Edit	Remove
generic_inf5		Edit	Remove
generic_inf7		Edit	Remove
generic_oig1		Edit	Remove
generic_oig2		Edit	Remove

5. Scroll down to the "Client-Server Process Mapping & Control section, and then check the "Always" radio button as depicted in the screen shot below.

Figure 7.54. Admin Assistant


The OpenLink Admin Assistant - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://localhost:8000/
OPENLINK SOFTWARE
Product Documentation
Client Components Administration
Server Components Administration
Admin Assistant Settings
Request Broker Administration
General Request Broker Information
Database Agent Administration
Database Agent Settings (Wizard)
Database Agent Settings (Form)
Session Rules Administration
Default Session Rules
Local intranet

These settings specify when an existing database agent instance will be reused as a result of a new OpenLink Client session request:

1. ○ **Never** map more than one OpenLink Client to an agent.
This setting will spawn a new agent for every OpenLink Client.
○ **Always**, map all OpenLink Clients to the same agent.
This setting will spawn a single agent for all OpenLink Clients.
[✓ **Conditionally**, as specified below.]
*This setting will **not** spawn a new OpenLink Agent process depending on the conditions checked below; checked conditions equate to the logical condition TRUE for the relevant OpenLink Client requesting the services provided by an OpenLink Agent.*
   a. ☐ When requesting the same **database**.
   b. ☐ When requesting the same **connect options**.
   c. ☐ When requesting a **read only** connection.
   d. ☐ When requested by the **same user**.
   e. ☑ When originating from the **same machine**.
   f. ☐ When originating from the **same process**.
   g. ☐ When originating from the **same application**.
   h. ☐ When originating from the **same operating system**.

6. Click on the "Update" button to complete the rule creation exercise and then reinitialize the Request Broker when prompted.
7. Restart your Broker in debug mode.
8. Attempt multiple OpenLink ODBC, JDBC, UDBC, OLE-DB connections from the same and different machines.
9. Then navigate the Admin Assistant menu down the following path: "Server Components Administration"->"Request Broker Administration"->"List of Current connections" . This will show you how

many OpenLink Client connections that you have open and how many OpenLink Agents are serving these sessions.

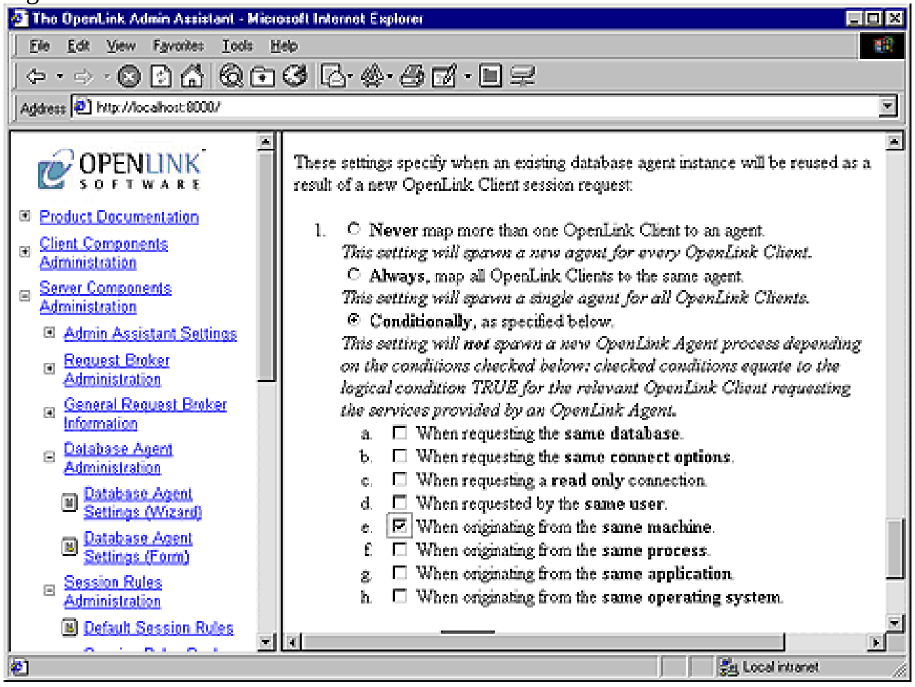
9.3.7.7.2 Conditionally Sharing One Database Agent Instance Across Numerous Data Access Clients

There may be times where you only want to conditionally pool connections from your OpenLink Clients against a particular OpenLink Agent instance. The steps that follow show you how to pool all OpenLink Client connections from a specific machine to a single Agent instance. Thereby creating a scenario in which an one OpenLink Agent instance is spawned per OpenLink Client machine.

The steps that follow guide you through the process of creating the appropriate session rule using the Admin Assistant:

1. Follow steps 1 -4 in the prior section.
2. Scroll down to the "Client-Server Process Mapping & Control section, then check the "Conditionally" radio button, then check the checkbox field labeled "When originating from the same machine" as depicted in the screen shot below.

Figure 7.55. Admin Assistant


The OpenLink Admin Assistant - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://localhost:8000/
OPENLINK SOFTWARE
Product Documentation
Client Components Administration
Server Components Administration
Admin Assistant Settings
Request Broker Administration
General Request Broker Information
Database Agent Administration
Database Agent Settings (Wizard)
Database Agent Settings (Form)
Session Rules Administration
Default Session Rules
Local intranet

These settings specify when an existing database agent instance will be reused as a result of a new OpenLink Client session request:

1. ○ **Never** map more than one OpenLink Client to an agent.
   This setting will spawn a new agent for every OpenLink Client.
   ○ **Always**, map all OpenLink Clients to the same agent.
   This setting will spawn a single agent for all OpenLink Clients.
   ◉ **Conditionally**, as specified below.
   *This setting will **not** spawn a new OpenLink Agent process depending on the conditions checked below: checked conditions equate to the logical condition TRUE for the relevant OpenLink Client requesting the services provided by an OpenLink Agent.*
   a. ☐ When requesting the **same database**.
   b. ☐ When requesting the **same connect options**.
   c. ☐ When requesting a **read only** connection.
   d. ☐ When requested by the **same user**.
   e. ☑ When originating from the **same machine**.
   f. ☐ When originating from the **same process**.
   g. ☐ When originating from the **same application**.
   h. ☐ When originating from the **same operating system**.

3. Click on the "Update" button to complete the rule creation exercise and then reinitialize the Request Broker when prompted.
4. Attempt multiple OpenLink ODBC, JDBC, UDBC, OLE-DB connections from the same and different machines.

Then navigate the Admin Assistant menu down the following path: "Server Components Administration"->"Request Broker Administration"->"List of current connections". This will show you how many OpenLink Client connections that you have open and how many OpenLink Agents are serving these sessions.

10 Chapter 8. Server Agent Administration

Abstract
Guide for successfully installing and running the OpenLink server components from the OpenLink Multi Tier Data Access Suite. This chapter describes the roles of the OpenLink Database/Service Agents and how to configure them. OpenLink Server Agents come in two main types, the Database agent and the Service agent. Database agents provide the only database dependant component, usually requiring a separate agent per database major version to connect to. Service agents provide other services such as proxying of connections or special interfaces to other ODBC data sources.

Table of Contents

OpenLink Database/Service Agents are the only database specific components within the OpenLink Multi-Tier middleware framework. It is these components that actually initiate and maintain database sessions with your OpenLink Clients, basically playing the role of a data access server.

Database Agents are servers implementing client data access interfaces such as ODBC, JDBC, UDBC, and OLE-DB using the lower level native call level interfaces provided by each supported database engine. These call level interfaces are themselves increasing based on the X/Open SQL Call Level Interface (CLI) specification. The older database engines that do not support or implement the X/Open SQL CLI specification simply provide traditional Embedded SQL interfaces.

The fact that Database Agents are built using natives interfaces provided by database engine vendors has some very important implications:

1. Through the eyes of a backend database, a database agent is a database client, no different to any other native client provided or bundled with the backend database engine.
2. A database agent inherits all of the functionality of a traditional database client, this includes: Distributed Database, SQL syntax, Stored Procedure support, and anything else that may be specific to the relevant backend database engine.
3. Configuring a database agent is no different to configuring an native database client, they share operating system environment variable dependencies etc..
4. Resource utilization is identical to resource consumption for native clients, this means that if you have special setting for your native client sessions, you will be able to apply these when configuring your database agents.
5. Any efficiencies or deficiencies in the database engines CLI client to database server inter process communications (IPC) also affects a database agent.

All database engines operate under a client-server paradigm, that is to say there are always two distinct processes involved in a database session, the database server and the database client. The database server must be up and running before a database client can communicate with a database. Database client and server processes may or may not reside on the same physical machine, at the same time this has no bearing on the fundamentals of database client and server process interaction as just explained.

10.1 Administration Assistant

Every database engine has a one or more key values that need to be set in order for database clients to be able to communicate with database servers. These values take the form of host operating system environment variables, database connection string formats/parameters, or a combination of both.

Configuring your OpenLink database agent is all about creating session initialization templates in the Sessions Rules Book which map key database client values with OpenLink Agent Template Attributes.

OpenLink provides a user friendly utility for configuring your database agents, namely the OpenLink Admin Assistant. Another option is to edit the rule book manually, but the availability of the utility makes this a less recommended option, certainly one for experienced OpenLink users only.

The preferred and much more flexible way of configuring your database agents is through the OpenLink Admin Assistant. This is a powerful HTML based GUI that is usable from any Web Browser, it provides you with two approaches to configuring database agents using Wizards or forms.

10.1.1 Wizard Based Administration

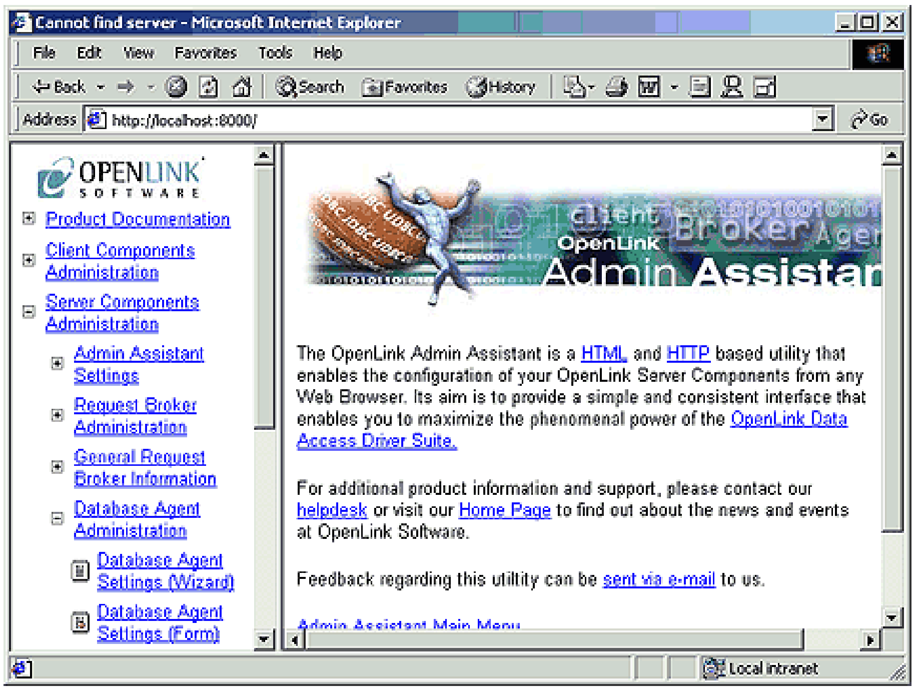
This approach to database agent administration is going to be exemplified using OpenLink Virtuoso as the database agent that is being administered. Please note that nothing in this example is database specific beyond the actual values entered. A database specific section follows which addresses these issues on database engine specific basis.

1. Start the Admin Assistant by entering the following URL into your browser: http://<IP Address or Network Alias of machine hosting the OpenLink Database Agents to be Administered>:<Admin Assistant Port number>

   For a network machine aliased as "mainserver" using the default OpenLink Admin Assistant port "8000" the URL required would be entered as follows: http://mainserver:8000

   You will be presented with a screen similar to the one below:

   Figure 8.1. Admin Assistant


Cannot find server - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://localhost:8000/
Go
OPENLINK SOFTWARE
Product Documentation
Client Components Administration
Server Components Administration
Admin Assistant Settings
Request Broker Administration
General Request Broker Information
Database Agent Administration
Database Agent Settings (Wizard)
Database Agent Settings (Form)
client Broker Agent
OpenLink Admin Assistant
Local intranet

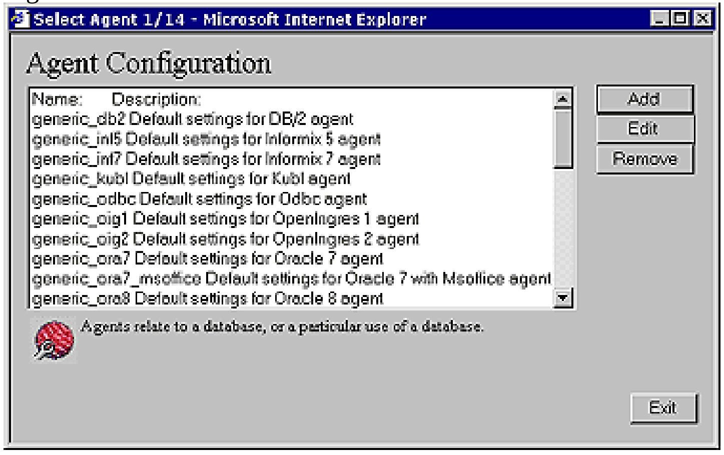
The OpenLink Admin Assistant is a HTML and HTTP based utility that enables the configuration of your OpenLink Server Components from any Web Browser. Its aim is to provide a simple and consistent interface that enables you to maximize the phenomenal power of the OpenLink Data Access Driver Suite.

For additional product information and support, please contact our helpdesk or visit our Home Page to find out about the news and events at OpenLink Software.

Feedback regarding this utiltity can be sent via e-mail to us.

2. Expand the "Server Components Administration", and "Database Agent Administration" menus. Once this is done you need to then click on the "OpenLink Database Agent Settings (Wizard)" hyperlink. This will bring you to the first Database Agent Wizard dialog as depicted below:

Figure 8.2. Admin Assistant

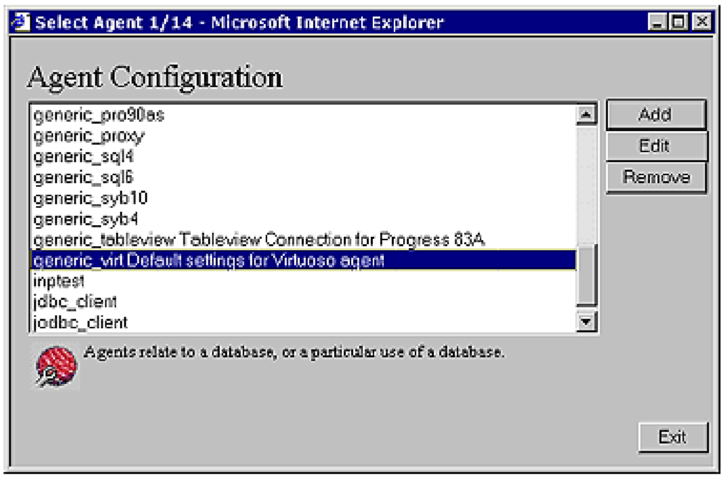


Name: Description:
generic_db2 Default settings for DB/2 agent
generic_inf5 Default settings for Informix 5 agent
generic_inf7 Default settings for Informix 7 agent
generic_kubl Default settings for Kubl agent
generic_odbc Default settings for Odbc agent
generic_oig1 Default settings for OpenIngres 1 agent
generic_oig2 Default settings for OpenIngres 2 agent
generic_ora7 Default settings for Oracle 7 agent
generic_ora7_msoffice Default settings for Oracle 7 with Msoffice agent
generic_ora8 Default settings for Oracle 8 agent

Agents relate to a database, or a particular use of a database.

3. From the list of pre-configured database agents select one that matches the database engine that your database agent will be communicating with, note that every database supported by OpenLink will have a pre-configured default agent initialization template listed.

Figure 8.3. Admin Assistant

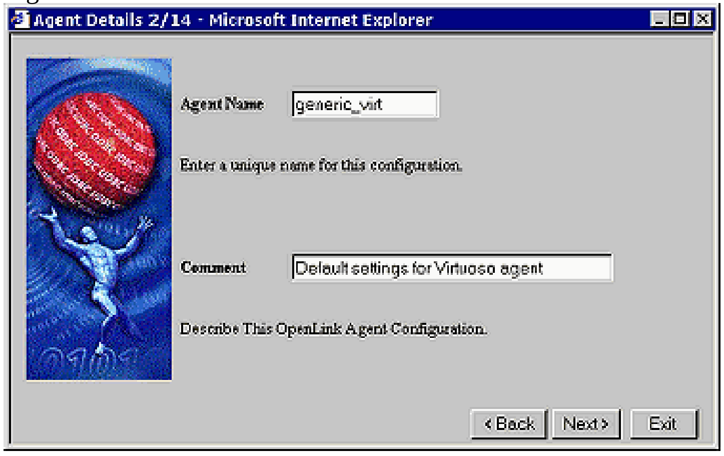

Select Agent 1/14 - Microsoft Internet Explorer

Agent Configuration

generic_pro90es
generic_proxy
generic_sql4
generic_sql6
generic_syb10
generic_syb4
generic_tableview Tableview Connection for Progress 83A
generic_virt Default settings for Virtuoso agent
inptest
jdbc_client
jodbc_client

Agents relate to a database, or a particular use of a database.

4. Click the "Edit" button to commence re configuring this database agent for your specific needs. Enter a name in the "Agent Name" field that is to be used to identify the database agent that your are configuring (it is recommended that you keep the default, if you do opt for a new name please don't enter names that contain spaces), also enter text into the "Comment" field describing your database agent. Once completed click on the "Next" button.

Figure 8.4. Admin Assistant

Agent Details 2/14 - Microsoft Internet Explorer

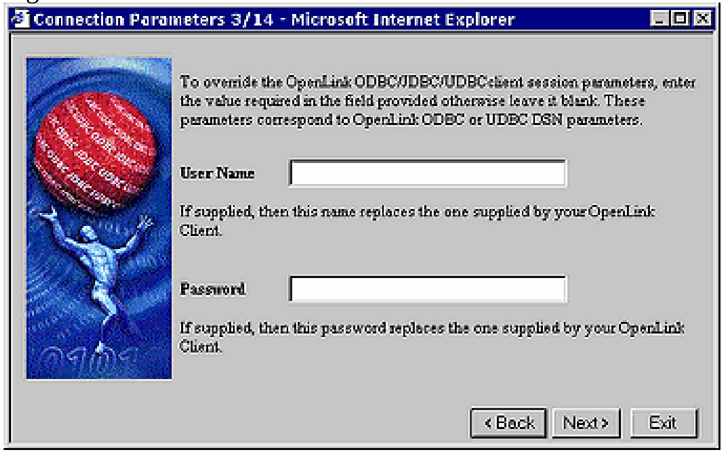
Agent Name: generic_virt

Enter a unique name for this configuration.

Comment: Default settings for Virtuoso agent

Describe This OpenLink Agent Configuration.

5. You can force your OpenLink clients to connect a backend database using a pre-assigned username and password combination, this is done by entering the values that you want to enforce into the "User Name" and "Password" fields in the Wizard dialog depicted below. This activity implies that irrespective of what username password combination is entered at OpenLink client configuration or connect time, the values that you provide will take precedence. If you do not want to enforce username and password combinations for your OpenLink clients simply leave the "User Name" and "Password" fields blank. Click on the "Next" button to continue.

Figure 8.5. Admin Assistant

Connection Parameters 3/14 - Microsoft Internet Explorer

To override the OpenLink ODBC/JDBC/UDBC client session parameters, enter the value required in the field provided otherwise leave it blank. These parameters correspond to OpenLink ODBC or UDBC DSN parameters.

User Name:

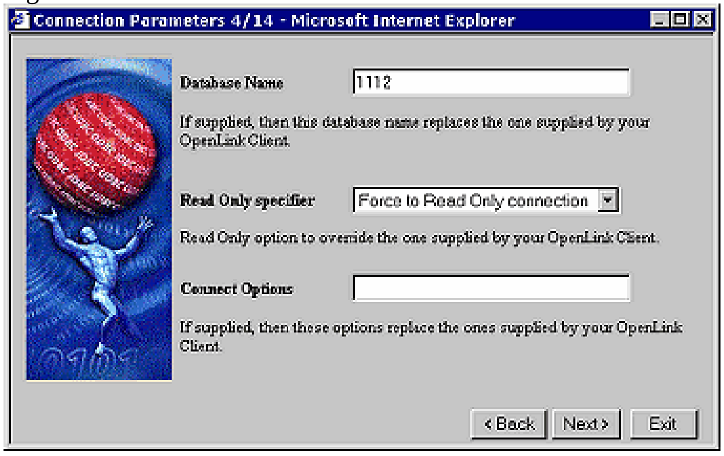
If supplied, then this name replaces the one supplied by your OpenLink Client.

Password:

If supplied, then this password replaces the one supplied by your OpenLink Client.

6. Enter a value in the "Database Name" field that identifies the actual database within your database engine's environment that you want to connect your OpenLink clients with, the format of these values vary by database engine type. You can control what type of session(s) your OpenLink clients conduct with your backend database by choosing the appropriate value from the "Read Only Specifier" listbox. If you are connecting to a database that does not reside on the same machine as your database agent, or circumstances require you to use the networking middleware provided by your database vendor (which is already installed on the same machine as the database agent), then you can enter the remote database connection values required by your remote database into the "ConnectOptions" field. Once completed click the "Next" button to continue.

   Figure 8.6. Admin Assistant

   Connection Parameters 4/14 - Microsoft Internet Explorer

   Database Name: 1112

   If supplied, then this database name replaces the one supplied by your OpenLink Client.

   Read Only specifier: Force to Read Only connection

   Read Only option to override the one supplied by your OpenLink Client.

   Connect Options:

   If supplied, then these options replace the ones supplied by your OpenLink Client.

7. Some database engines allow their clients to configure session resources, in situations where this holds true you can use the "Server Options" field to set these values for your database agent sessions.

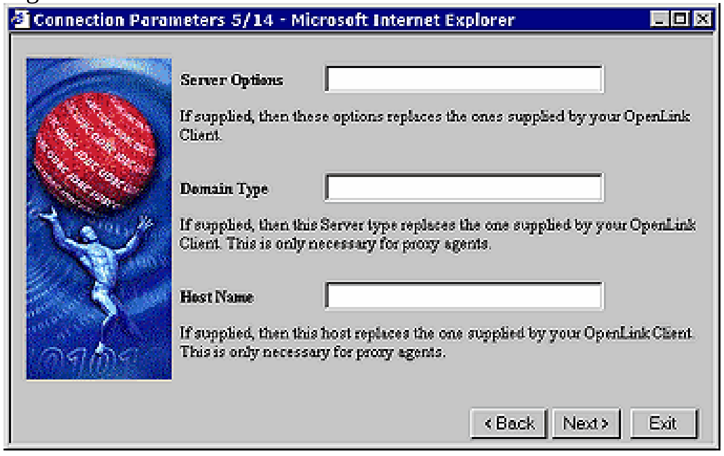
   You can also configure your database agents such that the database agent that services OpenLink clients acts as a PROXY to other database agents residing on a remote OpenLink server(s) within your network infrastructure. This is basically a 3-Tier or N-Tier Distributed Database Agent configuration and the key configuration values go into the "Domain Type" and "Host Name" fields.

   Domain Type - this is your OpenLink agent Server type (Virtuoso, Oracle. Informix, ODBC, PROXY, etc..), in the case of a typical 3-tier configuration you would enter the value PROXY, this implies that you are going to a PROXY agent rather than an database agent.

   Host Name - this is how you identify the machine hosting the Request Broker that binds your OpenLink clients with the OpenLink agent designated by the value entered into the "Domain Type" field.

   In most case all of these fields can simply be left blank. Click on the "Next button to continue.

   Figure 8.7. Admin Assistant

   Connection Parameters 5/14 - Microsoft Internet Explorer

   Server Options:

   If supplied, then these options replaces the ones supplied by your OpenLink Client.

   Domain Type:

   If supplied, then this Server type replaces the one supplied by your OpenLink Client. This is only necessary for proxy agents.

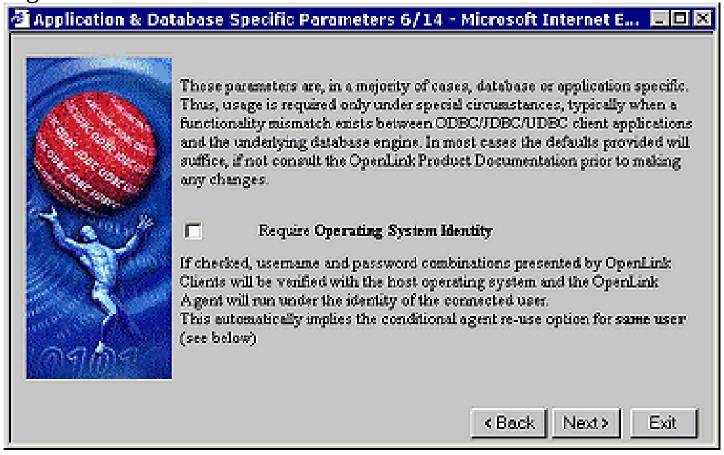
   Host Name:

   If supplied, then this host replaces the one supplied by your OpenLink Client. This is only necessary for proxy agents.

8. For security purposes there are times when you want to validate your OpenLink clients at the operating system level before they actually initiate sessions with your backend database. In some cases backend databases presume that your ability to attempt database connection implies your being a valid operating system users, this

can be a major security hole for those database engines that do not conduct their own username and password verification. By hatching the "Require Operating System Identity" checkbox your database agent will validate OpenLink clients at the operating system level before attempting a connection to your backend database. Click the "Next" button to continue.

Figure 8.8. Admin Assistant



Application & Database Specific Parameters 6/14 - Microsoft Internet E...

These parameters are, in a majority of cases, database or application specific. Thus, usage is required only under special circumstances, typically when a functionality mismatch exists between ODBC/JDBC/UDBC client applications and the underlying database engine. In most cases the defaults provided will suffice, if not consult the OpenLink Product Documentation prior to making any changes.

☐ Require **Operating System Identity**

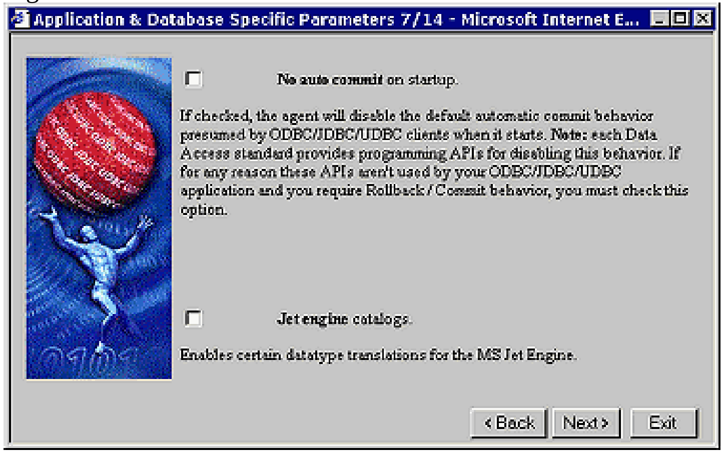
If checked, username and password combinations presented by OpenLink Clients will be verified with the host operating system and the OpenLink Agent will run under the identity of the connected user.
This automatically implies the conditional agent re-use option for **same user** (see below)

9. You can enforce consistent transaction behavior across your OpenLink clients through the "No Auto Commit On Startup" field, this ensures that database commands originating from OpenLink clients do not contain trailing "Commit" instructions. Without this restriction database integrity could easily be compromised by incomplete transactions creating broken records. You enable this protection by hatching the "No Auto Commit On Startup" checkbox.

   You can also enforce application specific data type translation handling specifically for Microsoft Jet Engine based OpenLink ODBC clients by hatching the "Jet Engine Catalogs" checkbox. This is a very application specific feature that is only required when Microsoft's Jet engine is the OpenLink ODBC client.

   Click the "Next" button to continue.

   Figure 8.9. Admin Assistant



Application & Database Specific Parameters 7/14 - Microsoft Internet E...

☐ **No auto commit** on startup.

If checked, the agent will disable the default automatic commit behavior presumed by ODBC/JDBC/UDBC clients when it starts. **Note:** each Data Access standard provides programming APIs for disabling this behavior. If for any reason these APIs aren't used by your ODBC/JDBC/UDBC application and you require Rollback / Commit behavior, you must check this option.

☐ **Jet engine** catalogs.

Enables certain datatype translations for the MS Jet Engine.

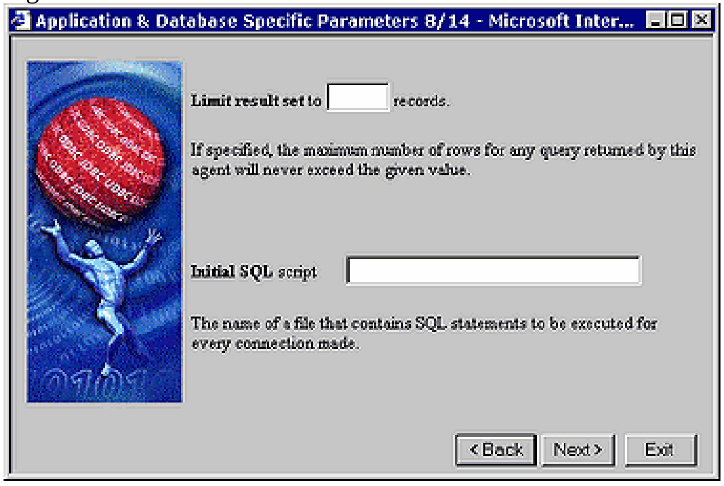
10. There are times when you may want to restrict the number of records that your database agent transmits to your OpenLink clients. In such scenarios you can enter a numeric value into the "Limit result set" field that represents the maximum number of records from a database query resultset that your database agent should transmit back to your OpenLink clients. This features protects your network from "Innocent Queries From Hell", these are queries that end-user unknowingly generate when using visual query tools, especially as they familiarize themselves with the concepts of SQL querying.

    There are also times when critical database functionality may not be implemented as part of the ODBC. JDBC., UDBC,or OLE-DB specifications, but you need to make use of such functionality in order to run your database infrastructure smoothly. When this situation arises you are able to use the "Initial SQL script" field to enter a value that points to a script file containing a set of SQL instructions that implemented the desired functionality. This field corresponds to specifying "CommandLine = +initsql filename" in the rulebook (oplrqb.ini).

In some cases you may find yourself having to deal with the fact that although the functionality may be implemented at the ODBC, UDBC, JDBC, OLE-DB specification level, the client application connecting to your database via OpenLink simply is not making use of this functionality. In this scenario the "Initial SQL Script" field comes in handy. A typical example is default Transaction Isolation Levels handling.

Click "Next" to continue.

Figure 8.10. Admin Assistant


Application & Database Specific Parameters 8/14 - Microsoft Inter...

Limit result set to records.

If specified, the maximum number of rows for any query returned by this agent will never exceed the given value.

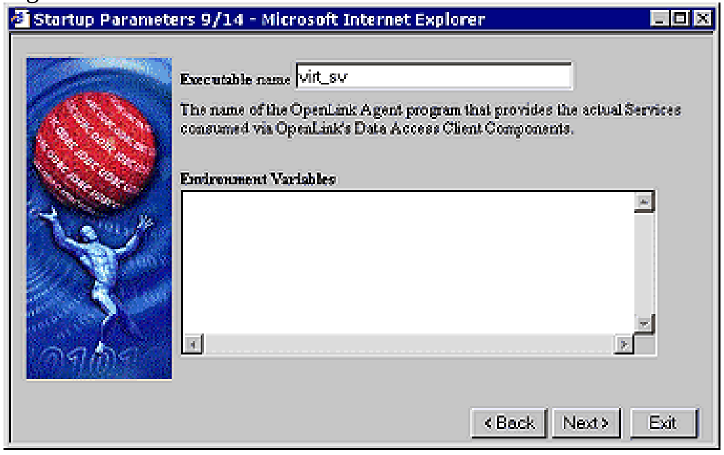
Initial SQL script

The name of a file that contains SQL statements to be executed for every connection made.

11. You identify the binary executable file that represents the specific database agent for your backend database by entering the binary files name in the "Executable Name" field. This set for you by default and unless you rename this file yourself it does not need to be changed.

    Many database environments are driven by operating system level environment variables, the "Environment Variables" field allows you to set or re-configure these values (see database specific settings section). Click the "Next" button to continue.

    Figure 8.11. Admin Assistant

    
    Startup Parameters 9/14 - Microsoft Internet Explorer

    Executable name virt_sv

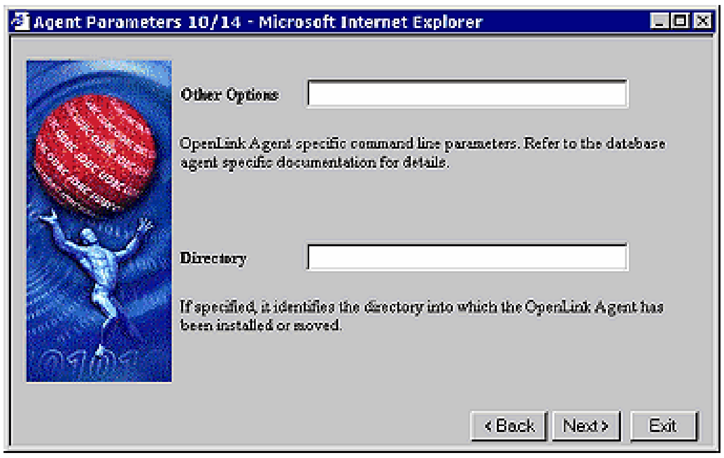
    The name of the OpenLink Agent program that provides the actual Services consumed via OpenLink's Data Access Client Components.

    Environment Variables

12. There may be some database agent specific options that you need to apply specifically to sessions with your backend database (see usage part of "obtaining database agent information" section for list of values), you enter these values into the "Other Options" field.

    You also need to indicate the directory in which your database agent executable binary file resides, this is only required if you have moved these files from their default location after installation. Click the "Next Button" to continue.

    Figure 8.12. Admin Assistant


Agent Parameters 10/14 - Microsoft Internet Explorer

Other Options

OpenLink Agent specific command line parameters. Refer to the database agent specific documentation for details.

Directory

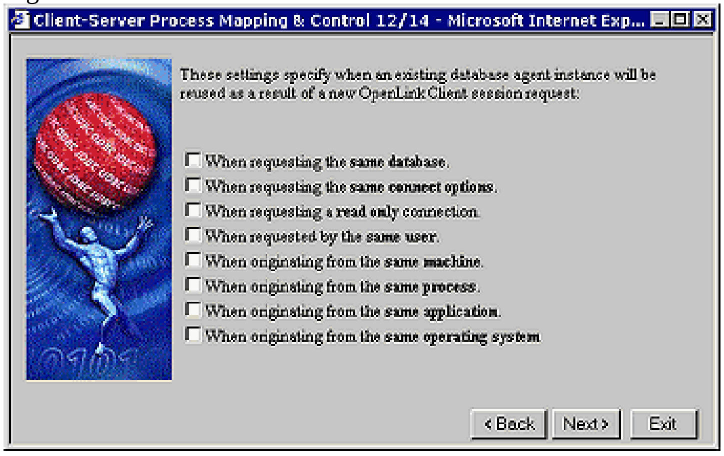
If specified, it identifies the directory into which the OpenLink Agent has been installed or moved.

13. Your database agents act as servers to your OpenLink clients, this implies that as server processes they do consume server operating system resources. You can control resource consumption by predefining how many database agent instances are to be started as a result of OpenLink clients connections, and the basis upon which these instance are reused by subsequent OpenLink client connections. The "Never", "Always", and "Conditionally" radio button allows you to choose the option that best suits your infrastructure's needs.

    The "Accept Up To" field allows you to enter a numeric value that indicates the maximum number of new database agent processes that can be instantiated on a given OpenLink server operating environment.

    If you choose the "Conditionally" radio button and then click the "Next" button you will be presented with an additional dialog with a list of checkboxes. These checkboxes allow you to customize the sets of circumstances under which you want an OpenLink client connection to result in the instantiation of a new database agent instances. Click the next "Button" to continue.

    Figure 8.13. Admin Assistant

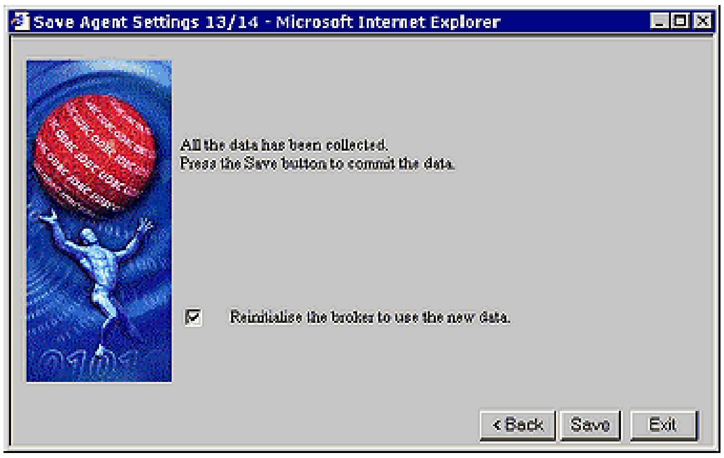

Client-Server Process Mapping & Control 12/14 - Microsoft Internet Exp...

These settings specify when an existing database agent instance will be reused as a result of a new OpenLink Client session request:

- ☐ When requesting the same **database**.
- ☐ When requesting the same **connect options**.
- ☐ When requesting a **read only** connection.
- ☐ When requested by the same **user**.
- ☐ When originating from the same **machine**.
- ☐ When originating from the same **process**.
- ☐ When originating from the same **application**.
- ☐ When originating from the same **operating system**

14. At the end of this Wizard interaction you can opt to make you database agent settings available to the next OpenLink client without shutting down and restarting the Request Broker, you do this by hatching the "Reinitialize" checkbox and then clicking the "Save" button to save your settings.

    Figure 8.14. Admin Assistant



All the data has been collected.
Press the Save button to commit the data.

☑ Reinitialise the broker to use the new data.

10.1.2 Forms Based Administration

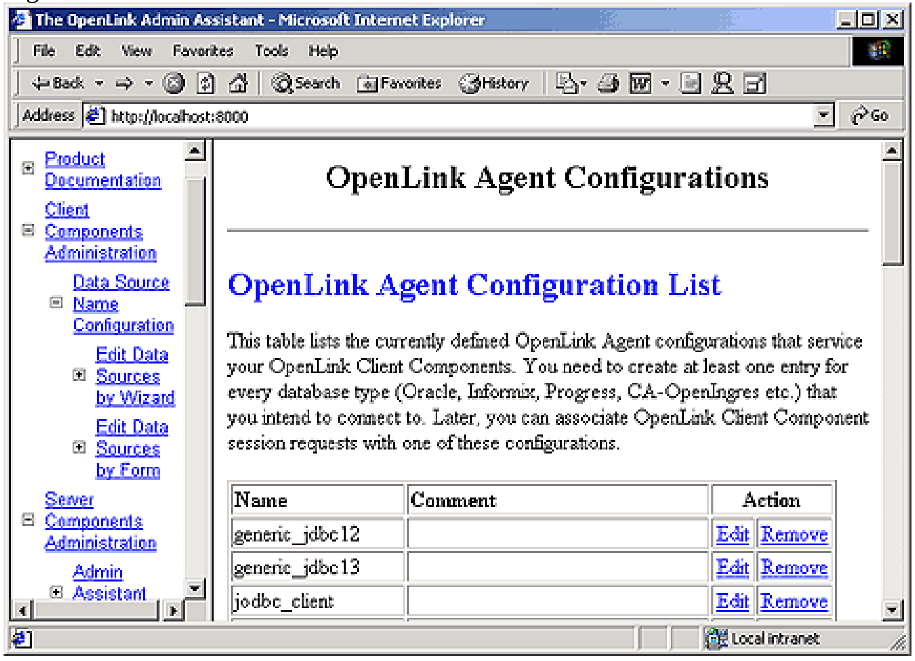
You can also administer database agents through the Admin Assistant using a "Forms" as opposed to "Wizards" based approach, the steps that follow guide you through this process using the same example (configuring the a database agent for the OpenLink Virtuoso database).

1. Start the Admin Assistant by entering the following URL into your browser: http://<IP Address or Network Alias of machine hosting the OpenLink Database Agents to be Administered>:<Admin Assistant Port number>

   For a network machine aliased as "mainserver" using the default OpenLink Admin Assistant port "8000" the URL required would be entered as follows: http://mainserver:8000

   Expand the menus for "Server Components Administration", and "Database Agent Administration". Once this is done you need to then click on the "Database Agent Settings (Form)" hyperlink. This will bring you to a database agent listing as depicted below:

   Figure 8.15. Admin Assistant


The OpenLink Admin Assistant - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://localhost:8000
Go
Product Documentation
Client Components Administration
Data Source Name Configuration
Edit Data Sources by Wizard
Edit Data Sources by Form
Server Components Administration
Admin Assistant
Local intranet

OpenLink Agent Configurations

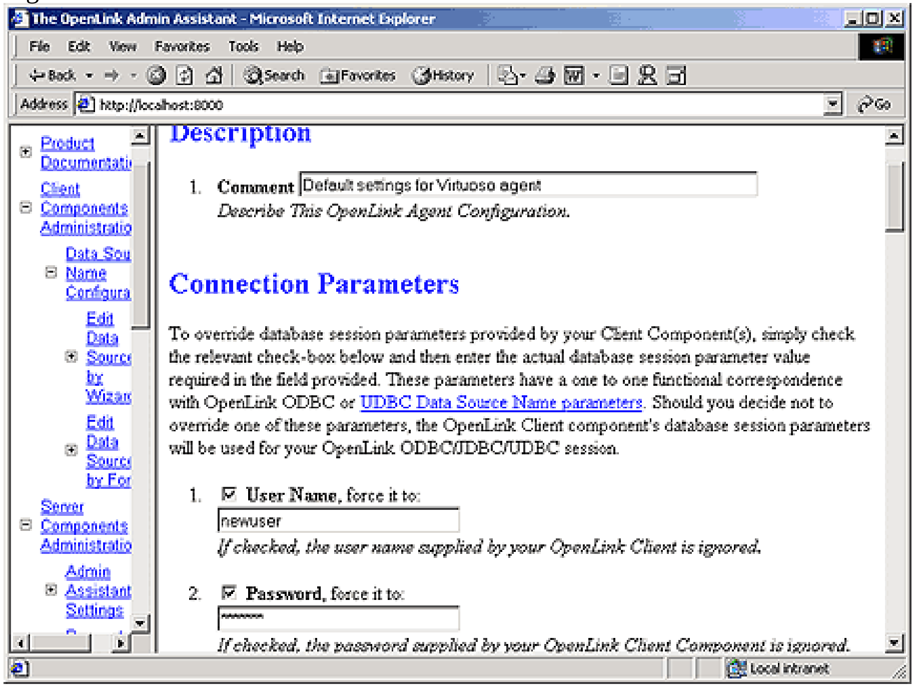
OpenLink Agent Configuration List

This table lists the currently defined OpenLink Agent configurations that service your OpenLink Client Components. You need to create at least one entry for every database type (Oracle, Informix, Progress, CA-OpenIngres etc.) that you intend to connect to. Later, you can associate OpenLink Client Component session requests with one of these configurations.

Name	Comment	Action	
generic_jdbc12		Edit	Remove
generic_jdbc13		Edit	Remove
jodbc_client		Edit	Remove

2. As our example for this exercise is based on the OpenLink Virtuoso database agent, click on the "Edit" hyperlink for the "generic_virt" default database agent (note you can substitute this with the default database agent that matches your backend database). Once these actions are completed you will be presented with the main database agent initialization template form. Complete the values for the fields that apply to needs and then click on the "Update" button at the bottom of the form to save your changes.

Figure 8.16. Admin Assistant


The OpenLink Admin Assistant - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://localhost:8000
Go
Product Documentati
Client Components Administratio
Data Sou Name Configura
Edit Data Source by Wizar
Edit Data Source by For
Server Components Administratio
Admin Assistant Settings
Local intranet

Description

1. Comment Default settings for Virtuoso agent
Describe This OpenLink Agent Configuration.

Connection Parameters

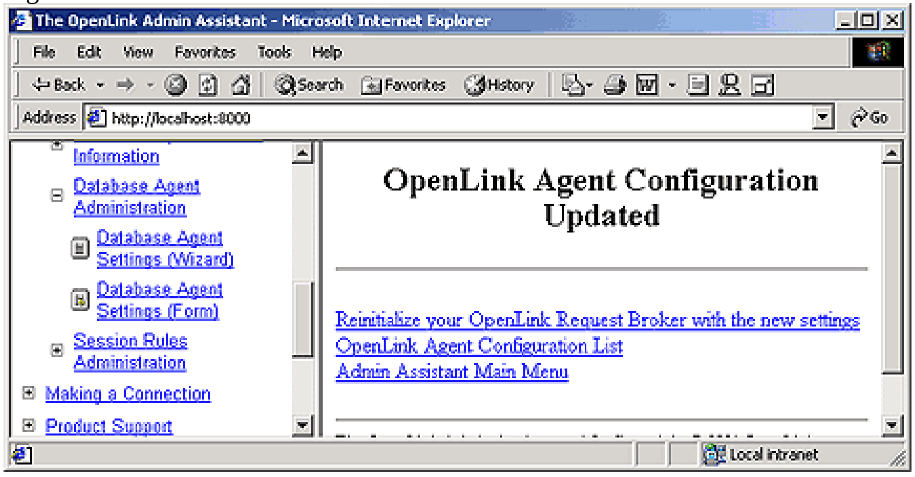
To override database session parameters provided by your Client Component(s), simply check the relevant check-box below and then enter the actual database session parameter value required in the field provided. These parameters have a one to one functional correspondence with OpenLink ODBC or UDBC Data Source Name parameters. Should you decide not to override one of these parameters, the OpenLink Client component's database session parameters will be used for your OpenLink ODBC/JDBC/UDBC session.

1. ☑ User Name, force it to: newuser
If checked, the user name supplied by your OpenLink Client is ignored.

2. ☑ Password, force it to: *******
If checked, the password supplied by your OpenLink Client Component is ignored.

3. Click on the "Reinitialize Your OpenLink Request Broker With New Settings" hyperlink, this enables your new setting to be applied to subsequent connections from OpenLink clients without disrupting existing OpenLink client sessions.

Figure 8.17. Admin Assistant


The OpenLink Admin Assistant - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://localhost:8000
Go
Information
Database Agent Administration
Database Agent Settings (Wizard)
Database Agent Settings (Form)
Session Rules Administration
Making a Connection
Product Support
Local intranet

OpenLink Agent Configuration Updated

Reinitialize your OpenLink Request Broker with the new settings
OpenLink Agent Configuration List
Admin Assistant Main Menu

10.2 Agent-Specific Settings

OpenLink database agents are database clients built using the SQL Call Level or Embedded SQL interfaces of the respective supported backend database engines. Thus, the process of configuring or administering a database agent is similar in essence to what you would have to do if you were administering a native database client.

Database engines use environment variables to creating a database specific operating space within which database clients and servers interact, these environment typically address the following important database session related issues:

- Database server identification - your database client needs to be able to connect to the appropriate database server, many database implementations support multiple database servers instances listening for client connections at different network ports on the same machine (e.g. OpenLink Virtuoso, Microsoft SQL Server, Sybase, Progress, Informix etc.).
- Database engine base installation directory - many database engine environments comprise shared or dynamically linked libraries and other runtime components that are required by database clients. Thus, there is a need for database clients to have a sense of what the actual base or root point of the database engine installation

is, this enables the construct of a component search path (similar to the "PATH" operating system environment variable) at runtime.

- Database session resource allocation - most database engines allow database session resources to be configured for clients via environment variables (sometimes these variables simply identify resource configuration files).
- Database session optimization - some database environments allow query optimizers, network packet sizes, and records transmission values for database clients to be configured via environment variables.

The sections that follow address specific environment settings that affect the configuration of your OpenLink database agents, the values provided can supplanted values used in the Admin Assistant configuration examples provided in the prior section.

10.2.1 Common Agent Configuration Options

The following parameters exist in all Agent Environment sections of the oplrqb.ini

Table 8.1. Default Rule Book settings

Rule Book Section & parameters	Default Rule Book Settings	Notes
[Environment xyz]		Any agent section
CURSOR_SENSITIVITY=	LOW	Set to HIGH after loading oplrvc.sql

10.2.1.1 High Cursor Sensitivity

Enables or disables the row version cache used with dynamic cursors.

When dynamic cursor sensitivity is set high, the Cursor Library calculates checksums for each row in the current rowset and compares these with the checksums (if any) already stored in the row version cache for the same rows when fetched previously. If the checksums differ for a row, the row has been updated since it was last fetched and the row status flag is set to SQL_ROW_UPDATED. The row version cache is then updated with the latest checksums for the rowset.

From the user's point of view, the only visible difference between the two sensitivity settings is that a row status flag can never be set to SQL_ROW_UPDATED when the cursor sensitivity is low. (The row status is instead displayed as SQL_ROW_SUCCESS.) In all other respects, performance aside, the two settings are the same - deleted rows don't appear in the rowset, updates to the row since the row was last fetched are reflected in the row data, and inserted rows appear in the rowset if their keys fall within the span of the rowset.

If your application does not need to detect the row status SQL_ROW_UPDATED, you should leave the 'High Cursor Sensitivity' checkbox unchecked, as performance is improved. The calculation and comparison of checksums for each row fetched carries an overhead.

If this option is enabled, the table oplrvc must have been created beforehand using the appropriate oplrvc.sql script for the target database.

10.2.2 Virtuoso

When configuring a Virtuoso database agent the critical configuration items are:

- Database Identification - this is the Virtuoso Database Server's port number, which identifies the actual Virtuoso server process that links you to an actual Virtuoso database file. This is the value that you enter into the "Database Name" field of either your Admin Assistant form or wizard dialog.

10.2.2.1 Application Server & 3-Tier Architecture Configuration

There may be situations in which you are unable to install your OpenLink Request Broker and Database Agents on the same machine as the one hosting your Virtuoso database server. Irrespective of the reasons that lead you to this scenario, it is possible to configure your OpenLink database agents hosted on your Application Server machine such that they connect to a remote Virtuoso database on your Database Server machine using Virtuoso's database specific networking as opposed to OpenLink's Database Independent Networking. The end result being a 3-tier distributed OpenLink architecture in which the communication between OpenLink clients and database agents use OpenLink's Database independent Networking, while the communication between the Virtuoso database agent and the Virtuoso database server uses Virtuoso database specific networking.

Configuration Steps:

Assuming you have a Virtuoso Database Server called "mainserver2" that has a Virtuoso server process listening for clients at port 1112

1. Ensure that you have a usable connection to Virtuoso using its native networking.
2. Add the following value to the "Database Name" field within the Admin Assistant Forms or Wizards used to configure your database agent. If you choose to set this value on the client simply enter the same values into the "Database Name" Attribute associated with the configuration of your OpenLink client (see OpenLink ODBC or JDBC or UDBC client configuration for more details):

```
mainserver2:1112
```

10.2.3 Informix

When configuring an Informix database agent the critical configuration items are:

- Database Identification - this is an actual database name e.g "stores7", which identifies the actual Informix database file that you want to be connected with. This is the value that you enter into the "Database Name" field of either your Admin Assistant's database agent configuration form or wizard dialog. If you choose to have database identification take place at the client rather than server, you enter this value into the "Database Name" field or connection attribute when configuring your OpenLink client.

Informix provides a number of environment variables for configuring database clients, the basic set required for successfully connecting your database agent to an Informix database server are tabulated below:

Table 8.2. Default Rule Book settings

Rule Book Section & parameters	Default Rule Book Settings	Notes
[Environment INFORMIX5]		Informix 5 Agent environment settings
INFORMIXDIR=	/dbs/informix5	Full path to the base directory for the Informix 5 installation.
TBCONFIG=	tbconfig	Database server configuration file
FORCE_ONLINE_DATABASE=	1	Force mode to (0) SE or (1) ONLINE
[Environment INFORMIX6]		
INFORMIXDIR=	/dbs/informix6	Full path to the base directory for the Informix 6 installation.
[Environment INFORMIX7]		
INFORMIXDIR=	/dbs/informix7	Full path to the base directory for the Informix 7 installation.
INFORMIXSERVER=	online7	The name of Informix 7 server that you want the agent to attach to. As long as you have I-Connect or I-Net installed, configured and up and running this value can connect your database agent with remote Informix database servers.
ONCONFIG=	ONCONFIG	Database server configuration file
FORCE_ONLINE_DATABASE=	1	Force mode: 0 for Standard Engine (SE) 1 for ONLINE
DELIMIDENT=	Y	This will allow quoted identifiers to be handled.
OPL_INF_MULTISESS=	Y	Enables Informix mutlisession mode. Y=Each ODBC connection gets its own database session. N=Each ODBC connection shares one database connection.
OPL_SPACEPADCHAR=	Y	Character data from a CHAR column is fetched with trailing spaces retained. The column data is padded upto the column width. N=Trailing spaces are stripped off.

10.2.3.1 Security Enhancement

Due to the fact that Informix leaves username and password verification to the host operating system, it is possible to close what could be an ODBC, UDBC, JDBC, or OLE-DB security loophole by utilizing the OpenLink database agent "OpsysLogin" facility which can be enabled through the Admin Assistant. By enabling this feature your Informix database agent will verify user accounts at the operating system level before attempting to connect to your Informix database. It is important to note that "super-user" or Administrator (depending on operating system) privileges are required to successfully use this feature. This implies that the account that starts the request broker must possess one of the aforementioned system level privileges, on the other hand these privileges aren't required for your actual OpenLink client sessions.

10.2.3.2 Rebuilding Informix Database agents

OpenLink provides a relinkable library and script files that enable you to rebuild your database agents as shared, as opposed to statically linked binaries, or for the purposes of getting a closer database implementation fit, should your Informix database environment be a more recent release than the actual version used by OpenLink to build the database agent installed on your system. Please read the Relinking OpenLink Database Agents section that follows, for details on how to perform this task.

Note that to run the Informix agents you may require the latest Informix Connect (a free download from the IBM/Informix site); in order to relink them, you require the Client SDK as well.

10.2.3.3 Application Server & 3-Tier Architecture Configuration

There may be situations in which you are unable to install your OpenLink Request Broker and Database Agents on the same machine as the one hosting your Informix database server. Irrespective of the reasons that lead you to this scenario, it is possible to configure your OpenLink database agents hosted on your Application Server machine such that they connect to a remote Informix database on your Database Server machine using Informix database specific networking (I-Connect or I-Net) as opposed to OpenLink's Database Independent Networking. The end result being a 3-tier distributed OpenLink architecture in which the communication between OpenLink clients and database agents use OpenLink Database independent Networking, while the communication between the Informix database agent and the Informix database server uses I-Connect or I-Net (depending on Informix version).

Configuration Steps:

Assuming you have an Informix Database Server machine called "mainserver2" that has an Informix I-Connect or I-Net server process running (this is setup and configured via the SQLHOSTS file on the database server machine).

1. On your Application Server (the machine hosting your database agent) create an I-connect or I-Net Connection Alias called "mainserver2" (for purpose of this example only) if a working Connection Alias doesn't already exist on this machine.
2. Ensure that you have a usable connection to your remote Informix database using Connection Alias "mainserver2".
3. Add the following values to the "Database Server Options" field within the Admin Assistant Forms or Wizards used to configure your database agent. If you choose to set this value on the client simply enter the same value into to the "Database Name" Attribute associated with the configuration of your OpenLink client (see OpenLink ODBC or JDBC or UDBC client configuration for more details):

   ```
   mainserver2
   ```

   You can also set the INFORMIXSERVER environment variable to "mainserver2".

10.2.4 See Also:

Application Server Architecture for various illustrations of distributed client-server architectures supported by OpenLink database agent

10.2.4.1 Agent handling of SQLGetInfo

The Informix agent is hard coded with responses to SQLGetInfo(). These are affected by the build version:

Table 8.3. SQLGetInfo() differences

Parameter	**Build Version <= 7.x**	**Build Version > 7.x (eg. 9.x)**
SQL_MAX_CATEGORY_NAME_LEN	8	32
SQL_MAX_OWNER_NAME_LEN	8	32
SQL_MAX_SCHEMA_NAME_LEN	18	128
SQL_MAX_QUALIFIER_NAME_LEN	18	128
SQL_EXPRESSIONS_IN_ORDERBY	N	N

10.2.4.2 Unicode Configuration

See the Unicode section for configuration details.

10.2.5 Ingres

When configuring an Ingres database agent the critical configuration items are:

- Database Identification - this is an actual database name e.g "demo", which identifies the actual Ingres database that you want to be connected with. This is the value that you enter into the "Database Name" field of either your Admin Assistant's database agent configuration form or wizard dialog. If you choose to have database identification take at the client rather than server, you enter this value into the "Database Name" field or connection attribute when configuring your OpenLink client.
- User Authentication - To log into OpenIngres | Ingres II databases you need to create an internal user entry in the database which matches the userid that starts the OpenLink Ingres Database agent (oig[1|2]_sv). Thus if your userid is openlink you need to create a user openlink that is capable of switching to another internal user. A default OpenIngres | Ingres II database will have entries for ingres and root. You can also enable the OpenLink 'opsyslogin' feature in the OpenLink Rule Book (oplrqb.ini) that will force the userid of the agent to be changed to that of the username that has been authenticated by the Operating system, and assuming this is a valid Ingres user will be allowed to connect by the database.

Ingres provides a number of environment variables for configuring database clients, the basic set required for successfully connecting your database agent to an Ingres database server are tabulated below:

Table 8.4. Default Rule Book settings

Rule Book Section & parameters	Default Rule Book Settings	Notes
[Environment OPENINGRES1] [Environment INGRES_II]		Parameters are common in Open Ingres 1 and Ingres II
II_DATE_FORMAT=	US	Defines the output format for dates as dd-mmm-yyyy. This should not be changed inside the Rule Book since it enables the best compatibility with OpenLink. This will not affect any other Progress applications.
II_SYSTEM=	/dbs	Full path to the directory immediately below the Progress/ directory e.g. if your Progress installation directory is /dbs/Progress then set this to /dbs
ING_SET=	set lockmode session where readlock=nolock	This makes sure when the Ingres agent is started, the first thing it does is runs this command. The command itself makes sure that a simple select statement does not lock all the rows it selects.
USE_OWNER=	Y or N	Only applicable for Ingres 6.4 databases. If set to Y forces the driver to return the ingres internal table owner as the schema name for the table, as this was required for early version of MSQuery.

10.2.5.1 Security Enhancement:

Due to the fact that Ingres 6.4 leaves username and password verification to the host operating system (Ingres II does not have this problem), it is possible to close what could be an ODBC, UDBC, JDBC, or OLE-DB security loophole by utilizing the OpenLink database agent "OpsysLogin" facility which can be enabled through the Admin Assistant. By enabling this feature your Ingres database agent will verify user accounts at the operating system level before attempting to connect to your Ingres database. It is important to note that "super-user" or Administrator (depending on operating system) privileges are required to successfully use this feature. This implies that the account that starts the request broker must possess one of the aforementioned system level privileges, on the other hand these privileges aren't required for your actual OpenLink client sessions.

10.2.5.2 Rebuilding Ingres Database Agents

OpenLink provides a relinkable library and script files that enable you to rebuild your database agents as shared, as opposed to statically linked binaries, or for the purposes of getting a closer database implementation fit, should your Ingres database environment be a more recent release than the actual version used by OpenLink to build the database agent installed on your system. Please read the Relinking OpenLink Database Agents section that follows, for details on how to perform this task.

10.2.5.3 Application Server & 3-Tier Architecture Configuration

There may be situations in which you are unable to install your OpenLink Request Broker and Database Agents on the same machine as the one hosting your Ingres database server. Irrespective of the reasons that lead you to this scenario, it is possible to configure your OpenLink database agents hosted on your Application Server machine such that they connect to a remote Ingres database on your Database Server machine using Ingres database specific networking (Ingres Net) as opposed to OpenLink's Database Independent Networking. The end result being a 3-tier distributed OpenLink architecture in which the communication between OpenLink clients and database agents use OpenLink Database independent Networking, while the communication between the Ingres database agent and the Ingres database server uses Ingres Net.

Configuration Steps:

Assuming that you have an Ingres Database server machine called "mainserver2" that has an Ingres Net server process running.

1. On your Application Server (the machine hosting your database agent) create an Ingres Net vnode called "mainserver2" (for purpose of this example only) if you do not have a working vnode on this machine.
2. Ensure that you have a usable connection to your remote Ingres database using the vnode "mainserver2".
3. Add the following values to the "Server Options" field within the Admin Assistant Forms or Wizards used to configure your database agent. If you choose to set this value on the client simply enter the same value into to the "Database Name" Attribute associated with the configuration of your OpenLink client (see OpenLink ODBC or JDBC or UDBC client configuration for more details):

```
mainserver2
```

10.2.6 See Also:

Application Server Architecture for various illustrations of distributed client-server architectures supported by OpenLink database agent

10.2.7 Note:

OpenLink Drivers for Ingres II, both Single- and Multi-Tier, are based on the OpenAPI Interface, and are intended only for use only with Ingres II. In some environments, these Drivers may also permit connection to OpenIngres 1.x and 2.x, which are also based on the OpenAPI Interface; however, these Drivers have not been certified for such connections, and they are made at the User's discretion and risk. These Drivers will not support connections to Ingres 6.4 or previous under any conditions. Drivers for versions of Ingres prior to Ingres II may be available for some platforms; please inquire at product availability, if you require such a Driver.

10.2.8 Progress

When configuring a Progress database agent the critical configuration items are:

- Database Identification - this is an actual database name e.g "demo" or "isports", which identifies the actual Progress database file that you want to be connected with. This is the value that you enter into the "Database Name" field of either your Admin Assistant form or wizard dialog. If you choose to have database identification take at the client rather than server, you enter this value into the "Database Name" field or connection attribute when configuring your OpenLink client.

Progress provides a number of environment variables for configuring database clients, the basic set required for successfully connecting your database agent to an Progress database server are tabulated below:

Table 8.5. Default Rule Book settings

Rule Book Section & parameters	Default Rule Book Settings	Notes
[Environment PROGRESS8]		
DLC=	/dbs/dlc8	Must be full path to the Progress dlc directory.
PROCFG=	/dbs/dlc8/progress.cfg	Must be the full path and filename to the progress.cfg file. This parameter is optional. Use it if the license file can not be found.
TABLEVIEW=		Must be the full path and filename to the table view file (tableview.dat). See detailed TABLEVIEW document for more information
TABLEVIEW_QUALIFIER=	Y	Add owner information to select statement
DEADLOCK_TIMEOUT=	60	Seconds to wait for lock to release.
INSERT_LOGICAL=	Y	Rewrite character TRUE/FALSE to logical
UPDATE_LOGICAL=	Y	Rewrite character TRUE/FALSE to logical
SPACE_QUOTE_IDENTIFIER=	Y	If the SPACE_QUOTE_IDENTIFIER keyword is unset, it defaults to true, i.e., a space is returned as the quote character. To make the driver return an empty string for the quote character, this keyword must be set to N. Most applications work properly with the default behavior (SPACE_QUOTE_IDENTIFIER = Y). Known exceptions include some versions of Microsoft Visio, the Microsoft SQL Server DTS Wizard and Business Objects.
MIN_FIELD_LEN=		Minimum length for char field or expression.
MAX_FIELD_LEN=		Minimum length for char field or expression.

To connect to multiple databases through a single OpenLink client connection and/or to make use array type columns you must run the OpenLink provided "setup.p" utility. Please refer to the setup.p document for detailed information on the use of this script.

10.2.8.1 Configuring Progress Session Resources

You can control default behavior and progress server session resource allocation by entering standard progress session parameters in the "Server Options" field within the Admin Assistant's database agent configuration wizard dialogs or forms.

The following values are set for you by default at installation time and displayed as depicted below within the "Server Options" fields of the Admin Assistant Forms and Wizard dialogs.

```
        -T /tmp -d mdy -TB 31 -TM 31
```

10.2.8.2 Database Agent Specific Issues

Progress database servers support sockets and shared memory based methods of Inter Process Communication (IPC), unfortunately the shared memory approach which is much faster than sockets and the preferred approach by many users bears a cost of version incompatibility. This implies that your OpenLink database agents need to be an exact version match with your backend Progress database server in order to successfully initiate shared memory based database sessions (note: these agents are built using the Progress Embedded SQL package).

10.2.8.3 Rebuilding Progress Database agents

To get around the issue explained above OpenLink provides a relinkable library and script file that enables you to build an OpenLink database agent that has an exact match to the version of Progress that you have installed. See the section below on Relinking Progress Agents for details.

If shared memory based IPC isn't an issue for you then start your Progress server with the -S, -N, and -H options indicating the use of a sockets based Progress database server. This mode of operation is Progress version independent.

10.2.8.4 Application Server & 3-Tier Architecture Configuration

There may be situations in which you are unable to install your OpenLink Request Broker and Database Agents on the same machine as the one hosting your Progress database server. Irrespective of the reasons that lead you to this scenario, it is possible to configure your OpenLink database agents hosted on your Application Server machine such that they connect to a remote Progress database on your Database Server machine using Progress database specific networking (Progress Client Networking) as opposed to OpenLink's Database Independent Networking. The end result being a 3-tier distributed OpenLink architecture in which the communication between OpenLink clients and database agents use OpenLink Database independent Networking, while the communication between the Progress database agent and the Progress database server uses Progress Client Networking.

Configuration Steps:

Assuming you have a Progress Database Server machine called "mainserver2" that has a sockets based Progress Server process running, you would enter the following (assuming a TCP/IP based network):

1. Ensure that you have a usable connection to Progress using its native networking (Progress Client Networking) using the following remote database connection parameters:

   ```
   -S mainserver2 -H mainserver -N tcp .
   ```
2. Add the following values to the "Connect Options" field within the Admin Assistant Forms or Wizards used to configure your database agent in the Rulebook. If you choose to set this value on the client simply enter the same value into to the "Options" attribute associated with the configuration of your OpenLink client (see OpenLink ODBC or JDBC or UDBC client configuration for more details):

   ```
   -S mainserver2 -H mainserver -N tcp
   ```

10.2.9 See Also:

Progress Connection Trouble Shooting

10.2.10 See Also:

Application Server Architecture for various illustrations of distributed client-server architectures supported by OpenLink database agent

10.2.10.1 Unicode Configuration

See the Unicode section for configuration details.

10.2.11 Oracle

When configuring an Oracle database agent the critical configuration items are:

- Database Identification - this is an actual Oracle System Identifier (SID) e.g "ORCL", which identifies the actual Oracle environment that you want to be connected with. This is the value that you enter into the "Database Name" field of either your Admin Assistant's database agent configuration form or wizard dialog. If you choose to have database identification take at the client rather than server, you enter this value into the "Database Name" field or connection attribute when configuring your OpenLink client.

Oracle provides a number of environment variables for configuring database clients, the basic set required for successfully connecting your database agent to an Oracle database server are tabulated below:

Table 8.6. Default Rule Book settings

Rule Book Section & parameters	Default Rule Book Settings	Notes
[Environment ORACLE7]		
ORACLE_HOME=	/dbs/oracle7	The home directory for the Oracle installation.
ODBC_CATALOGS=	Y	Uncomment after loading the "odbccat7.sql" script.
MULTIPLEX_LDA=	5	Allow 5 OpenLink clients via a single database session
OPL_USER_TBLS_FIRST=	N	set to Y to show OpenLink User Tables first
SHOW_REMARKS=	N	Set to Y to retrieve SQLColumns REMARKS field
[Environment ORACLE8]		Parameters are common in Oracle 8 and 9i
[Environment ORACLE9]		
ORACLE_HOME=	/dbs/oracle8 or /dbs/oracle9	The home directory for the Oracle installation.
ODBC_CATALOGS=	Y	Uncomment after loading the "odbccat8.sql" script.
MULTIPLEX_LDA=	5	Allow 5 OpenLink clients via a single database session
OPL_USER_TBLS_FIRST=	N	set to Y to show OpenLink User Tables first
SHOW_REMARKS=	N	Set to Y to retrieve SQLColumns REMARKS field
OCI_PREFETCH_ROWS=	120	Sets the number of rows to be prefetched
OCI_PREFETCH_MEMORY=	64	Kb of memory allocated for rows to be prefetched

10.2.11.1 Database Agent Specific Settings

The "odbccat.sql" scripts explained:

These scripts exist for each version of Oracle supported, the files "odbccat6.sql", "odbccat7.sql", and "odbccat8.sql" representing Oracle versions 6 up to version 8 respectively. Oracle 9i can use the "odbccat8.sql" script. These scripts are to be applied to your Oracle instance to enable efficient and extended functionality between OpenLink and Oracle when handling ODBC, JDBC, UDBC, and OLE-DB catalog calls such as SQLForeignKeys() and SQLPrimaryKeys() functions. These functions have significant impact on the performance of your OpenLink clients.

To run these scripts you need to start the Oracle server manager (svrmgr or sqldba if you do this from the command line). Connect as internal and run the script by locating the relevant script file as you would any other Oracle SQL script file.

10.2.11.2 Rebuilding Oracle Database Agents

OpenLink provides a relinkable library and script files that enable you to rebuild your database agents as shared, as opposed to statically linked binaries, or for the purposes of getting a closer database implementation fit if your Oracle database environment is a more recent release than the actual version used by OpenLink to build the database agent installed on your system. Please read the Relinking OpenLink Database Agents section that follows, for details on how to perform this task.

10.2.11.3 Application Server & 3-Tier Architecture Configuration

There may be situations in which you are unable to install your OpenLink Request Broker and Database Agents on the same machine as the one hosting your Oracle database server. Irrespective of the reasons that lead you to this scenario, it is possible to configure your OpenLink database agents hosted on your Application Server machine such that they connect to a remote Oracle database on your Database Server machine using Oracle database specific networking (SQL*Net or Net8) as opposed to OpenLink's Database Independent Networking. The end result being a 3-tier distributed OpenLink architecture in which the communication between OpenLink clients and database agents use

OpenLink Database independent Networking, while the communication between the Oracle database agent and the Oracle database server uses Oracle SQL*Net or Net8.

Configuration Steps:

Assuming you have an Oracle Database Server machine called "mainserver2" that has an Oracle Listener process running, you would enter the following (presuming that your SQL*Net or Net8 alias for this Listener is also named "mainserver2"):

1. On your Application Server (the machine hosting your database agents) create a SQL*Net or Net8 Alias named "mainserver2" (for purposes of this example only).
2. Ensure that you have a usable connection to the remote Oracle database server using the SQL*Net or Net8 alias "mainserver2"
3. Add the following values to the "Server Options" field within the Admin Assistant Forms or Wizards used to configure your database agent. If you choose to set this value on the client simply enter the same value into to the "Database Name" Attribute associated with the configuration of your OpenLink client (see OpenLink ODBC or JDBC or UDBC client configuration for more details):

```
mainserver2
```

10.2.12 See Also:

Application Server Architecture for various illustrations of distributed client-server architectures supported by OpenLink database agent

10.2.12.1 Unicode Configuration

See the Unicode section for configuration details.

10.2.13 Sybase

10.2.14 Note:

The Sybase Agent is no longer required, all Sybase connectivity is handled by the TDS Agent.

See: TDS Agent

This section remains purely for reference on previous OpenLink versions

When configuring a Sybase database agent the critical configuration items are:

- Database Identification - this is the name of the Sybase database to be connected to ie pubs2 in the case of the Sybase provided sample database. This is the value that you enter into the "Database Name" field of either your Admin Assistant's database agent configuration form or wizard dialog. If you choose to have database identification take at the client rather than server, you enter this value into the "Database Name" field or connection attribute when configuring your OpenLink client.

Sybase provides a number of environment variables for configuring database clients, the basic set required for successfully connecting your database agent to a Sybase database server are tabulated below:

Table 8.7. Default Rule Book settings

Rule Book Section & parameters	Default Rule Book Settings	Notes
[Environment SYBASE4]		
SYBASE=	/dbs/sybase4	Full path to the base directory for the Sybase installation.
DSQUERY=	SYBASE	Name of the Sybase Query Server that you are connecting to.
[Environment SYBASE10]		
SYBASE=	/dbs/sybase10	Full path to the base directory for the Sybase installation.

Rule Book Section & parameters	Default Rule Book Settings	Notes
DSQUERY=	SYBASE	Name of the Sybase Query Server that you are connecting to.
[Environment SYBASE11]		
SYBASE=	/dbs/sybase11	Full path to the base directory for the Sybase installation.
DSQUERY=	SYBASE	Name of the Sybase Query Server that you are connecting to.

10.2.15 Note:

On some systems, like RedHat Linux, if the environment variable LANG=en_US is set it may need to be unset, or the environment variable LC_ALL=default set in the RuleBook to avoid RPC:Time Out errors on connect.

10.2.15.1 Rebuilding Sybase Database Agents

OpenLink provides a relinkable library and script files that enable you to rebuild your database agents as shared, as opposed to statically linked binaries, or for the purposes of getting a closer database implementation fit, should your Sybase database environment be a more recent release than the actual version used by OpenLink to build the database agent installed on your system. Please read the Relinking OpenLink Database Agents section that follows, for details on how to perform this task.

10.2.15.2 Application Server & 3-Tier Architecture Configuration

There may be situations in which you are unable to install your OpenLink Request Broker and Database Agents on the same machine as the one hosting your Sybase database server. Irrespective of the reasons that lead you to this scenario, it is possible to configure your OpenLink database agents hosted on your Application Server machine such that they connect to a remote Sybase database on your Database Server machine using Sybase database specific networking (Open Client) as opposed to OpenLink's Database Independent Networking. The end result being a 3-tier distributed OpenLink architecture in which the communication between OpenLink clients and database agents use OpenLink Database independent Networking, while the communication between the Sybase database agent and the Sybase database server uses Sybase Open Client.

Configuration Steps:

Assuming you have an Sybase Database Server machine called "mainserver2" that has an Sybase Server named "mainserver2" up and running:

1. On your Application Server (the machine hosting your database agents) create an Open Client Database Connection Alias named "mainserver2" (for purposes of this example only).
2. Ensure that you have a usable connection to the remote Sybase database server using the Open Client Database alias "mainserver2"
3. Add the following values to the "Server Options" field within the Admin Assistant Forms or Wizards used to configure your database agent. If you choose to set this value on the client simply enter the same value into to the "Database Name" Attribute associated with the configuration of your OpenLink client (see OpenLink ODBC or JDBC or UDBC client configuration for more details):

   ```
   mainserver2
   ```

   You may also enter the following values into the "Database Server Options" field: -s mainserver2

10.2.16 See Also:

Application Server Architecture for various illustrations of distributed client-server architectures supported by OpenLink database agent

10.2.16.1 Unicode Configuration

See the Unicode section for configuration details.

10.2.17 Microsoft SQL Server

You can specify the following environment variables for this agent:

- *FREETDSCONF* : a path to the freetds.conf file to use.
- *DSQUERY* : the section to use within freetds.conf
- *TDSSERVER* : the IP address or alias name for the server running SQL Server.
- *TDSPORT* : the TCP port number the SQLServer instance is running on
- *TDSVER* : the TDS Protocol version for the type of SQLServer instance being used.
- *TDSDBASE* : the name of the SQLServer Database

The following Connection Options can be passed to the driver:

- -*H* - Hostname of the machine SQL Server is running
- -*P* - SQL Server TCP Port
- -*V* - SQL Server version . Values supported are; 9.0 equal SQL Server 2005; 8.0 equal SQL Server 2000; 7.0 equal SQL Server 7
- -*S* - SQL Server name on the specified host. A SQL Server instance can also be specified by appending "\InstanceName" to the ServerName?? ie "ServerName\InstanceName"
- -*F* - Specify Fail over server name for Database Mirroring
- -*N* - Network Packet size, which is a value that determines the number of bytes per network packet transferred from the database server to the client. The correct setting of this attribute can improve performance. When set to 0, the initial default, the driver uses the default packet size as specified in the Sybase server configuration. When set to -1, the driver computes the maximum allowable packet size on the first connect to the data source and saves the value in the system information.
- -*O Prepared Method, which is a value that determines whether stored procedures are created on the server for every call to SQLPrepare. When set to Full (2), stored procedures are created for every call to SQLPrepare, which can decrease performance when processing static statements. When set to Partial(1), the initial default, the driver creates stored procedures only if the statement contains parameters. Otherwise, the statement is cached and executed directly at SQLExecute time. When set to None (0), the driver never creates stored procedures.*
- -*L* - The name of a Microsoft SQL Server supported national language. The default language is English.
- -*C* - The name of an installed character set on the server. By default it is the setting on the Microsoft SQL Server.
- -*A* - Turn Off ANSI NULL handling
- -*E* - Turn On SSL Strong Data Encryption
- -*R* - Turn On SSL Server Certificate Verification
- -*D* - Specify Name of Certificate Authority file to be used for SSL Certificate Verification
- -*Z* - Turn On Snapshot Serializable Isolation
- -*M* - Turn On Multiple Active Result Sets (MARS)

10.2.18 Note

Note on Precedence:

The web-based Administrative Assistant modifies/sets environment variables, above. These environment variables are used in preference to any -H, -V or -P connect-options that might be specified.

10.2.18.1 Application Server & 3-Tier Architecture Configuration

There may be situations in which you are unable to install your OpenLink Request Broker and Database Agents on the same machine as the one hosting your Microsoft SQL Server database server. Irrespective of the reasons that lead you to this scenario, it is possible to configure your OpenLink database agents hosted on your Application Server machine such that they connect to a remote Microsoft SQL Server database on your Database Server machine using Microsoft SQL Server database specific networking (NETLIB) as opposed to OpenLink's Database Independent Networking. The end result being a 3-tier distributed OpenLink architecture in which the communication between OpenLink clients and database agents use OpenLink Database independent Networking, while the communication between the Microsoft SQL Server database agent and the Microsoft SQL Server database server uses Microsoft SQL Server's NETLIB.

Configuration Steps:

Assuming you have an Microsoft SQL Server Database Server machine called "oplwinnt" that has a Microsoft SQL Server Server named "oplwinnt" up and running:

1. On your Application Server (the machine hosting your database agents) create a NETLIB Database Connection Alias named "oplwinnt" (for purposes of this example only).
2. Ensure that you have a usable connection to the remote Microsoft SQL Server database server using the Open Client Database alias "oplwinnt" (this the value you provide whenever you are prompted for a Server Name by native SQL Server utilities)
3. Add the following values to the "Server Options"field within the Admin Assistant Forms or Wizards used to configure your database agent. If you choose to set this value on the client simply enter the same value into to the "Database Name" Attribute associated with the configuration of your OpenLink client ODBC or JDBC or UDBC client configuration for more details):

   ```
   oplwinnt
   ```

   You may also enter the following values into the "Database Server Options" field: -s oplwinnt

10.2.18.2 Unicode Configuration

See the Unicode section for configuration details.

10.2.19 TDS

The TDS Agent supercedes the Sybase and SQLServer database agents. Although they share a common connection mechanism there are some subtle value differences that must be observed.

TDS provides a number of environment variables for configuring database clients, a typical set used for successfully connecting your database agent to a SQL Server are tabulated below:

Table 8.8. Default Rule Book settings

Rule Book Section & parameters	Default Rule Book Settings	Notes
[Environment SQLServer 2000]		The section name will vary with the agent version and database version.
FREETDSCONF=	/home/openlink/bin/freetds.conf	
DSQUERY=	SQLSERVER	Load this section from the freetds.conf file.
TDSHOST=	sqlsrv	Point this to your SQL Server /Sybase machine
TDSPORT=	1433	The port on which you SQL Server/Sybase listens for incoming connections. SQL Server uses port 1433 by default, however Sybase uses 4100.
TDSVER=	7.0	The required TDS connectivity version. Refer to the table below.
SQLSERVER_CATALOG=	Y	
CURSOR_SENSITIVITY=	LOW	You can set this to HIGH after loading oplrvc.sql

The TDSVER option above must be set correctly for connection to the desired database. Use the following table to find which.

Table 8.9. TDSVER settings

Version Number	Database Supported
4.2	SQLServer 6 SQLServer 6.x
4.6	Sybase 4 Sybase 4.x

Version Number	Database Supported
5.0	Sybase 10 Sybase 11 Sybase 12 Sybase 15
7.0	SQLServer 7
8.0	SQLServer 2000
9.0	SQLServer 2005

10.2.20 DB2

Table 8.10. Default Rule Book settings

Rule Book Section & parameters	Default Rule Book Settings	Notes
[Environment DB2]		
DB2DIR=	/DB2	Full path to the base directory for the DB2 installation. Note,DB2 version 5 and below use the environment variable DB2PATH instead.
DB2INSTANCE=	DB2	Name of the instance you want to connect to. DB2 is the default DB2 instance name.

Agent Section

OpsysLogin=No; Validation of users is left to DB2

Database Agent default name: db2_sv

10.2.20.1 Unicode Configuration

See the Unicode section for configuration details.

10.2.21 PostgresSQL

Table 8.11. Default Rule Book settings

Rule Book Section & parameters	Default Rule Book Settings	Notes
[Environment POSTGRES]		
;ODBC_CATALOGS=	Y	Uncomment after loading odbccat defs

Agent Section

OpSysLogin= Yes; Users are validated against the operating system.

The following options can be used in the *ConnectOptions* field of the rule book (oplrqb.ini) to override client *Options* settings:

- -H <hostname>. Connect to postmaster on <hostname>. This defaults to localhost if unspecified
- -P <port>. Contact postmaster on port <port>, which defaults to PGPORT (5432) is unspecified.
- -T. Disable ODBC transaction management. All transactions will be automatically committed. This prevents palloc() failures with out-of-memory errors when doing really big transactions such as Exporting 10000 records from MS/Access.

Database Agent default name: pgr7_mv (or pgr7_sv which is single threaded)

This ODBC database agent has been built using PostgreSQL release 7.2. Using this agent with a previous version of Postgres will probably fail. If you experience any problems with older releases, please try against release 7.2 first before contacting OpenLink.

10.2.21.1 Data type Mapping from ODBC Data types to Postgres Data types

This information can be queried by the SQLTypeInfo catalog call.

Table 8.12. Default Rule Book settings

SQL type	Postgres type
SQL_CHAR	char
SQL_VARCHAR	varchar
SQL_LONGVARCHAR	text
SQL_DECIMAL	varchar
SQL_NUMERIC	varchar
SQL_BIT	bit
SQL_TINYINT	int2
SQL_SMALLINT	int2
SQL_INTEGER	int4
SQL_BIGINT	varchar
SQL_REAL	float4
SQL_DOUBLE	float8
SQL_FLOAT	float8
SQL_BINARY	not supported
SQL_VARBINARY	not supported
SQL_LONGVARBINARY	not supported
SQL_DATE	not supported
SQL_TIME	not supported
SQL_TIMESTAMP	not supported

10.2.21.2 Data type Mapping from Postgres Data types to ODBC Data types

This information can be queried by the SQLColumns catalog call.

Table 8.13. Default Rule Book settings

Postgres type	SQL type	Precision
bool	SQL_CHAR	1
char	SQL_CHAR	1
name	SQL_VARCHAR	32
char16	SQL_CHAR	16
int2	SQL_SMALLINT	
int4	SQL_INTEGER	
regproc	SQL_VARCHAR	16
text	SQL_LONGVARCHAR	8000
oid	SQL_INTEGER	(*)
tid	SQL_VARCHAR	19
xid	SQL_VARCHAR	12
cid	SQL_VARCHAR	3
oid8	SQL_VARCHAR	89
smgr	SQL_VARCHAR	12
char2	SQL_VARCHAR	2
char4	SQL_VARCHAR	4
char8	SQL_VARCHAR	8
filename	SQL_VARCHAR	255

Postgres type	SQL type	Precision
float4	SQL_REAL	
float8	SQL_DOUBLE	
abstime	SQL_VARCHAR	60
reltime	SQL_VARCHAR	60
tinterval	SQL_VARCHAR	60
unknown	SQL_VARCHAR	255
bpchar	SQL_CHAR	255
varchar	SQL_VARCHAR	255
date	SQL_CHAR	10
time	SQL_CHAR	16
other type	SQL_LONGVARCHAR	8000

(*) Note: An oid is returned as a SQL_INTEGER, because otherwise it would not be useable for SQLSpecialColumns.

Binary data / large objects are not supported in this release.

10.2.22 MySQL

Table 8.14. Default Rule Book settings

Rule Book Section & parameters	Default Rule Book Settings	Notes
[Environment MySQL]		
;ODBC_CATALOGS=	Y	Uncomment after loading odbccat defs

Agent Section

OpSysLogin= Yes; Users are validated against the operating system.

The following options can be used in the *ConnectOptions* field of the rule book (oplrqb.ini) to override client *Options* settings:

- -H <hostname>. Connect to server running on <hostname>. This defaults to localhost if unspecified
- -P <port>. Contact server on port <port>, which defaults to (3306) is unspecified.
- -T. Disable all ODBC transaction support. The agent will never issue any extra statements for transaction management (commit, rollback, modification of transaction isolation level)

This ODBC database agent has been built and verified using MySQL release 3.23.48.

10.2.22.1 Data type Mapping from ODBC Data types to MySQL Data types

This information can be queried by the SQLTypeInfo catalog call.

Table 8.15. Default Rule Book settings

SQL type	MySQL type
SQL_CHAR	char
SQL_VARCHAR	varchar
SQL_LONGVARCHAR	text
SQL_DECIMAL	decimal
SQL_NUMERIC	decimal
SQL_BIT	not supported
SQL_TINYINT	tinyint
SQL_SMALLINT	smallint
SQL_INTEGER	integer

SQL type	MySQL type
SQL_BIGINT	bigint
SQL_REAL	real
SQL_DOUBLE	double
SQL_FLOAT	double
SQL_BINARY	tinyblob
SQL_VARBINARY	tinyblob
SQL_LONGVARBINARY	longblob
SQL_DATE	date
SQL_TIME	time
SQL_TIMESTAMP	timestamp

10.2.22.2 Data type Mapping from MySQL Data types to ODBC Data types

This information can be queried by the SQLColumns catalog call.

Table 8.16. Default Rule Book settings

MySQL type	SQL type	Precision
bigint	SQL_BIGINT	19
blob	SQL_VARBINARY	2^16-1
char(n)	SQL_CHAR	n
date	SQL_CHAR	32
datetime	SQL_CHAR	32
decimal(p,s)	SQL_DECIMAL	p
double	SQL_DOUBLE	15
enum	SQL_VARCHAR	255
float	SQL_REAL	7
int	SQL_INTEGER	10
longblob	SQL_LONGVARBINARY	2^32-1
longtext	SQL_LONGVARCHAR	2^32-1
mediumblob	SQL_VARBINARY	2^24-1
mediumint	SQL_INTEGER	8
mediumtext	SQL_LONGVARCHAR	2^24-1
set	SQL_VARCHAR	255
smallint	SQL_SMALLINT	5
text	SQL_LONGVARCHAR	2^16-1
time	SQL_VARCHAR	32
timestamp(n)	SQL_VARCHAR	n
tinyblob	SQL_VARBINARY	255
tinyint	SQL_TINYINT	3
tinytext	SQL_VARCHAR	255
varchar(n)	SQL_VARCHAR	n
varchar(n) binary	SQL_VARCHAR	n
year	SQL_VARCHAR	4

10.2.23 OpenLink ODBC Agent Installation & Configuration

The OpenLink ODBC Agent is an ODBC Proxy Service that facilitates the integration of non OpenLink ODBC Drivers in the Sophisticated OpenLink Multi Tier ODBC Architecture, thereby extending the benefit of this architecture beyond the scope of OpenLink ODBC drivers.

10.2.23.1 Typical Utilization:

A typical and very popular use of the OpenLink ODBC Agent is the exposure of tradition desktop database engines within your organization to your new Intranet or Internet based infrastructures, using a client-server distributed computing model. This implies that you can have multiple concurrent clients within your Intranet or remote Internet clients connecting to your Microsoft Access, DBASE, Fox PRO, Paradox database engines without any compromises in security and with astonishing performance.

10.2.23.2 Installation:

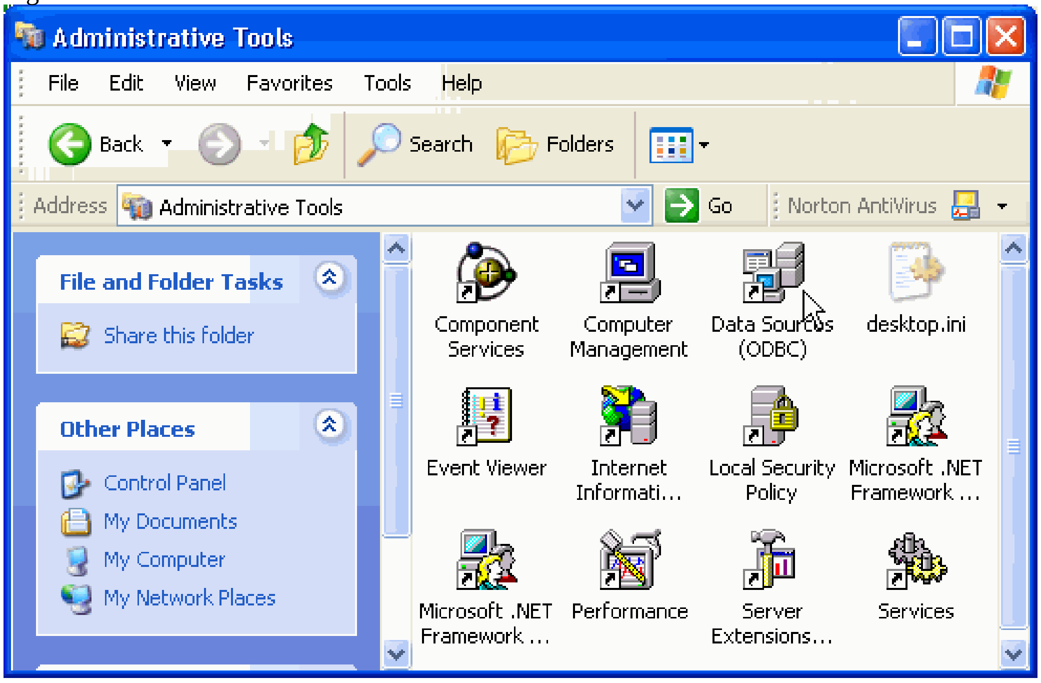
After downloading the OpenLink Data Access Driver Suite for your chosen desktop operating system please perform the following steps:-

1. Move into your temporary installation directory
2. Extract the contents of the OpenLink ZIP archive into the directory in step 1
3. Double click on the program "setup.exe"
4. Follow the on-screen instructions

10.2.23.3 Post Installation & Pre Configuration Check List

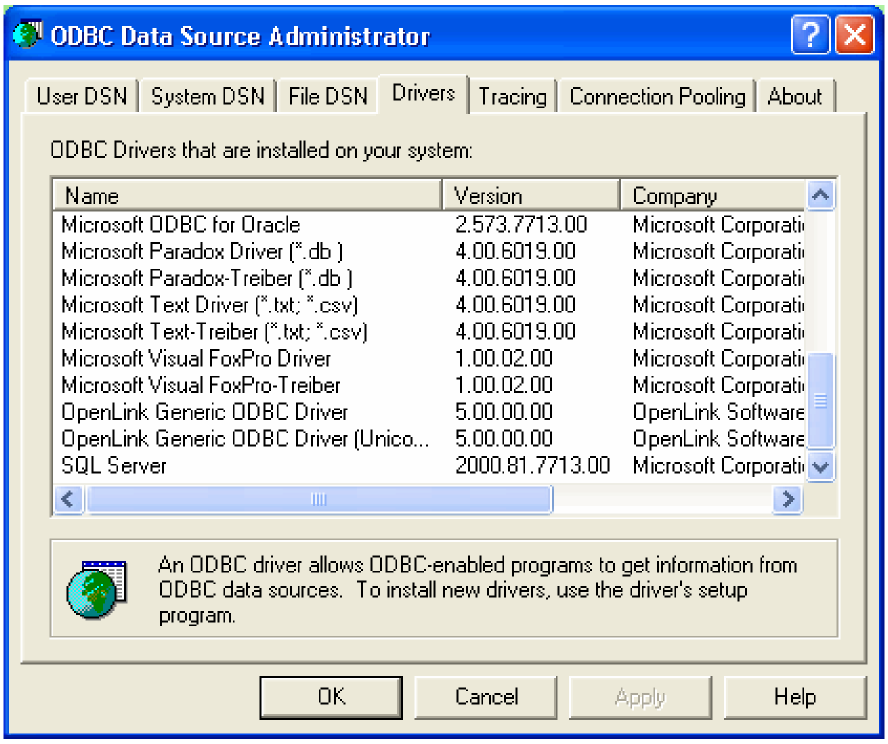
1. Verify that the ODBC Driver Manager exists on your system by opening up your desktop's control panel group. This will be under Administrative Tools on a Windows XP system.

   Figure 8.18. Admin Assistant



2. Verify the existence of an ODBC Driver for the desktop database engine that you will be connecting with (the example below shows OpenLink and Microsoft drivers installed:

   Figure 8.19. Admin Assistant



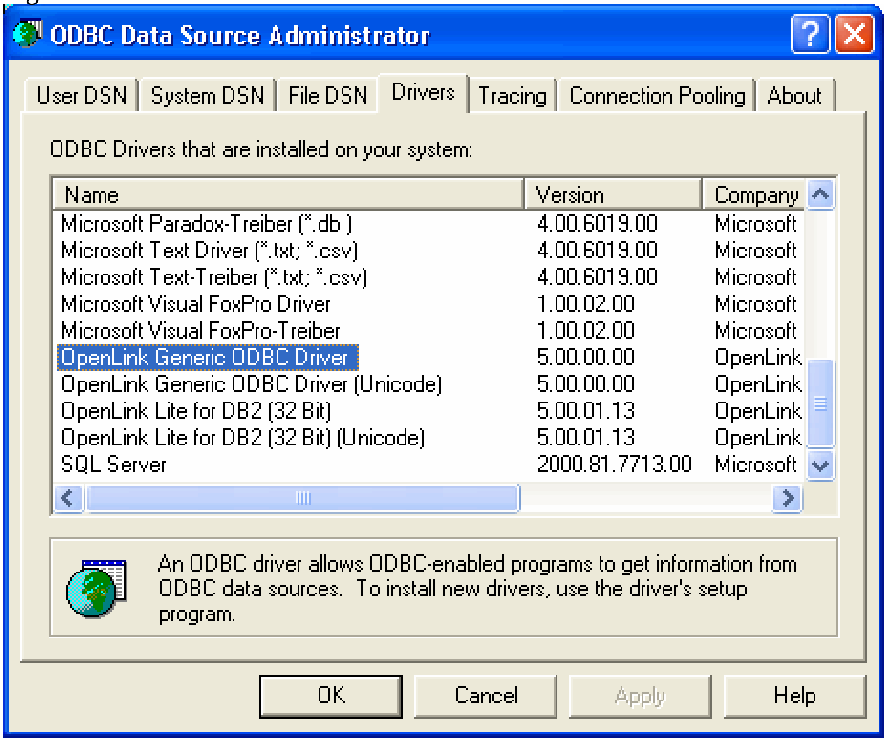
ODBC Drivers that are installed on your system:

Name	Version	Company
Microsoft ODBC for Oracle	2.573.7713.00	Microsoft Corporati
Microsoft Paradox Driver (*.db)	4.00.6019.00	Microsoft Corporati
Microsoft Paradox-Treiber (*.db)	4.00.6019.00	Microsoft Corporati
Microsoft Text Driver (*.txt; *.csv)	4.00.6019.00	Microsoft Corporati
Microsoft Text-Treiber (*.txt; *.csv)	4.00.6019.00	Microsoft Corporati
Microsoft Visual FoxPro Driver	1.00.02.00	Microsoft Corporati
Microsoft Visual FoxPro-Treiber	1.00.02.00	Microsoft Corporati
OpenLink Generic ODBC Driver	5.00.00.00	OpenLink Software
OpenLink Generic ODBC Driver (Unico...	5.00.00.00	OpenLink Software
SQL Server	2000.81.7713.00	Microsoft Corporati

An ODBC driver allows ODBC-enabled programs to get information from ODBC data sources. To install new drivers, use the driver's setup program.

3. Verify the existence of an ODBC System Data Source Name (DSN) for the database engine that you will be exposing via the OpenLink ODBC Agent, this ODBC DSN must be associated with the appropriate ODBC Driver for your desktop database
4. Verify the existence of an OpenLink ODBC Driver installation on your PC

Figure 8.20. Admin Assistant



ODBC Drivers that are installed on your system:

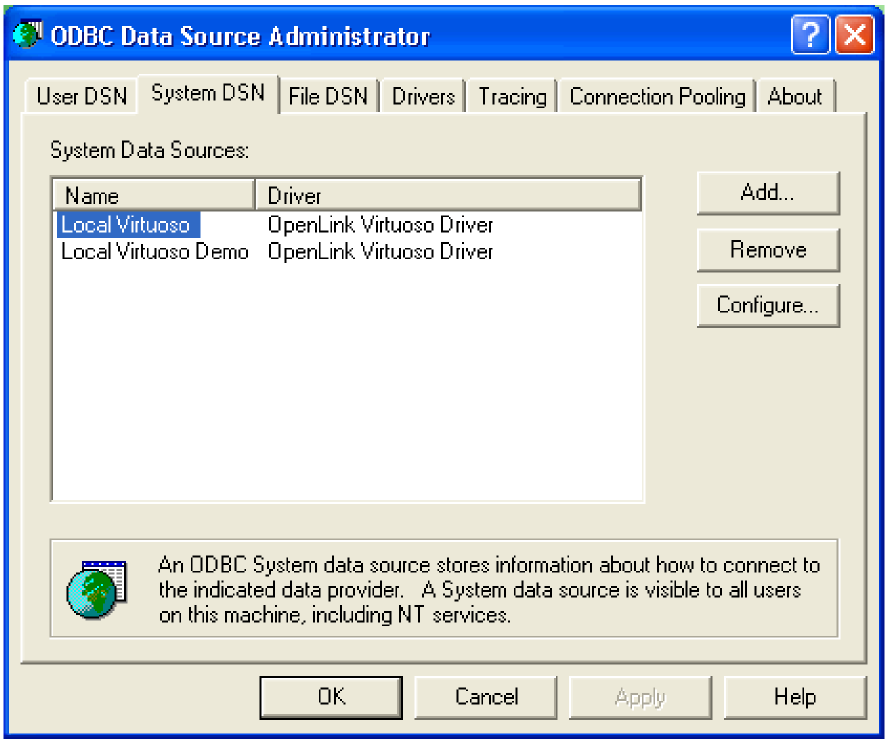
Name	Version	Company
Microsoft Paradox-Treiber (*.db)	4.00.6019.00	Microsoft
Microsoft Text Driver (*.txt; *.csv)	4.00.6019.00	Microsoft
Microsoft Text-Treiber (*.txt; *.csv)	4.00.6019.00	Microsoft
Microsoft Visual FoxPro Driver	1.00.02.00	Microsoft
Microsoft Visual FoxPro-Treiber	1.00.02.00	Microsoft
OpenLink Generic ODBC Driver	5.00.00.00	OpenLink
OpenLink Generic ODBC Driver (Unicode)	5.00.00.00	OpenLink
OpenLink Lite for DB2 (32 Bit)	5.00.01.13	OpenLink
OpenLink Lite for DB2 (32 Bit) (Unicode)	5.00.01.13	OpenLink
SQL Server	2000.81.7713.00	Microsoft

An ODBC driver allows ODBC-enabled programs to get information from ODBC data sources. To install new drivers, use the driver's setup program.

10.2.23.4 Configuring An OpenLink ODBC Agent Based ODBC Data Source Name (DSN)

1. Open up the ODBC Administrator within your desktop control panel, and then Click on the "Add" button to indicate that you want to add a new ODBC DSN to the current list of installed ODBC DSNs

Figure 8.21. Admin Assistant



ODBC Data Source Administrator

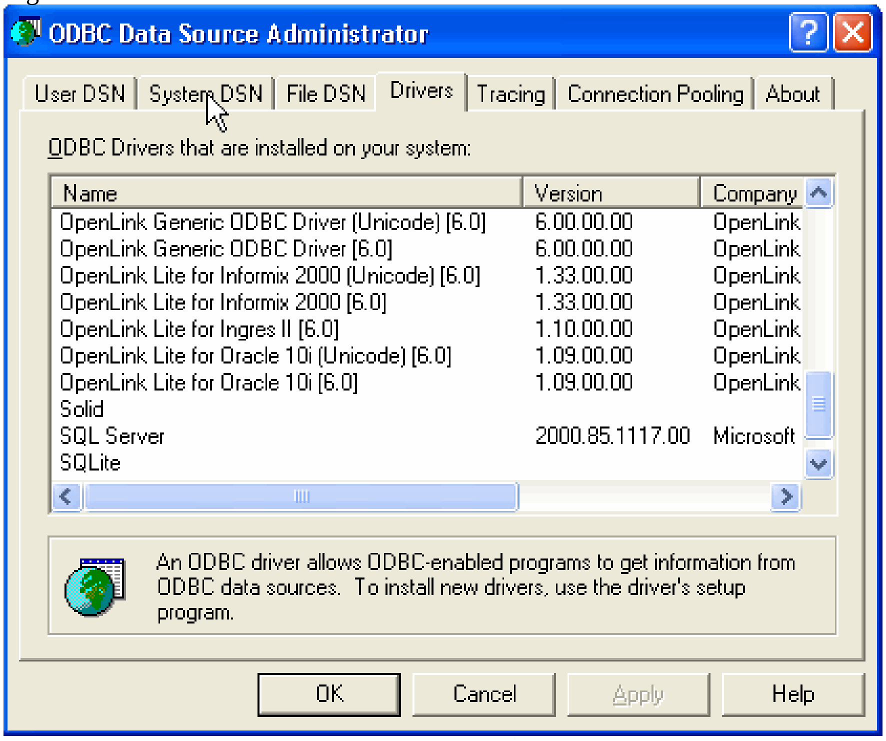
System Data Sources:

Name	Driver
Local Virtuoso	OpenLink Virtuoso Driver
Local Virtuoso Demo	OpenLink Virtuoso Driver

An ODBC System data source stores information about how to connect to the indicated data provider. A System data source is visible to all users on this machine, including NT services.

2. Click on the appropriate ODBC Driver that you will be associating this new ODBC DSN with, in this case the "OpenLink Generic ODBC Driver"

Figure 8.22. Admin Assistant



ODBC Data Source Administrator

ODBC Drivers that are installed on your system:

Name	Version	Company
OpenLink Generic ODBC Driver (Unicode) [6.0]	6.00.00.00	OpenLink
OpenLink Generic ODBC Driver [6.0]	6.00.00.00	OpenLink
OpenLink Lite for Informix 2000 (Unicode) [6.0]	1.33.00.00	OpenLink
OpenLink Lite for Informix 2000 [6.0]	1.33.00.00	OpenLink
OpenLink Lite for Ingres II [6.0]	1.10.00.00	OpenLink
OpenLink Lite for Oracle 10i (Unicode) [6.0]	1.09.00.00	OpenLink
OpenLink Lite for Oracle 10i [6.0]	1.09.00.00	OpenLink
Solid		
SQL Server	2000.85.1117.00	Microsoft
SQLite		

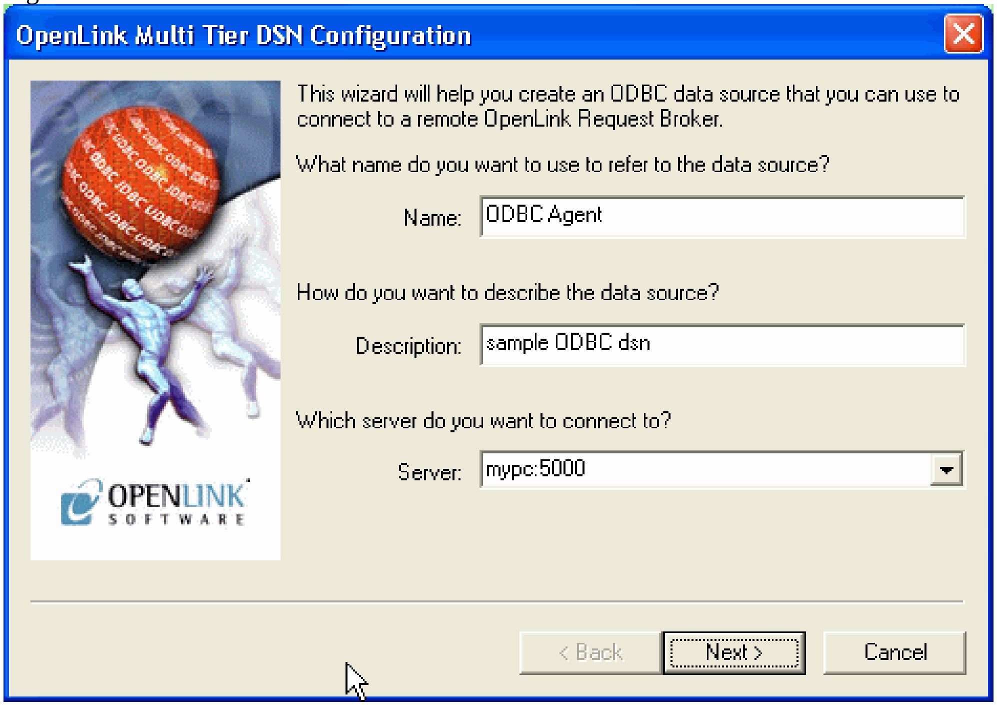
An ODBC driver allows ODBC-enabled programs to get information from ODBC data sources. To install new drivers, use the driver's setup program.

3. Choose a Name for your OpenLink ODBC DSN and then type it into the "Name" field, the example below presumes the DSN is to be called "ODBC Agent"

   Type a comment to describe the DSN.

   Enter the name and port of the machine hosting the OpenLink ODBC agent in the "Server" field. The example below presumes that the machine network alias for your desktop computer is "mypc" (note: you can also use the machines actual IP address or even use the "localhost" account if you are connecting to a local as opposed to remote non OpenLink ODBC DSN). The port is the default of 5000.

Figure 8.23. Admin Assistant



This wizard will help you create an ODBC data source that you can use to connect to a remote OpenLink Request Broker.

What name do you want to use to refer to the data source?

Name: ODBC Agent

How do you want to describe the data source?

Description: sample ODBC dsn

Which server do you want to connect to?

Server: mypc:5000

4. Choose an OpenLink "Server Type" of "ODBC" from the "Domain" listbox

   Type the name of the non OpenLink ODBC DSN that you would like to associate this OpenLink ODBC DSN with into the "Database" field. The example below presumes the existence of a non OpenLink ODBC DSN named "LocalAccess" that is bound to the Microsoft ODBC Driver for MS Access.

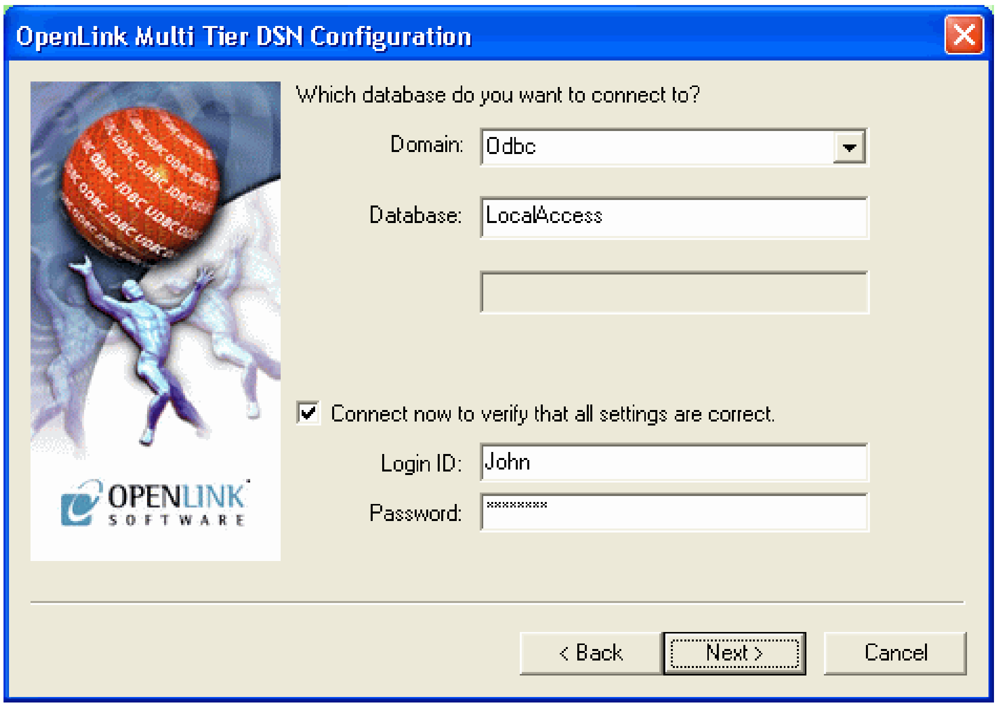
   When the "Connect now.." tick box is checked, a test connection is made to verify the Data Source connection.

   If there is no check then the Login ID and Password fields are ignored, and no test is performed.

   The Login ID is the default database UserID to use when logging on to a remote database engine (identified by the Domain above).

   Password is for the login of the above UserID.

   Figure 8.24. Admin Assistant



Which database do you want to connect to?

Domain: Odbc

Database: LocalAccess

☑ Connect now to verify that all settings are correct.

Login ID: John

Password: ********

5. Now define additonal connection parameters:

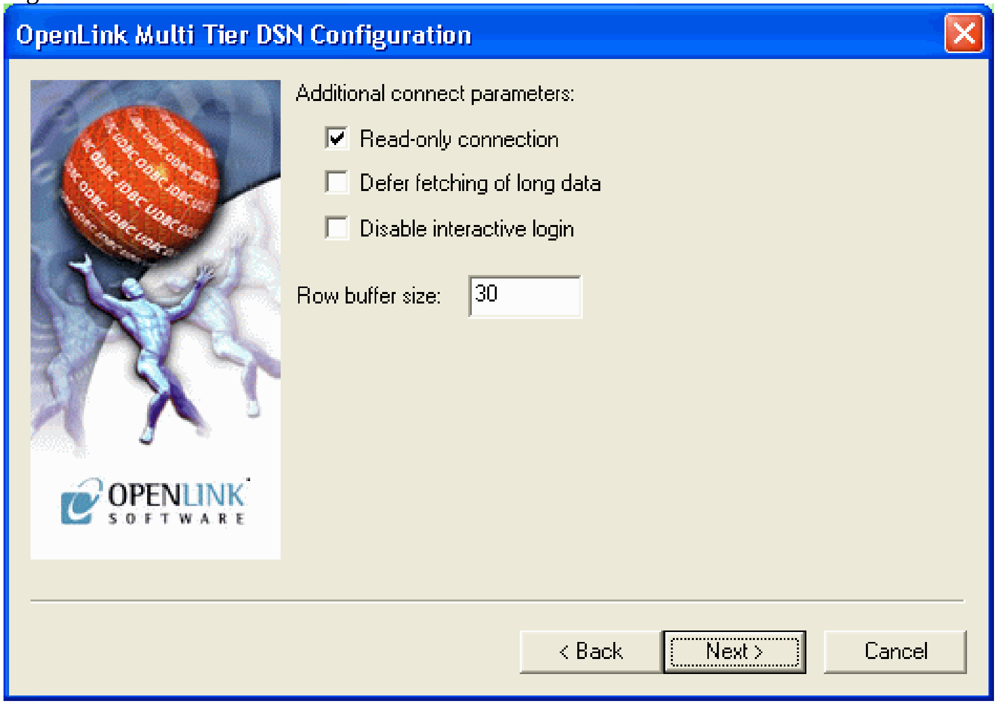
   Read-only connection. Specify whether the connection is to be "Read-only". Make sure the checkbox is unchecked to request a "Read/Write" connection.

   Defer fetching of long data. Check this box to defer the fetching of long data.

   Disable interactive login. Suppress the ODBC "Username" and "Password" login dialog box when interacting with your ODBC DSN from within an ODBC compliant application.

   Row Buffer Size. This attribute specifies the number of records to be transported over the network in a single network hop. Values can range from 1 to 99.

   Figure 8.25. Admin Assistant



Additional connect parameters:

☑ Read-only connection

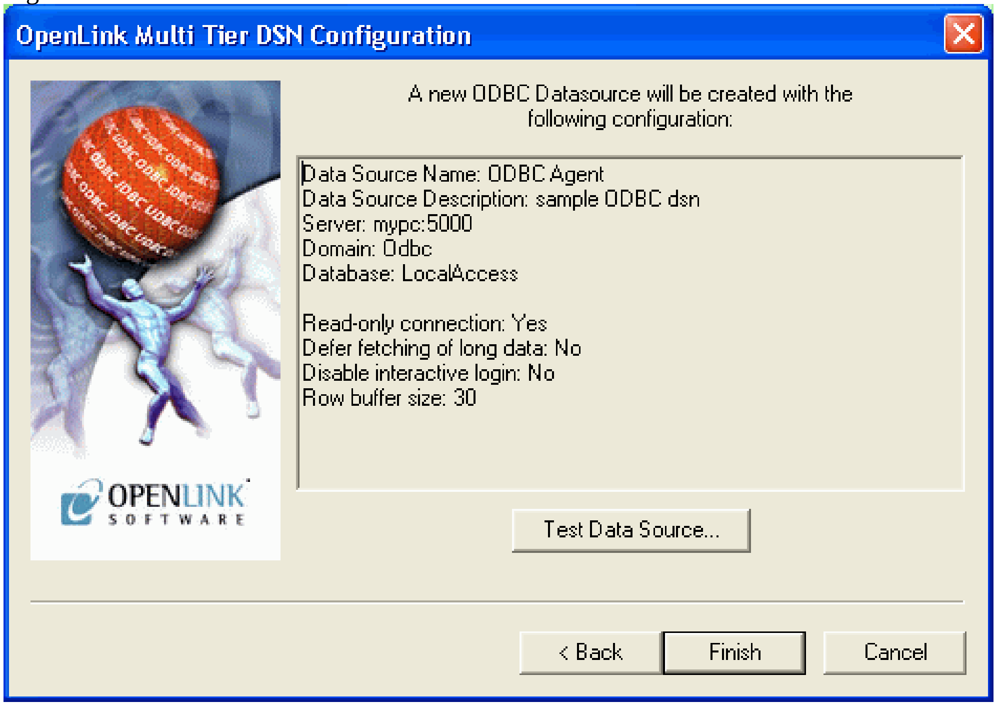
☐ Defer fetching of long data

☐ Disable interactive login

Row buffer size: 30

6. The details of the DSN are now shown. To test the DSN, press the Test Data Source button.

Figure 8.26. Admin Assistant

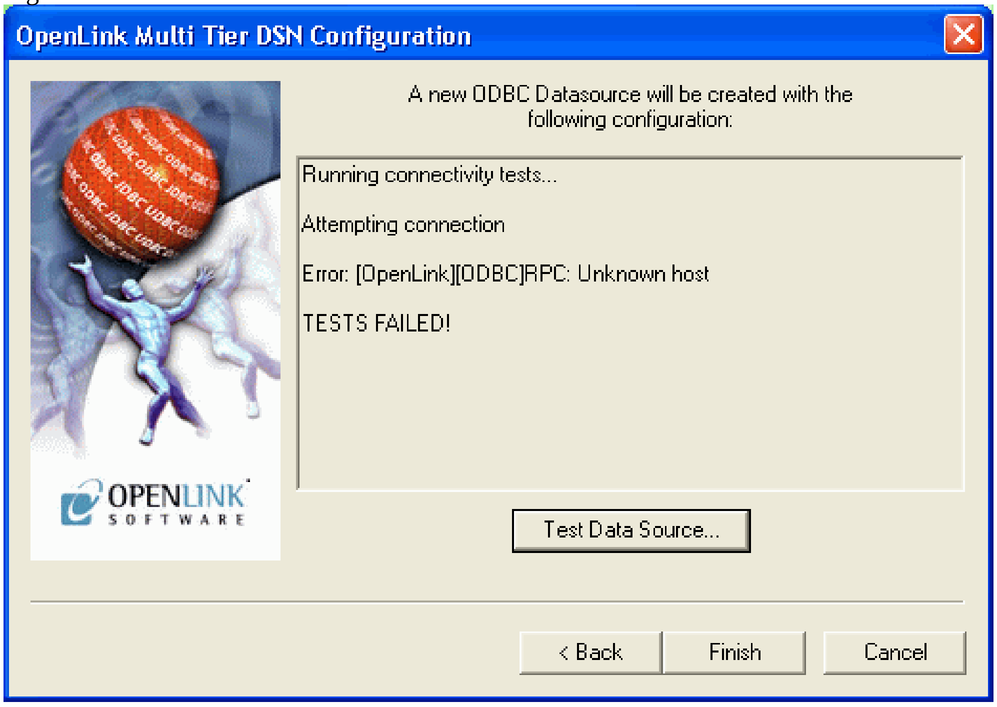


A new ODBC Datasource will be created with the following configuration:

```
Data Source Name: ODBC Agent
Data Source Description: sample ODBC dsn
Server: mypc:5000
Domain: Odbc
Database: LocalAccess

Read-only connection: Yes
Defer fetching of long data: No
Disable interactive login: No
Row buffer size: 30
```

7. If a Test is requested then the status of the DSN test connection is shown:

Figure 8.27. Admin Assistant



A new ODBC Datasource will be created with the following configuration:

```
Running connectivity tests...

Attempting connection

Error: [OpenLink][ODBC]RPC: Unknown host

TESTS FAILED!
```

8. Click on the "Finish" button to complete the creation of your new OpenLink ODBC DSN.

10.2.23.5 Making A Test Connection To Your OpenLink ODBC Agent Based ODBC DSN

To verify that your installation and configuration is ready for use, please follow the steps below in order to make a test connection to the OpenLink ODBC DSN that you have just created:-

1. Start the OpenLink Request Broker in debug mode, this can be done from a DOS shell by executing the command : oplrqb -dv or from your Services Panel (note you must change the startup mode to manual to enable the OpenLink Request Broker run in Debug Mode)
2. Locate the program "VBDemo" or "C++ Demo" situated within the "OpenLink Data Access Drivers" group on

your desktop (Windows start menu item)
3. Attempt to make a connection to the OpenLink ODBC DSN
4. If step 3 is successful and you see data exchanged between your ODBC Client and your OpenLink ODBC DSN, exit the ODBC application, and then shutdown and restart the Broker without the Debug Mode options using the command: oplrqb -v. If step 3 is unsuccessful repeat step 3 and then capture the Request Broker output and proceed to instigating contact with OpenLink Technical support via the OpenLink Support Page
5. Establish connection between your own ODBC Applications and the OpenLink ODBC DSN created in step 3
6. Shutdown the Request Broker using the command: oplshut -f

10.2.23.6 Adding An OpenLink ODBC Agent To An Early OpenLink System

1. Place the ODBC Agent executable in the openlink/bin sub-directory
2. Edit the OpenLink "Session Rules" Book (the file oplrqb.ini) using a text editor
3. Insert a new OpenLink Session Mapping rule to the top of the "[Mapping Rules]" section of the Rule Book in the manner depicted below:

```
odbc:*:*:*:*:*:* = accept odbc_agent
```
4. Then create a new OpenLink Agent section as follows:

```
[odbc_agent]
Program = odbc_sv
```

10.2.24 Note:

Program key must be set to the exact file name. (For NT this would be odbc_sv.exe)
5. Save the file
6. Shutdown and restart your OpenLink Request Broker.

10.2.25 OpenLink ODBC-JDBC Agent Bridge Installation & Configuration

The OpenLink ODBC-JDBC Agent bridge is a JDBC Proxy Service enabling ODBC connectivity to a JDBC data repository via an existing JDBC driver for the data repository. Thus essentially it is the reverse of the more traditional JDBC-ODBC driver bridge used for accessing databases via JDBC.

10.2.25.1 Typical Utilization:

A typical and very popular use of the OpenLink ODBC-JDBC Agent bridge would be to access a Database written purely in Java for which a JDBC driver is the only data access mechanism available. As databases written purely in Java become more popular the use of the ODBC-JDBC bridge will become more relevant.

10.2.25.2 Installation:

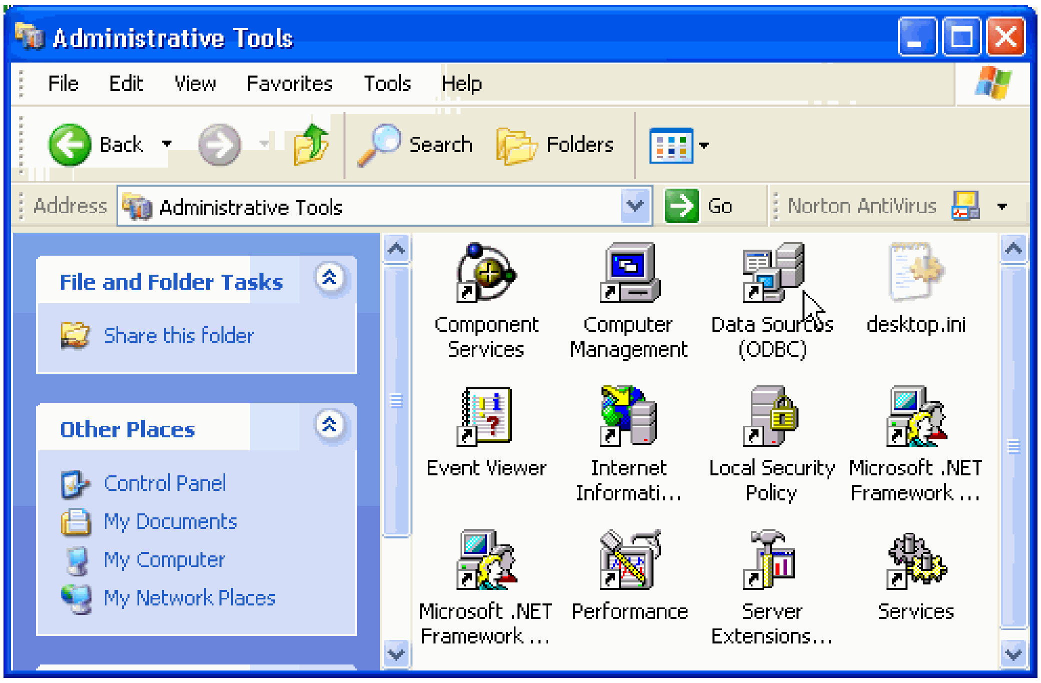
After downloading the OpenLink Data Access Driver Suite for your chosen desktop operating system please perform the following steps:-

1. Move into your temporary installation directory
2. Extract the contents of the OpenLink ZIP archive into the directory in step 1
3. Double click on the program "setup.exe"
4. Follow the on-screen instructions

10.2.25.3 Post Installation & Pre Configuration Check List

1. Verify that the ODBC Driver Manager exists on your system by opening up your desktop's control panel group

Figure 8.28. Admin Assistant

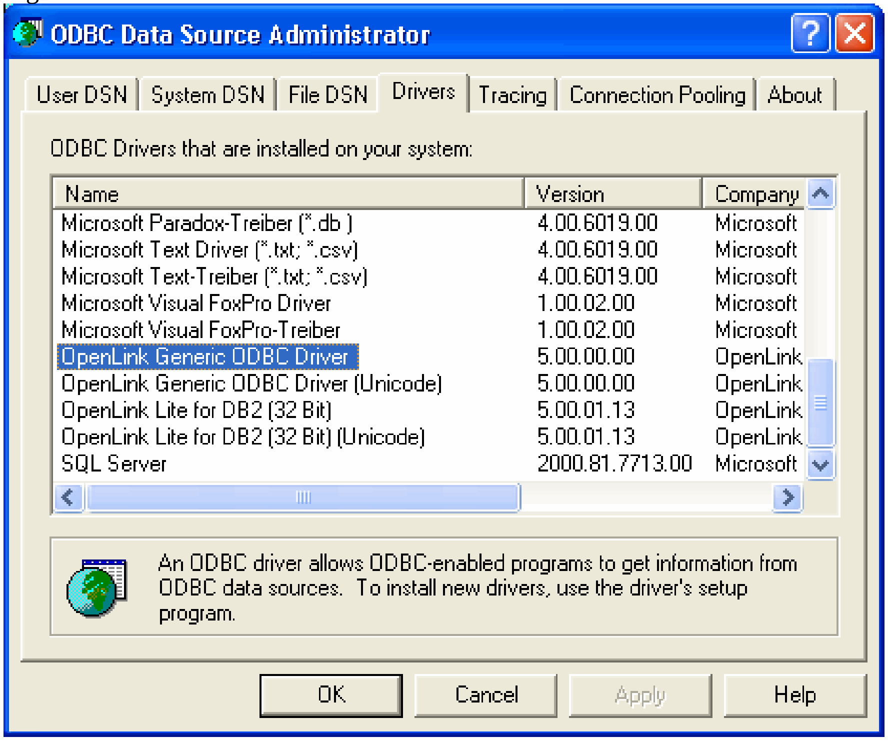


2. Verify the Java environment for the JDBC driver to be used is configured. In particular the CLASSPATH and PATH environment variables must be set correctly for a successful connection. If this is not set in the user's environment by default, you can configure this via the OpenLink session Rules Book (oplrqb.ini) via the relevant [Environment JDBCXX] section where XX = 12 or 13, indicating the version of the JDK in use. Example here is wrapped across lines for presentation.

```
[Environment JDBC12]
CLASSPATH = .;D:\Software\oracle\ora81\jdbc\lib\classes12.zip;D:\Softwa
re\oracle\ora81\jdbc\lib\nls_charset12.zip;c:\program files\openlink\vi
rtuoso 2.0\jdk1.2\virtjdbc2.jar;c:\program files\openlink\jdk1.2\opljdb
c2.jar
PATH = D:\Software\oracle\ora81\bin;D:\Software\oracle\ora81\jdbc\lib;D
:\Software\jdk1.2\jre\bin;D:\Software\jdk1.2\jre\bin\classic;C:\WINNT\S
YSTEM32;C:\WINNT
```

3. Verify the existence of an OpenLink ODBC Driver installation on your PC

Figure 8.29. Admin Assistant



ODBC Drivers that are installed on your system:

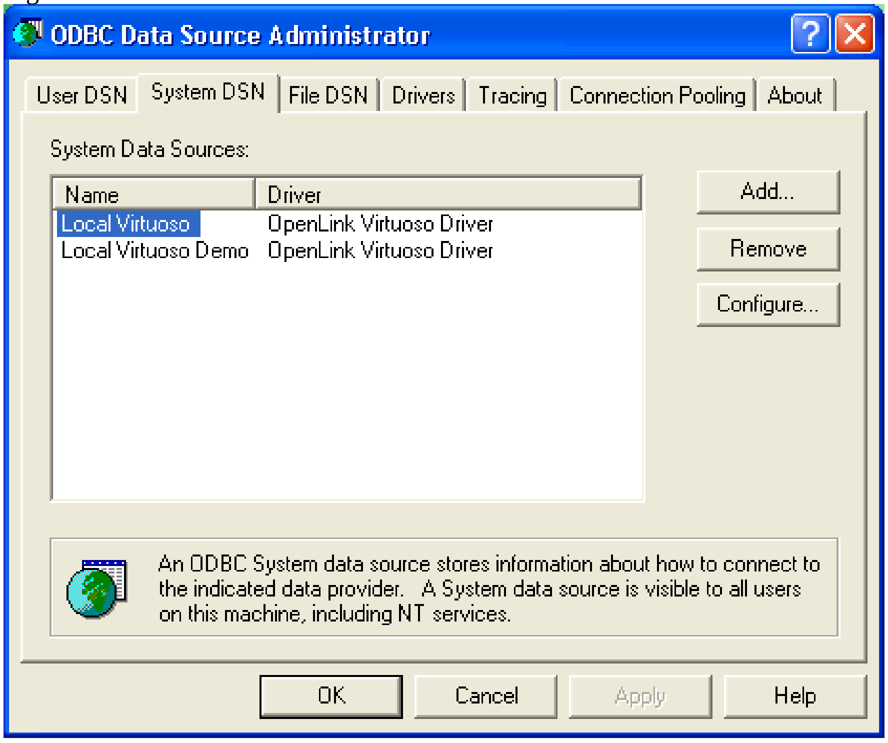
Name	Version	Company
Microsoft Paradox-Treiber (*.db)	4.00.6019.00	Microsoft
Microsoft Text Driver (*.txt; *.csv)	4.00.6019.00	Microsoft
Microsoft Text-Treiber (*.txt; *.csv)	4.00.6019.00	Microsoft
Microsoft Visual FoxPro Driver	1.00.02.00	Microsoft
Microsoft Visual FoxPro-Treiber	1.00.02.00	Microsoft
OpenLink Generic ODBC Driver	5.00.00.00	OpenLink
OpenLink Generic ODBC Driver (Unicode)	5.00.00.00	OpenLink
OpenLink Lite for DB2 (32 Bit)	5.00.01.13	OpenLink
OpenLink Lite for DB2 (32 Bit) (Unicode)	5.00.01.13	OpenLink
SQL Server	2000.81.7713.00	Microsoft

An ODBC driver allows ODBC-enabled programs to get information from ODBC data sources. To install new drivers, use the driver's setup program.

10.2.25.4 Configuring An OpenLink ODBC-JDBC Agent Bridge Based ODBC Data Source Name (DSN)

1. Open up the ODBC Administrator within your desktop control panel, and then Click on the "Add" button to indicate that you want to add a new ODBC DSN to the current list of installed ODBC DSNs

Figure 8.30. Admin Assistant



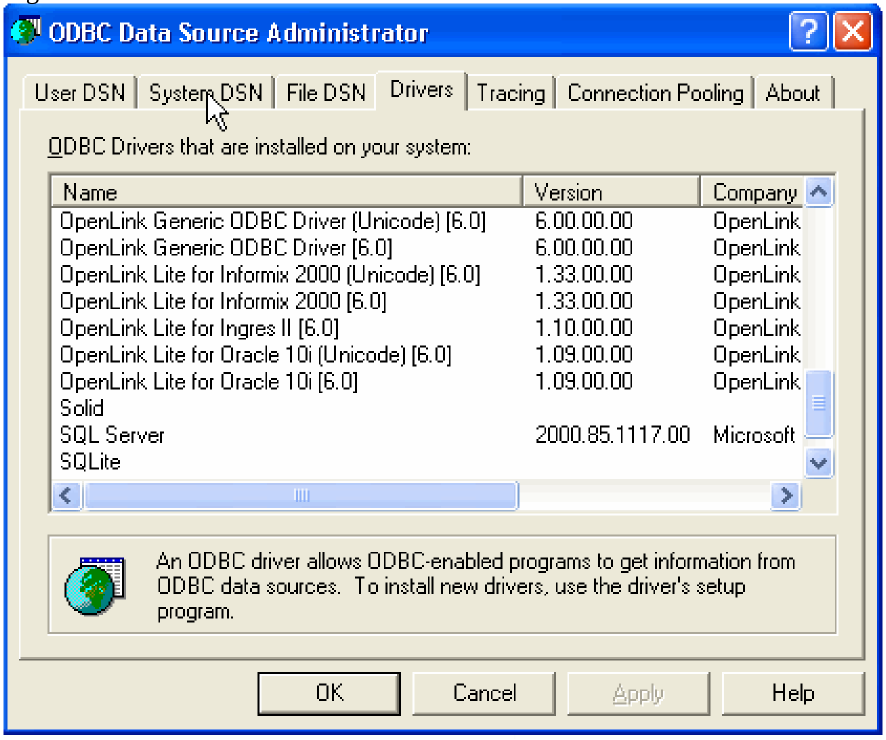
System Data Sources:

Name	Driver
Local Virtuoso	OpenLink Virtuoso Driver
Local Virtuoso Demo	OpenLink Virtuoso Driver

An ODBC System data source stores information about how to connect to the indicated data provider. A System data source is visible to all users on this machine, including NT services.

2. Click on the appropriate ODBC Driver that you will be associating this new ODBC DSN with, in this case the "OpenLink Generic 32 Bit Driver v4.0"

Figure 8.31. Admin Assistant



ODBC Drivers that are installed on your system:

Name	Version	Company
OpenLink Generic ODBC Driver (Unicode) [6.0]	6.00.00.00	OpenLink
OpenLink Generic ODBC Driver [6.0]	6.00.00.00	OpenLink
OpenLink Lite for Informix 2000 (Unicode) [6.0]	1.33.00.00	OpenLink
OpenLink Lite for Informix 2000 [6.0]	1.33.00.00	OpenLink
OpenLink Lite for Ingres II [6.0]	1.10.00.00	OpenLink
OpenLink Lite for Oracle 10i (Unicode) [6.0]	1.09.00.00	OpenLink
OpenLink Lite for Oracle 10i [6.0]	1.09.00.00	OpenLink
Solid		
SQL Server	2000.85.1117.00	Microsoft
SQLite		

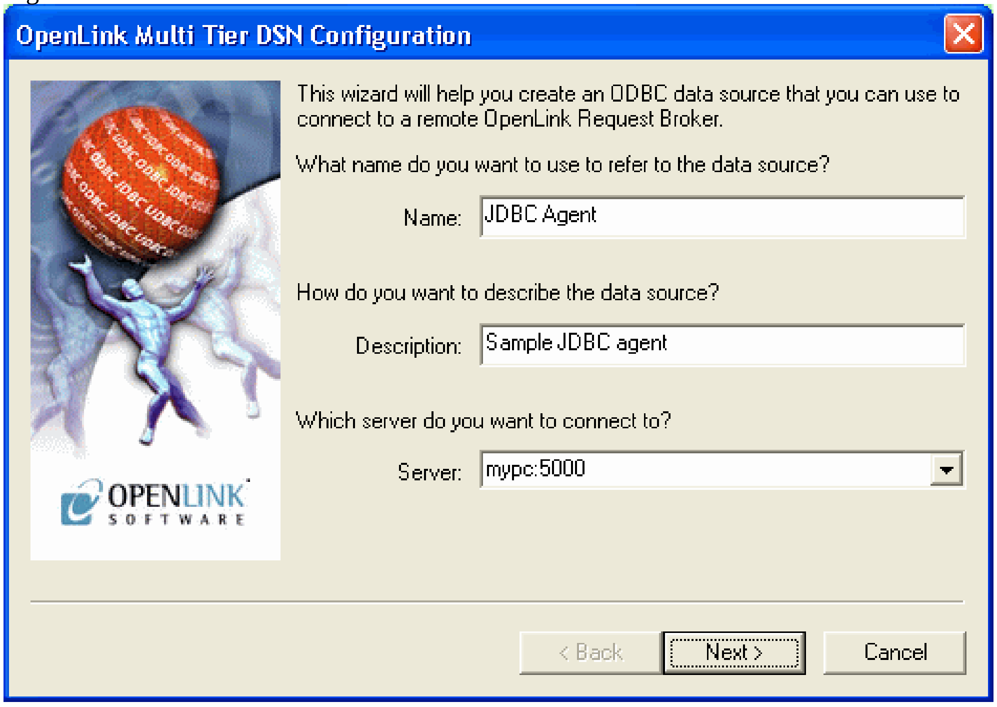
An ODBC driver allows ODBC-enabled programs to get information from ODBC data sources. To install new drivers, use the driver's setup program.

3. Choose a Name for your OpenLink ODBC DSN and then type it into the "Name" field, the example below presumes the DSN is to be called "JDBC Agent"

   Type a comment to describe the DSN.

   Enter the name and port of the machine hosting the OpenLink ODBC agent in the "Server" field. The example below presumes that the machine network alias for your desktop computer is "mypc" (note: you can also use the machines actual IP address or even use the "localhost" account if you are connecting to a local as opposed to remote JDBC driver). The port is the default of 5000.

   Figure 8.32. Admin Assistant



OpenLink Multi Tier DSN Configuration

This wizard will help you create an ODBC data source that you can use to connect to a remote OpenLink Request Broker.

What name do you want to use to refer to the data source?

Name: JDBC Agent

How do you want to describe the data source?

Description: Sample JDBC agent

Which server do you want to connect to?

Server: mypc:5000

4. Choose an OpenLink "Server Type" of "JDBCXX" from the "Domain" listbox where XX can be 12 or 13 to represent a JDK 1.2 or 1.3 installation is in use.

   Type the name of the JDBC driver you would like to use for your connection into the "Database" field. The example below assumes you are connecting to Oracle 8i using the JDBC driver provided by Oracle named "oracle.jdbc.driver.OracleDriver".

   Type the JDBC connect string for the Database you wish to connect to in the "Options" field. The example below assumes you are connecting to Oracle 8i using the JDBC connect string of "jdbc:oracle:oci8:@ORCL".

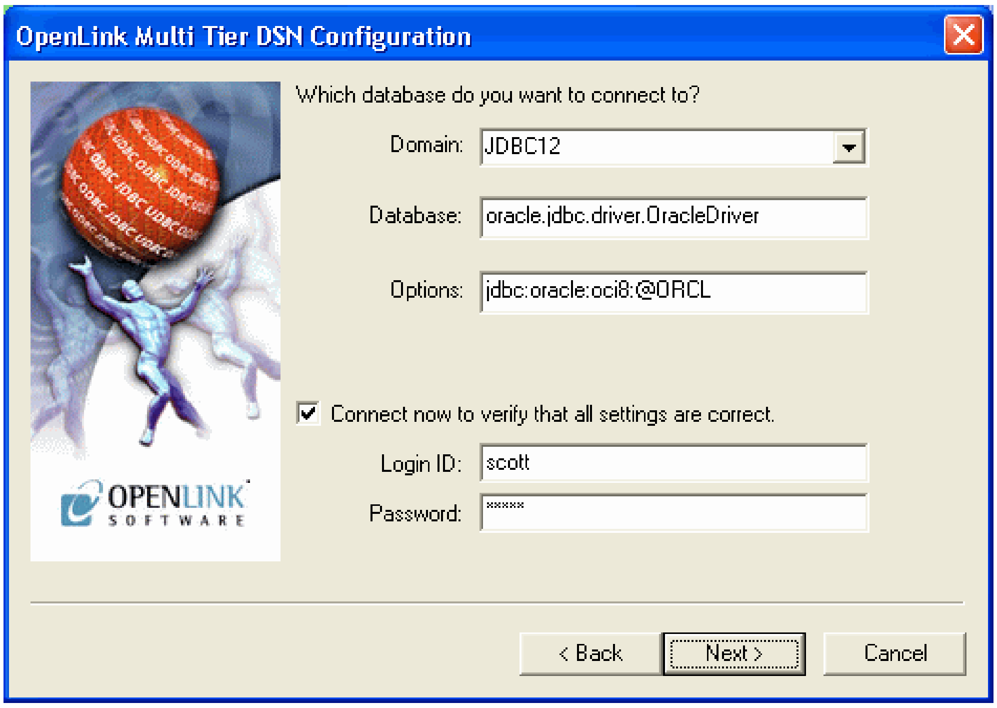
   When the "Connect now.." tick box is checked, a test connection is made to verify the Data Source connection.

   If there is no check then the Login ID and Password fields are ignored, and no test is performed.

   The Login ID is the default database UserID to use when logging on to a remote database engine (identified by the Domain above).

   Password is for the login of the above UserID.

   Figure 8.33. Admin Assistant


OpenLink Multi Tier DSN Configuration

Which database do you want to connect to?

Domain: JDBC12

Database: oracle.jdbc.driver.OracleDriver

Options: jdbc:oracle:oci8:@ORCL

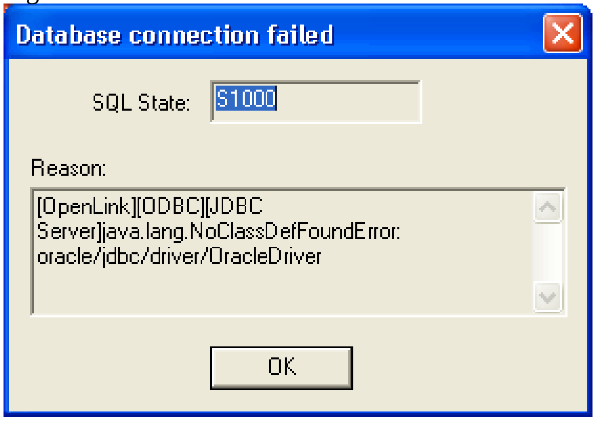
[✓] Connect now to verify that all settings are correct.

Login ID: scott

Password: *****

5. If a test is perfomed, then errors are reported like in this example:

Figure 8.34. Admin Assistant


Database connection failed

SQL State: S1000

Reason:

[OpenLink][ODBC][JDBC Server]java.lang.NoClassDefFoundError: oracle/jdbc/driver/OracleDriver

6. Now define additonal connection parameters:

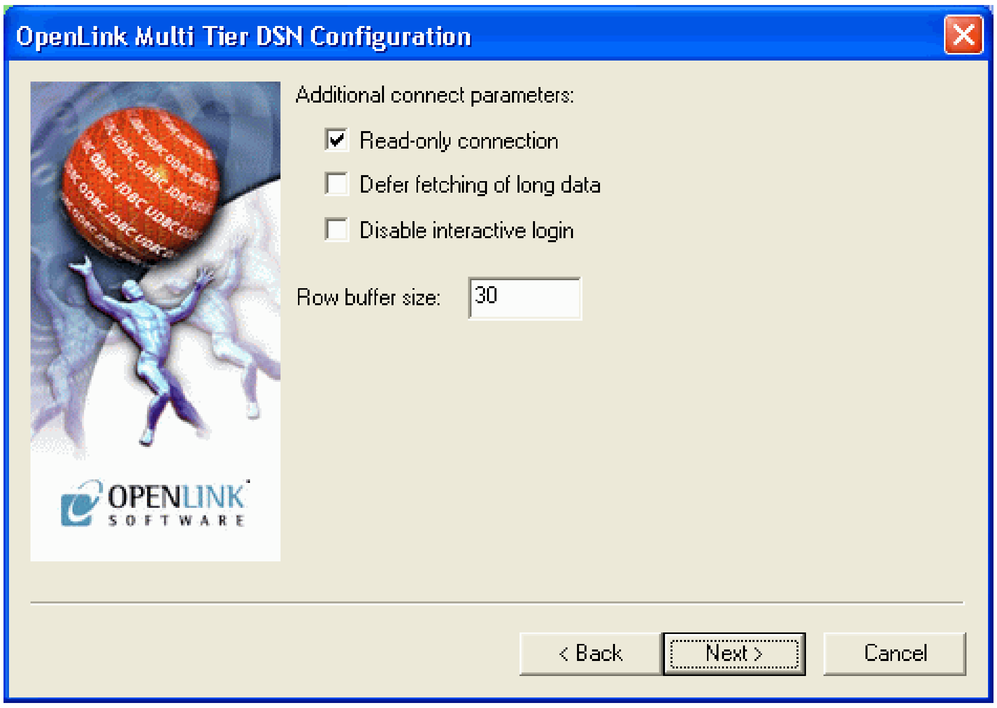
Read-only connection. Specify whether the connection is to be "Read-only". Make sure the checkbox is unchecked to request a "Read/Write" connection.

Defer fetching of long data. Check this box to defer the fetching of long data.

Disable interactive login. Suppress the ODBC "Username" and "Password" login dialog box when interacting with your ODBC DSN from within an ODBC compliant application.

Row Buffer Size. This attribute specifies the number of records to be transported over the network in a single network hop. Values can range from 1 to 99.

Figure 8.35. Admin Assistant



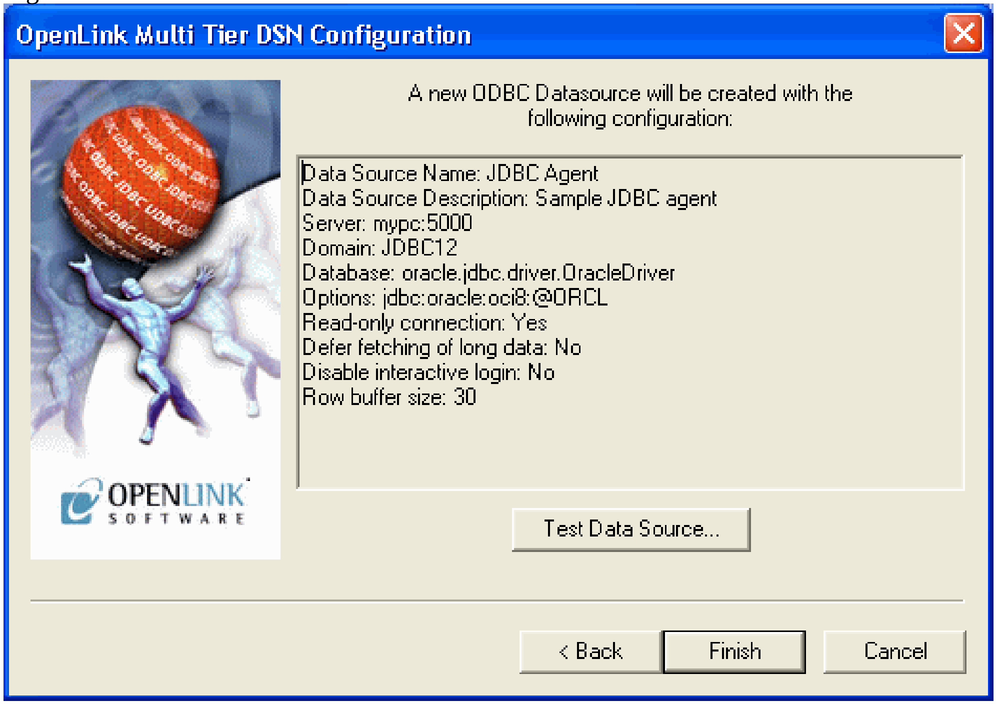
Additional connect parameters:

- [✓] Read-only connection
- [ ] Defer fetching of long data
- [ ] Disable interactive login

Row buffer size: 30

7. The details of the DSN are now shown. To test the DSN, press the Test Data Source button.

Figure 8.36. Admin Assistant

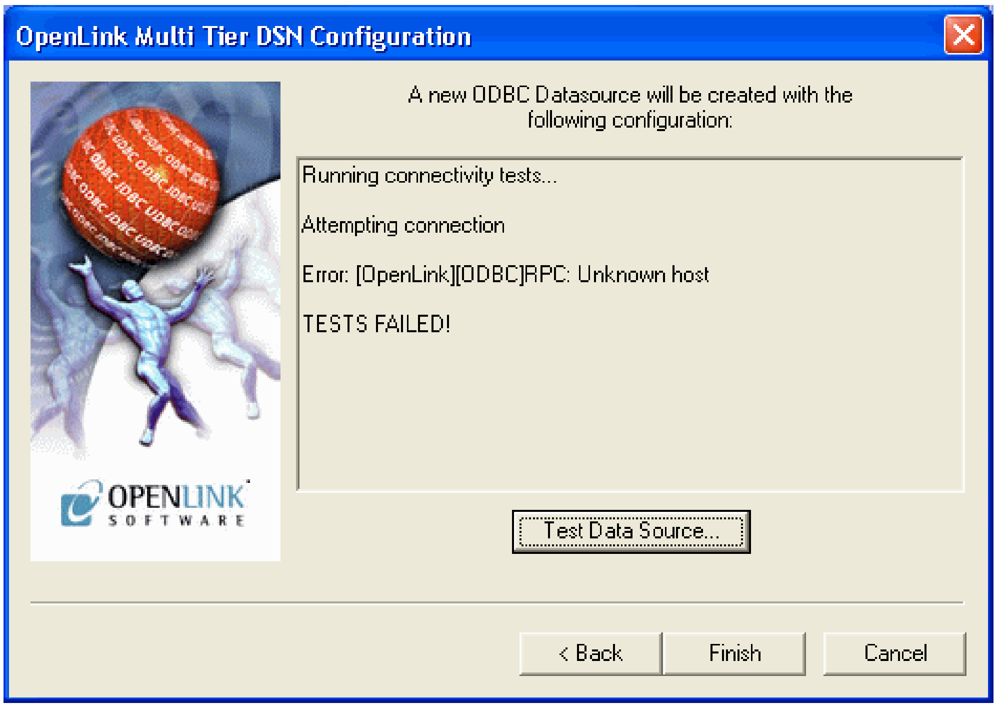


A new ODBC Datasource will be created with the following configuration:

```
Data Source Name: JDBC Agent
Data Source Description: Sample JDBC agent
Server: mypc:5000
Domain: JDBC12
Database: oracle.jdbc.driver.OracleDriver
Options: jdbc:oracle:oci8:@ORCL
Read-only connection: Yes
Defer fetching of long data: No
Disable interactive login: No
Row buffer size: 30
```

8. If a Test is requested then the status of the DSN test connection is shown. Here is an example error message:

Figure 8.37. Admin Assistant



A new ODBC Datasource will be created with the following configuration:

```
Running connectivity tests...

Attempting connection

Error: [OpenLink][ODBC]RPC: Unknown host

TESTS FAILED!
```

9. Click on the "Finish" button to complete the creation of your new OpenLink ODBC DSN.

10.2.25.5 Making A Test Connection To Your OpenLink ODBC-JDBC agent ODBC DSN

To verify that your installation and configuration is ready for use, please follow the steps below in order to make a test connection to the OpenLink ODBC DSN that you have just created:-

1. Start the OpenLink Request Broker in debug mode, this can be done from a DOS shell by executing the command : oplrqb -dv or from your Services Panel (note you must change the startup mode to manual to enable the OpenLink Request Broker run in Debug Mode)
2. Locate the program "VBDemo" or "C++ Demo" situated within the "OpenLink Data Access Drivers" group on your desktop (Windows start menu item)
3. Attempt to make a connection to the OpenLink ODBC DSN
4. If step 3 is successful and you see data exchanged between your ODBC Client and your OpenLink ODBC DSN, exit the ODBC application, and then shutdown and restart the Broker without the Debug Mode options using the command: oplrqb -v. If step 3 is unsuccessful repeat step 3 and then capture the Request Broker output and proceed to instigating contact with OpenLink Technical support via the OpenLink Support Page
5. Establish connection between your own ODBC Applications and the OpenLink ODBC DSN created in step 3
6. Shutdown the Request Broker using the command: oplshut -f

10.2.25.6 Adding An OpenLink ODBC-JDBC Agent To An Early OpenLink System

1. Place the ODBC-JDBC Agent executable in the openlink/bin sub-directory
2. Edit the OpenLink "Session Rules" Book (the file oplrqb.ini) using a text editor
3. Insert a new OpenLink Domain alias at the top of the "[Domain Alias]" section of the Rule Book in the manner depicted below:

```
JDBC12 = jdbc12
```
4. Insert a new OpenLink Session Mapping rule to the top of the "[Mapping Rules]" section of the Rule Book in the manner depicted below:

```
jdbc12:*:*:*:*:*:* = accept generic_jdbc12
```
5. Then create a new OpenLink Agent section as follows:

```
[generic_jdbc12]
Program = jdbc12_mv.exe
Environment = JDBC12
ReUse       = always
```

10.2.26 Note:

Program key must be set to the exact file name. (For Unix this would be jdbc12_mv)

6. Then create a new OpenLink Environment section as follows: (lines wrapped for presentation here)

```
[Environment JDBC12]
CLASSPATH = .;D:\Software\oracle\ora81\jdbc\lib\classes12.zip;D:\Softwa
re\oracle\ora81\jdbc\lib\nls_charset12.zip;c:\program files\openlink\vi
rtuoso 2.0\jdk1.2\virtjdbc2.jar;c:\program files\openlink\jdk1.2\opljdb
c2.jar
PATH = D:\Software\oracle\ora81\bin;D:\Software\oracle\ora81\jdbc\lib;D
:\Software\jdk1.2\jre\bin;D:\Software\jdk1.2\jre\bin\classic;C:\WINNT\S
YSTEM32;C:\WINNT
```

7. Save the file
8. Shutdown and restart your OpenLink Request Broker.

10.2.26.1 ODBC-JDBC agent

Table 8.17. Default Rule Book settings

Rule Book Section & parameters	Default Rule Book Settings	Notes
[Environment Jdbc12*] [Environment Jdbc13*]		Parameters are common to JDBC 1.2 and 1.3
JET_SQLSTATISTICSOFF=	FALSE	If it is set to 'TRUE', the function SQLStatistics returns an empty ResultSet. It helps to resolve the problem with the SQLStatistics for some jdbc drivers.
JET_DROPCATALOGFROMDBMETACALLS=	FALSE	If it is sets to 'TRUE', the NULL will be assigned to the field 'Catalog' in the ResultSet of : SQLTables, SQLColumns & SQLStatistics calls.
JET_DROPSCHEMAFROMDBMETACALLS=	FALSE	If it is sets to 'TRUE', the NULL will be assigned to the field 'Schema' in the ResultSet of : SQLTables, SQLColumns & SQLStatistics calls.
JET_NOSUPPORTOFQUOTEDIDENTIFIER=	FALSE	If it is sets to 'TRUE', the call SQLGetInfo for 'SQL_IDENTIFIER_QUOTE_CHAR' will return the space (" "). It can be used if DBMS doesn't support quoted SQL like select * from "account"
PATCHNULLSIZEOFSQLCHAR=		If DBMS is returning a field of type SQLCHAR or SQLVARCHAR with zero size, the size will be replaced with the value of PATCHNULLSIZEOFSQLCHAR

Typical settings for PointBase, HyperSonic and InstantDb are as follows:

```
[Environment Jdbc12PointBase]
;CLASSPATH =.;e:\java\pointbase\embedded\classes\pbembedded34EV.jar
;PATH =
JET_DROPCATALOGFROMDBMETACALLS = YES
JET_DROPSCHEMAFROMDBMETACALLS =  YES
JET_NOSUPPORTOFQUOTEDIDENTIFIER = YES
[Environment Jdbc12HyperSonic]
;CLASSPATH = .;e:\java\hypersonic\demo\hsql.jar
;PATH =
PATCHNULLSIZEOFSQLCHAR = 4096
JET_SQLSTATISTICSOFF = YES
[Environment Jdbc12InstantDB]
;CLASSPATH = .;e:\InstantDB\Classes\idb.jar
;PATH =
JET_NOSUPPORTOFQUOTEDIDENTIFIER = YES
```

```
JET_DROPSCHEMAFROMDBMETACALLS =  YES
```

10.2.27 OpenLink Proxy Agent

An OpenLink Proxy agent is a specialized Agent that acts on behalf of another remotely or locally situated OpenLink Database Agent. This Agent format is typically used in 3-Tier Internet based environments in which you place an Agent on an external machine (typically running your Web Server) and then have it masquerade for an actual database agent behind your organization's firewall.

A Proxy Agent can also be used in conjunction with the Session Rules Book for centralized configuration and control of all of your OpenLink Clients, by controlling the configuration of all of the OpenLink Session Elements on one or more server machines.

Like other Proxy services, an OpenLink Client connects to the Proxy Agent instead of to the actual service; the Proxy Agent then connects to the actual service that presumably lies on a machine that shouldn't normally be accessible from outside the network. With this setup, it is possible to grant selective access to databases that are otherwise not accessible from the Internet; this greatly enhances the functionality of data access standards like JDBC, ODBC, OLE-DB.

10.2.27.1 Installation

The OpenLink Proxy Agent is automatically installed with your Request Broker on any platform.

You have to install a Request Broker installation archive on the machine that is to act as a host for the OpenLink Proxy agent (typically the middle tier machine in a three 3-tier architecture). You then install another Request Broker archive and relevant Database Agents archives on the machine(s) hosting the backend database engine(s) that you are connecting to via an OpenLink Client.

10.2.27.2 Configuration

You configure the Proxy Agent like all other OpenLink Agents using the Admin Assistant. The process is broken into two parts, the first part involves creating a Proxy Agent Template, the second part involves a Session Rules that conditionally associates OpenLink Clients with the Proxy Agent Template that you have created.

The configuration guide that follows presumes that we are creating a Proxy on a middle-tier server for the OpenLink Virtuoso Database engine such that any OpenLink Client connection (ODBC, JDBC, UDBC, or OLE-DB) to this Server ends up being connected to Virtuoso. The steps that follow guide you through this process.

10.2.27.3 Creating Proxy Agent Initialization Template

1. Start the Request Broker your middle-tier server machine
2. Start a Web Browser session
3. Enter the following URL into your browser:

   If you started the Request Broker on your local machine enter:

   ```
   http://localhost:8000/
   ```

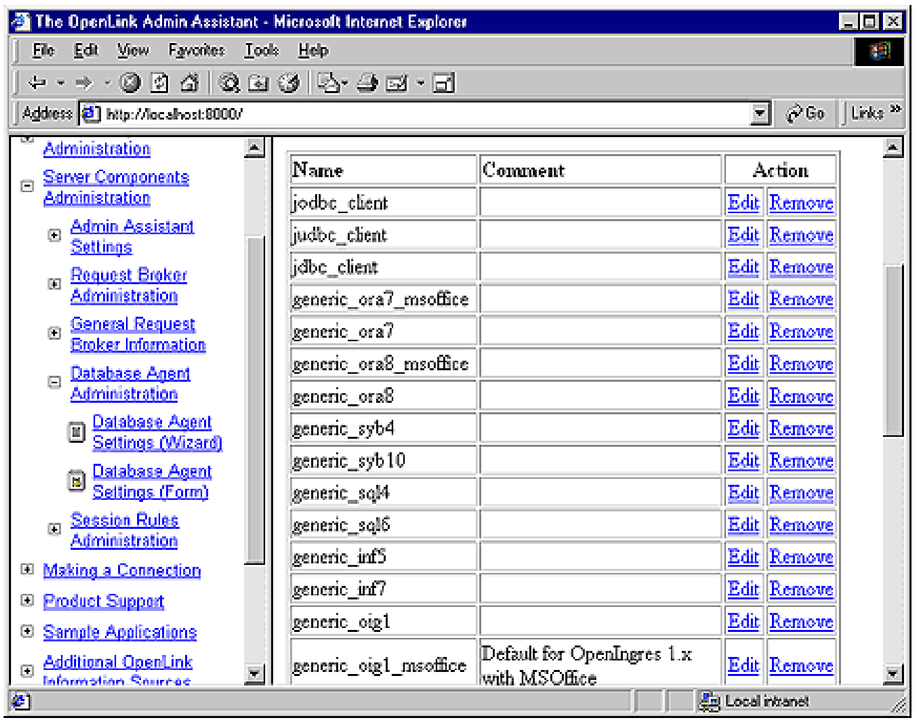
   (assuming you accepted port 8000 as the Admin Assistant port number at installation time).

   If the Request Broker in on another machine enter:

   ```
   http://<hostname or IP address>:8000/
   ```

   (assuming you accepted port 8000 as the Admin Assistant port number at installation time).
4. Navigate the Admin Assistant menu tree as follows: OpenLink Database Agent(s) Settings-->Database Agent(s) Settings (Form).

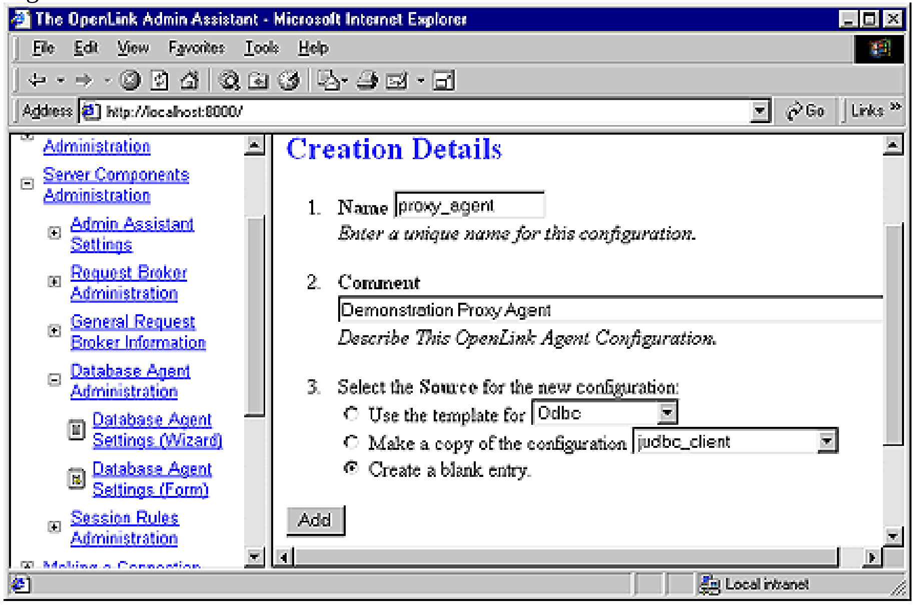
   Figure 8.38. Admin Assistant


The OpenLink Admin Assistant - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://localhost:8000/
Go
Links
Administration
Server Components Administration
Admin Assistant Settings
Request Broker Administration
General Request Broker Information
Database Agent Administration
Database Agent Settings (Wizard)
Database Agent Settings (Form)
Session Rules Administration
Making a Connection
Product Support
Sample Applications
Additional OpenLink Information Sources
Local intranet

Name	Comment	Action	
jodbc_client		Edit	Remove
judbc_client		Edit	Remove
jdbc_client		Edit	Remove
generic_ora7_msoffice		Edit	Remove
generic_ora7		Edit	Remove
generic_ora8_msoffice		Edit	Remove
generic_ora8		Edit	Remove
generic_syb4		Edit	Remove
generic_syb10		Edit	Remove
generic_sql4		Edit	Remove
generic_sql6		Edit	Remove
generic_inf5		Edit	Remove
generic_inf7		Edit	Remove
generic_oig1		Edit	Remove
generic_oig1_msoffice	Default for OpenIngres 1.x with MSOffice	Edit	Remove

5. Scroll to the bottom of the Agent Templates listing page and then click on the "Add" button, this opens up a default agent initialization template page, enter a Name and Description for your new Agent Initialization Template, then select the "create blank entry" radio button and then click the "Add" hyperlink. Reinitialize the Broker when prompted.

   Figure 8.39. Admin Assistant


The OpenLink Admin Assistant - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://localhost:8000/
Go
Links
Administration
Server Components Administration
Admin Assistant Settings
Request Broker Administration
General Request Broker Information
Database Agent Administration
Database Agent Settings (Wizard)
Database Agent Settings (Form)
Session Rules Administration
Making a Connection
Local intranet

Creation Details

1. Name proxy_agent
   Enter a unique name for this configuration.
2. Comment
   Demonstration Proxy Agent
   Describe This OpenLink Agent Configuration.
3. Select the Source for the new configuration:
   ○ Use the template for Odbc
   ○ Make a copy of the configuration judbc_client
   ● Create a blank entry.

Add

6. Navigate the Admin Assistant menu tree as follows: OpenLink Database Agent(s) Settings-->Database Agent(s) Settings (Form). Then locate the new Agent Template created in the previous step. Now click on the "Edit" hyperlink.

   Enter values into the following fields representing key OpenLink Session Elements:

   "User Name" - Leave empty (this is handled on the server using the Virtuoso Agent initialization Template)

   "Password" - ditto

   "Database Name" - ditto

"Read Only" - ditto

"Connect Options" - ditto

"Server Options" - ditto

"Server Type" - Virtuoso (you enter an valid OpenLink Domain values here, e.g Oracle 8, Informix 7, Progress 83B etc.)

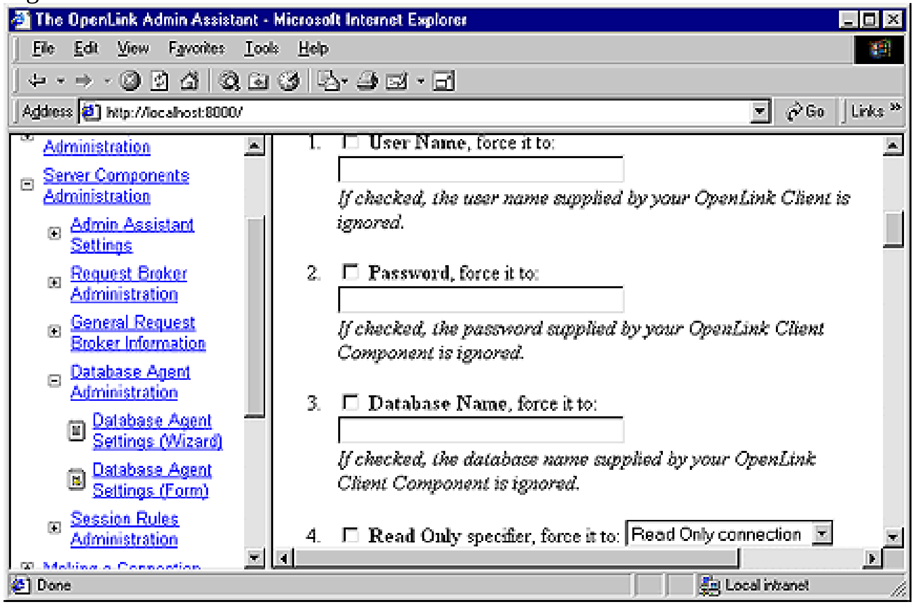
"Host Name" - enter IP address or network alias of database server machine. this examples presumes the IP address of the database server 123.123.123.100

"Executable Name" - enter "proxy_sv" (Linux or UNIX) or "proxy_sv.exe" (for Windows)

In the "Client-Server Mapping Process & Control" section of this page select the "Conditionally" radio button and then hatch the "When originating from same machine" checkbox. This ensures that each new OpenLink client machine has a distinct proxy agent process servicing all the ODBC, JDBC, UDBC, OLE-DB clients on that machine thereby restricting the number of new proxy agents processes initialized.

Click on the "Update" button and the reinitialize the Request Broker.

Figure 8.40. Admin Assistant


The OpenLink Admin Assistant - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://localhost:8000/
Go
Links
Administration
Server Components Administration
Admin Assistant Settings
Request Broker Administration
General Request Broker Information
Database Agent Administration
Database Agent Settings (Wizard)
Database Agent Settings (Form)
Session Rules Administration
Making a Connection
Done
Local intranet

1. User Name, force it to:

If checked, the user name supplied by your OpenLink Client is ignored.

2. Password, force it to:

If checked, the password supplied by your OpenLink Client Component is ignored.

3. Database Name, force it to:

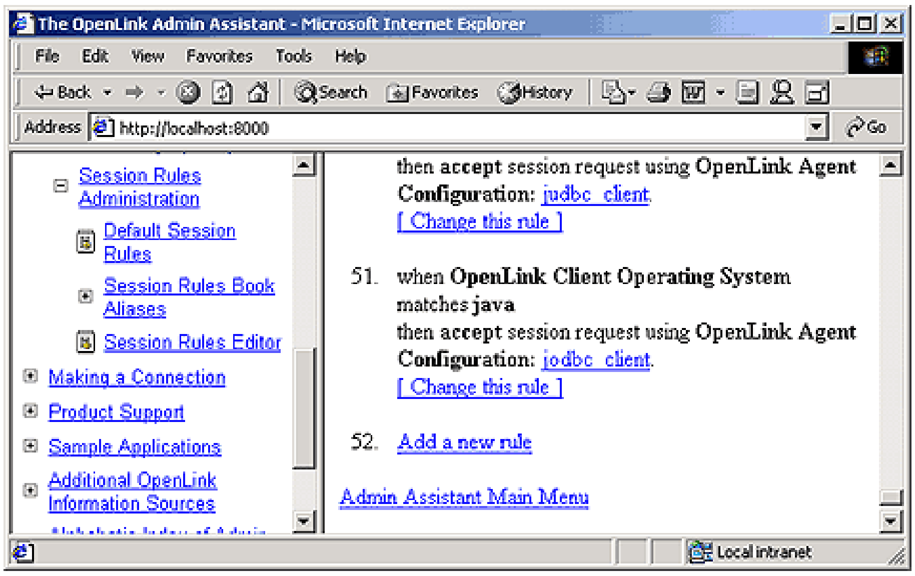
If checked, the database name supplied by your OpenLink Client Component is ignored.

4. Read Only specifier, force it to: Read Only connection

10.2.27.4 Creating Session Rule That Maps Connections to Proxy Agent's Template

1. Follow the "Server Components Administration"->"Session Rules Administration"->"Session Rules Editor" menu path which brings you to a screen identical to the one depicted below, this presents you with a list of existing session rules (all OpenLink installations come with a set of pre-configured session rules). Click on the "Add new rule" hyperlink to open up the session rule creation page.

   Figure 8.41. Admin Assistant


The OpenLink Admin Assistant - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://localhost:8000
Go
Session Rules Administration
Default Session Rules
Session Rules Book Aliases
Session Rules Editor
Making a Connection
Product Support
Sample Applications
Additional OpenLink Information Sources
Local intranet

then accept session request using OpenLink Agent Configuration: judbc_client.
[Change this rule]

51. when OpenLink Client Operating System matches java
then accept session request using OpenLink Agent Configuration: jodbc_client.
[Change this rule]

52. Add a new rule

Admin Assistant Main Menu

2. Create a new session rule by doing the following:

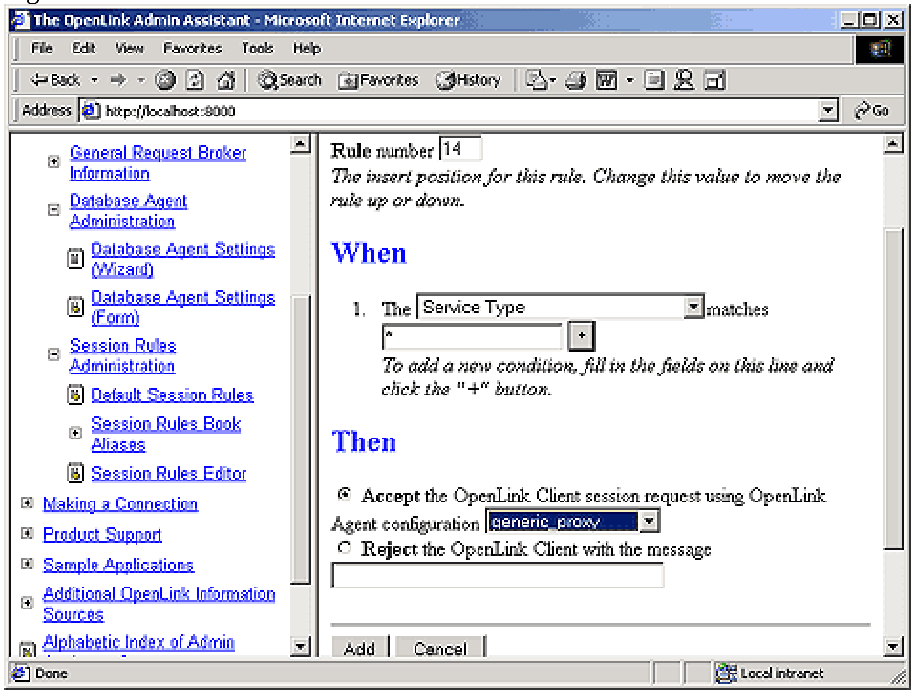
   Set the Rule Number field to 1 (this means that this rule gets evaluated before others)

   Leave the default value of "*" in the Server Type field this ensure that this rule applies to any Domain Type.

   Pick the "proxy_agent" initialization template from the agent initialization template list box used by the "Then" processing instruction to determine how calls associated with this rule are to be handled.

   Click on the "Add" button to save your new rule to the rule book. Then reinitialize the Request Broker.

   Figure 8.42. Admin Assistant


The OpenLink Admin Assistant - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://localhost:8000
Go
General Request Broker Information
Database Agent Administration
Database Agent Settings (Wizard)
Database Agent Settings (Form)
Session Rules Administration
Default Session Rules
Session Rules Book Aliases
Session Rules Editor
Making a Connection
Product Support
Sample Applications
Additional OpenLink Information Sources
Alphabetic Index of Admin
Add
Cancel
Done
Local intranet

Rule number 14
The insert position for this rule. Change this value to move the rule up or down.

When

1. The Service Type matches
*
+
To add a new condition, fill in the fields on this line and click the "+" button.

Then

Accept the OpenLink Client session request using OpenLink Agent configuration generic_proxy
Reject the OpenLink Client with the message

3. Create an OpenLink ODBC, JDBC, UDBC or OLE-DB client session with the Domain Type attribute set to Proxy and the Host attribute set to your middle-tier server.

   If your database server is behind a firewall you need to perform the following additional additional steps:

1. Enable UDP support, and then make port 60001 available, this is the port number used by the OpenLink Request Broker. Since we are now connecting to a database server running the Request Broker that resides behind your firewall we need to open up this port.

2. Start the Request Broker on the Database Server
3. Start a Web Browser session and then initialize the Admin Assistant running on the Database Server machine by entering the following URL:

   ```
   http://<hostname or IP address of Database Server machine>:8000/
   ```

   (presuming you took the default number of 8000 for the Web Assistant at install time on the Database Server).
4. Navigate the Admin Assistant menu tree to: Server Components Administration--> Request Broker Administration-->Edit Request Brokers Parameters (Form).

   Locate item number 4 on the form which reads "Only use ports in the range...", enter a range of TCP port numbers that you have enabled within your firewall software. The Broker will then automatically starts the Virtuoso agent (or other database agents depending on your settings) on the first available port in the range.

10.3 Database Configuration for Unicode

Below are instructions on the configuration of the Unicode enabled drivers and databases for testing. Typically the first task to be performed is the creation of a Unicode enabled Database, which for most databases means configuring them to store data using the UTF8 encoding.

10.3.1 Oracle 8 & 9

The Oracle 9i, 8i and 8.0 databases store Unicode data in the UTF8 encoding scheme, which is an ASCII compatible multibyte encoding for Unicode.

10.3.1.1 Database Confguration

Using the Oracle Database Configuration Assistant wizard follow the options for creating a new database, selecting the custom option when presented and you will during the configuration of this Custom database be allowed to Change the Character Set , at which point this can be changed to UTF8.

To check the character set in use by your database, execute the following query in SQL*Plus:

```
SQL> SELECT parameter, value FROM nls_database_parameters
   WHERE parameter = 'NLS_CHARACTERSET';
PARAMETER             VALUE
-----------------     --------------------
NLS_CHARACTERSET      UTF8
SQL>.
```

Unicode support is dependent on the Unicode features available through the Oracle Call Interface (OCI). OCI 8.1.5 supports inputting Unicode data into a database and retrieving Unicode data from a database.

The Following Oracle Data types can be using for storing Unicode data:

```
CHAR
VARCHAR
VARCHAR2
```

10.3.1.2 Driver Configuration

The Oracle configuration parameter for control character sets is the NLS_LANG environment variable, which should be set to the correct character set for your client. Oracle 8.1.7 claims to be capable of dynamically determining the character set in use on the client and does not require the NLS_LANG to be set, but it is not a bad idea to set it anyway.

Additional information on Oracle Unicode support can be found from otn.oracle.com

10.3.2 Informix 9.x

10.3.2.1 Database Configuration

When a database is created, the DB_LOCALE in effect at the time is stored in the system catalog and used throughout the lifetime of the database. Using DBACCESS, create a new database with DB_LOCALE set beforehand to EN_US.UTF8 . Specifying UTF8 as the codeset allows the creation of schema objects with names which can contain multibyte characters.

The database locale being used by a database can be viewed in DBACCESS using the menu commands: Database > Info > NLS.

The documentation for Informix GLS states that DB_LOCALE is also used to correctly interpret the locale sensitive datatypes NCHAR and NVARCHAR. The code set specified in DB_LOCALE specifies which characters are valid in any character column as well as the names of database objects.

10.3.2.2 Setting the Client Locale

The codeset to be used by an Informix client application is specified as part of the client locale. The client locale takes the form:

```
language_territory.codeset[@modifier]
```

An Informix 9 Lite driver or agent should use UTF-8 as the codeset. The language and territory should not matter; so it should be possible, for example, to use French (fr_fr) or American English (en_us). For Informix clients on Windows, the client locale is typically set through SetNet32. Rather than rely on the SetNet32 settings, our agent or Lite driver instead sets the client locale at runtime. For an Informix Lite driver on Windows, you must manually add an entry to the registry to set the client locale under the entry for the appropriate DSN in the ODBC.INI hive, add the value:

```
ClientLocale:REG_SZ:<client locale>
e.g.
ClientLocale:REG_SZ:EN_US.57372
```

This example uses a codeset number (57372) rather than a codeset name (UTF8) to specify UTF-8 as the codeset. Either form can be used. The registry file included in an Informix client installation lists the supported code sets and the correspondence between codeset names and numbers.

10.3.2.3 Driver Configuration

For an Informix agent (on Windows or Unix), specify the CLIENT_LOCALE environment variable setting in the [Environment INFORMIX] section of the rulebook. For an Informix Unix Lite driver, set the CLIENT_LOCALE environment variable appropriately.

10.3.3 Sybase 12.5 +

10.3.3.1 Database Configuration

The pre-requisites for Unicode with Sybase are:

- Sybase Active Server (ASE) Version 12.5 or later. (Unicode support is NOT enabled for Version 12.0);
- Default character set for the Sybase Server needs to be "UTF-8".

To set this:

1. Make sure the Sybase SQL Server is not running. (Cancel it from the "Services" screen).
2. Run "Server Configuration" from the "Sybase" entry in the Task Menu bar, or run SYCONFIG.EXE directly;
3. Either Create a new Active Server or Configure an existing Active Server. (Selection is via Pushbuttons on the dialog box);
4. For either method, select the "Language" pushbutton;
5. Select the "Character Set" pushbutton;
6. Select the "Set Default" pushbutton;
7. Select "Unicode 3.0.1 UTF-8" from the list box.

If this entry is not available, you will have to add it. From the Character Set selection dialog box, select the "Add / Delete" pushbutton. Select the character set from the list box of those available. Select the "Add" pushbutton (or the "Add All" pushbutton to make all character sets available). Select OK. One the default character set has been selected, select "OK" and "Exit". Start (or restart) the Sybase SQL-Server.

10.3.3.2 Driver Configuration

There is no need to set anything at the Client end. The character set in use is actually set using Sybase locale functions at connection time. However, it may be useful to ensure that "utf8" is one of the enabled character sets for the relevant platform, in the file [SYBASE]/locales/locales.dat.

10.3.4 Progress 9.1 (SQL-92)

10.3.4.1 Database Configuration

The Progress database can be run in the UTF-8 Unicode codepage. The sql-92 client can be also be run in unicode. The SQL-92 server uses the codepage of the connected database as its internal codepage. Conversion between the database codepage and the SQL-92 client codepage is done by the server. There are no specific functions provided for converting between codepages within an ESQL-92 program.

The easiest method to create to Progress Unicode enabled database is to use the proutil program to convert an existing database to utf8 format using the following command:

```
proutil <db-name> -C convchar convert utf-8
```

Multibyte characters can be used in character and varchar fields. Character string literals and the arguments to string functions can also be multibyte characters. There are some provisos for specific functions noted in the documentation. In addition, when the SQL-92 language element syntax requires single quotes, double quotes, parentheses, or braces, the requirement is for the single-byte ASCII encoding of these characters, and other encodings are not equivalent. The string operators in Progress SQL-92 consider the unit of length to be the character count, not a byte count or a column count.

When a column of type CHAR or VARCHAR is created the maximum length specified is a number of characters so the actual number of bytes storage required depends on the database codepage. The length of character data returned in the sqlda is in bytes not characters.

10.3.4.2 Driver Configuration

For ESQL-92 clients the internal codepage is determined by the value of the client's SQL_CLIENT_CHARSET environment variable, if set. Otherwise, the internal codepage is that of the client's locale. There is a similar environment variable that controls the codepage of messages sent by the database server.

10.3.5 DB/2 v7.x

10.3.5.1 Database Configuration

Using the DB/2 Control Center create a new database instance using the wizard provide. During the create of this database you will be prompted to specify the locale for the new database, which should be set to a code set type of UTF-8. Unicode data can be stored in the following DB/2 datatypes:

- GRAPHIC
- VARGRAPHIC
- LONGVARGRAPHIC
- DBCLOB

10.3.5.2 Driver Configuration

There are no specific environment variables that need to be set for the DB/2 Driver to handle Unicode data. One special consideration when inserting Unicode data into the daatbase though is that you cannot insert literal Unicode values into the database. Instead these values have to be inserted as bound parameters as follows:

```
CREATE TABLE UTEST (F1 GRAPHIC(20), F2 VARGRAPHIC(20), F3 LONG VARGRAPHIC,
                F4 DBCLOB(100));
                 Successfully connected to DSN 'UO_db2'.
        SQLBindParameter:
          In: StatementHandle = 0x00751860, ParameterNumber = 1,
              InputOutputtype = SQL_PARAM_INPUT=1, ValueType = SQL_C_WCHAR=-8,
              ParameterType = SQL_WCHAR=-8, ColumnSize = 0, DecimalDigits = 0,
              ParameterValuePtr = "?????", BufferLength = 0,
              StrLen_or_IndPtr = SQL_NTS=-3, SQL_LEN_DATA_AT_EXEC = FALSE,
              Buffer Size = 600
           Return:       SQL_SUCCESS=0
        SQLExecDirect:
           In: Statementhandle = 0x00751860, StatementText = "insert into utest(f1)
        values(?)", Statementlength = 31
           Return:       SQL_SUCCESS=0
        SQLExecDirect:
           In: Statementhandle = 0x00751860, StatementText = "select * from utest",
        Statementlength = 19
           Return:       SQL_SUCCESS=0
        Get Data All:
        "F1", "F2", "F3", "F4"
        "АБВГД               ", <Null>, <Null>, NO DATA
        1 row fetched from 4 columns.
```

This is because the Graphic string data types are compatible only with other graphic string data types, and never with numeric, character string, or datetime data types.

Note that additional Unicode support has been added to the DB/2 agent for VARCHAR, LONGVARCHAR, CLOB & BLOB types, although a specific Patch (FIXPAK7) is required from IBM to obtain this support in DB/2 v 7.2 databases and FIXPAK 3 & 7 are required for DB/2 v7.1 databases.

The application code page must be set to UTF-8, which can be done by issuing the command:

```
              db2set DB2CODEPAGE=1208
```

on the client (DB2 Lite) or server (DB2 agent) as appropriate.

10.3.6 MS SQLServer 2000

There are no Unicode-specific settings for SQLServer. When creating a Database, the collation type for the database can be specified but there is no UTF8 or Unicode specific setting, and a wide (Unicode) language type like Chinese or similar has to be selected, after which wide (Unicode) data can be inserted into the SQLServer wide character types NCHAR & NVARCHAR.

10.3.7 Operational Notes

If you are debugging a unicode connection, you can expect to see this in the request broker log - note the serveropts field:

```
        ...
        14:08:11 using mapping: db2:*:*:*:*:*:*
        14:08:11 using [generic_db2] ServerProgram=db2_mv
        14:08:11 connect params: domain=DB2 db=sample serveropts=W readonly=0
        ...
```

The Unicode parameters that are supplied to the server options cannot be displayed properly in the broker log so the above will be seen; this is normal behaviour.

10.4 Obtaining Agent Information

OpenLink Database agents have a specific naming convention, reflecting the identification of a particular database engine, and in some case specific versions of a given database engine.

The OpenLink executable binary file naming convention consists of three distinctive logical parts:

<db_engine>[<db_ver>]_<[sv|mv]>[.exe]

Database Engine - first three or four characters

Database Version - next two or three characters (depending on database engine version number-functionality issues)

OpenLink Service Provider Identifier - the characters "_sv" for single threaded or "_mv" for multi-threaded.

On a Windows system, the file extension is .exe which is shown as optional in this description but omitted in the table below.

The table below shows you how the current pool of OpenLink agents are identified based on the convention described above:

Table 8.18. OpenLink Service providers

Executable Binary File	**Database Engine & Version**
virt_mv	OpenLink Virtuoso
inf7_mv	Informix 7.x (Windows)
inf71_sv	Informix 7.1 (not Windows)
inf72_sv	Informix 7.2 (not Windows)
inf73_mv	Informix 7.3 (not Windows)
mys3_mv	MySQL 3.x
oig1_sv	OpenIngres 1.x
oig2_sv	Ingres II
pgr7_mv	PostgreSQL 7
pro73c_sv	Progress 7.3C
pro73e_sv	Progress 7.3E
pro82a_sv	Progress 8.2A
pro82c_sv	Progress 8.2C
pro83a_sv	Progress 8.3A
pro83b_sv	Progress 8.3B
pro90b_sv	Progress 9.0B
pro91a_sv	Progress 9.1A
pro91b_sv	Progress 9.1B
prs91_sv	Progress 91 SQL-92
pro81as_sv	Progress 8.1A small (Windows)
pro82as_sv	Progress 8.2A small (Windows)
pro82cs_sv	Progress 8.2C small (Windows)
pro83as_sv	Progress 8.3A small (Windows)
pro83bs_sv	Progress 8.3B small (Windows)
pro90bs_sv	Progress 9.0B small (Windows)
pro91as_sv	Progress 9.1A small (Windows)
pro91bs_sv	Progress 9.1B small (Windows)
ora7_sv	Oracle 7.x
ora803_mv	Oracle 8.0.3
ora805_mv	Oracle 8.0.5
ora81_mv	Oracle 8.1.x (8i windows)
ora8i_mv	Oracle 8i (not windows)
ora9i_mv	Oracle 9i (windows)
sybc10_mv	Sybase 10.x (CTLib)
sybc11_mv	Sybase 11.x (CTLib)
sql6_mv	Microsoft SQL Server 6.x & 7.x
sql2k_mv	Microsoft SQL Server 2000
db2_mv	DB2

10.4.1 Specific Agent Information

In a manner similar to the Request Broker, you can obtain component version, release, and functionality related information about your database agent through your operating system's command line interface.

To obtain the information about your database agent simply type in the name of the binary executable file for the relevant agent and the --help switch. The example below shows how this is done assuming you are seeking information about the OpenLink Virtuoso database agent:

```
virt_sv --help
```

The output returned is depicted below:

```
OpenLink Virtuoso Database Agent
Version 1.2 (Release 4.0) as of Mon Jul 31 2000 (cvsid 00051).
Compiled for Linux 2.4.0-test1 (i686-pc-linux-gnu)
Copyright (C) OpenLink Software.
Usage:
virt_sv [-CmijrlLd] [+noautocommit] [+maxrows num] [+initsql arg] [+jetfix]
[+norowsetlimit] [+loglevel num] [+logfile arg] [+debug]
+noautocommit          turn autocommit off by default
+maxrows               maximum allowed rows to fetch
+initsql               execute SQL from this file for every connection made
+jetfix                Jet Engine Compatibility Features
+norowset              disable the limit rowsetsize
```

Version - relates to the actual source code archive from which a database agent has been assembled.

Binary Platform - indicates what platform the database agent has been built to run on.

Usage - describes the command line options that are to be used with a particular database agent, these can be database agent specific. The command line options are to be used within the Database Agent configuration section of "Session Rules Book" (the file oplrqb.ini), this is the section that handles database agent specific items. You can add these entries as required to the rule book, you do so by manually editing the rule book file or via the Web based Admin Assistant (see section on Configuring Database Agents).

10.5 Linking OpenLink DB Agents

10.5.1 General Linking Approach

To relink Progress database Agents, see the next section.

It is sometimes necessary to link or re-link your OpenLink database agents when one of the following situations arise:

- No database agent executable was supplied with the OpenLink server components installed on your OpenLink server machine
- A database agent executable file exists but is unable to converse with your database server due to version incompatibility

OpenLink provides a database agent relink script for each supported database engine, this script is named using the following convention:

- ld - indicates a link script
- XXX - first three letters (perhaps 4) of database name
- NN - major and perhaps minor release number
- mv - If the agent is multhreaded

Examples: ldora81mv is Oracle 8i mulithreaded agent. ldoig2 is for Ingres II.

10.5.1.1 Relink Steps

1. Move to the "lib" sub-directory of your OpenLink server components installation directory e.g: "/usr/openlink/lib"
2. Setup key OpenLink link script environment variables which create a reference to key database environment variables required for making a native connections to your database engine:

```
DB2
DB2DIR=$DB2INST
Informix
INFORMIX5DIR=$INFORMIXDIR
INFORMIX6DIR=$INFORMIXDIR
INFORMIX7DIR=$INFORMIXDIR
Ingres
INGRES6DIR=$II_SYSTEM/ingres
INGRES7DIR=$II_SYSTEM/ingres
Oracle
ORACLE6DIR=$ORACLE_HOME
ORACLE7DIR=$ORACLE_HOME
ORACLE8DIR=$ORACLE_HOME
Solid
SOLIDDIR=$SOLID
Sybase
SYBASE10DIR=$SYBASE
SYBASE11DIR=$SYBASE
SYBASE4DIR=$SYBASE
Unify
UNIFYDIR=$UNIFY../lib
Velocis
VELOCISDIR=$VELOCIS
Virtuoso
VIRTUOSODIR=$VIRTUOSO
```

   You can also opt to manually configure these values in the "dbpaths" file situated in the same directory as your database agent link script.
3. Ensure that the "lib" subdirectory that your are working within is also part of your system PATH, if this isn't the case add this directory to the system PATH.
4. Run the appropriate link script (ldXXXNN) which will attempt to build a new database agent (XXX_sv or XXX_mv). Once the relink process is completed the new database agent executable is placed in the "bin" subdirectory directory of your OpenLink server components installation using. The new database agent binary will be named accordingly using the "XXX_sv" or "XXX_mv" database agent naming convention.

10.5.2 Relinking Progress Agents

The files described below should all be part of current agent installations.

We have attempted to provide a set of generic link scripts for the versions of Progress we support, to enable you to relink if necessary.

Please note that to relink against Progress 6 you require a HLI/C license from Progress in order to do so. For Progress 7, 8, 9 this is not required.

Remember to back up OpenLink files before you begin.

Follow the instructions below to relink an OpenLink Progress Database agent:

1. Change directory to the 'openlink/lib' directory.
2. Ensure the Progress environment variable DLC is set correctly.
3. Set the Progress PROLOAD environment variable to $DLC/probuild.
4. Run the 'mkdlcX' script which will produce a file called 'dlcX.o', where X represent Progress version 6, 7, 8, 9 depending on the major version number of your Progress installation.
5. If you receive an error stating that any of the Progress libraries cannot be found, edit the "mkdlcX" script and then remove the reference to the named file. Then rerun it "mkdlcX".

6. Run the script 'ldproX' which will build a new OpenLink Progress X agent (proX_sv or proX_mv), and place it directly in the 'openlink/bin' directory.

10.5.3 Note:

Relinking is only necessary if you are running a different version of Progress to the one used for linking the agents provided

10.5.4 Note:

If you cannot find the files described above with your installation then you may have an older version of the software. Please contact OpenLink Support for assistance at: http://www.openlinksw.com/support/suppindx.htm .

10.6 Progress Troubleshooting & Advanced Configuration

Assumptions. It assumed at this point that you have successfully installed OpenLink's client and server components on one or more computers within your I.T. infrastructure.

Conceptual Overview. Understanding the fundamental concepts that drive the OpenLink Data Access Drivers for Progress is central to you exploiting the deliberate simplicity of our product architecture. A failure to grasp these concepts will certainly make the last sentence an absolute contradiction in terms.

10.6.1 OpenLink Data Access Components

Your OpenLink Data Access Driver Suite (Multi-Tier Edition) consists of the following components, subdivided by prime function:

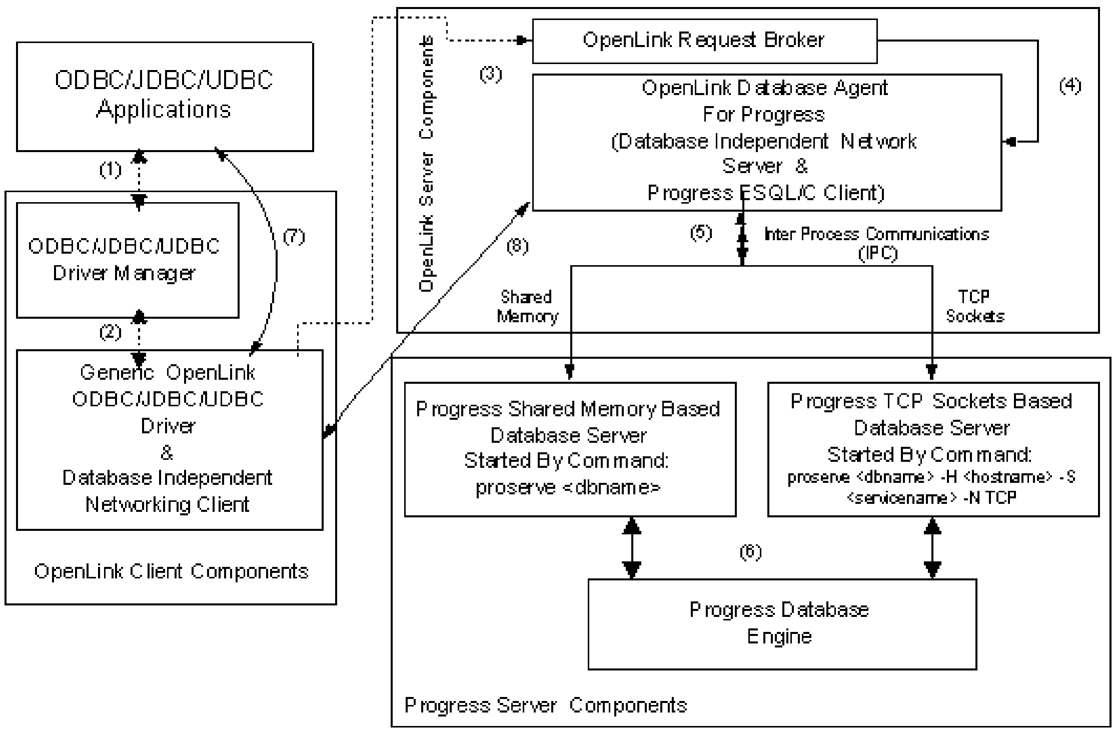
- Client Components (OpenLink Database Agent Service Consumers)
    - Generic ODBC Driver (the file "oplodb32.dll" under windows or OS/2, "oplodbc.sl" or "oplodbc.so" under UNIX)
    - Generic Driver for JDBC (the files "opljdbc.zip" or "opljdbc.jar")
    - Generic UDBC Driver (the end product of statically linking the file libudbc.o or dynamically linking the file "libudbc.so" or libudbc.sl" under UNIX)
- Server Components (OpenLink Database Agent Service Providers)
    - OpenLink Request Broker (the file "oplrqb" or "oplrqb.exe" depending on operating system)
    - OpenLink Database Agents (the files "pro63e_sv" or "pro73c_sv" or "pro82a_sv" depending on major and minor Progress Database Server version and sub-version number)

The prime function of your OpenLink Components does not determine their physical location within your I.T. infrastructure, this is driven instead by the logical structure and nature of your database connectivity requirements.

10.6.2 How OpenLink's Data Access Drivers Connect To Your Progress Database

A process diagram is provided below to assist with the understanding of how the various OpenLink components work in concert, with the sole objective of providing you with Secure, High-Performance, and Highly Configurable access to Progress Database Engines.

Figure 8.43. Progress Connection Conceptual Diagram


ODBC/JDBC/UDBC Applications
(1)
ODBC/JDBC/UDBC Driver Manager
(2)
Generic OpenLink ODBC/JDBC/UDBC Driver & Database Independent Networking Client
OpenLink Client Components
(7)
(8)
OpenLink Server Components
OpenLink Request Broker
(3)
(4)
OpenLink Database Agent For Progress (Database Independent Network Server & Progress ESQL/C Client)
(5)
Inter Process Communications (IPC)
Shared Memory
TCP Sockets
Progress Shared Memory Based Database Server Started By Command: proserve <dbname>
Progress TCP Sockets Based Database Server Started By Command: proserve <dbname> -H <hostname> -S <servicename> -N TCP
(6)
Progress Database Engine
Progress Server Components

The steps explained:

1. An ODBC/JDBC/UDBC Application uses the relevant data access standards Driver binding mechanism to locate and associate itself with the relevant OpenLink Driver.
2. The chosen OpenLink Driver then uses its in-built Database Independent Communications layer to broadcast an OpenLink Database Agent service request to the OpenLink Request Broker.
3. The OpenLink Request Broker analyzes the request in step 3 and then determines from the information obtained from the OpenLink Sessions Rules Book (the file oplrqb.ini), if an OpenLink Database Agent for Progress is available. If the OpenLink Request Broker determines that a Progress Database Agent does not exist, it will indicate it findings via an error condition and resultant error message viewable via your ODBC/JDBC/UDBC application.
4. The OpenLink Request Broker initializes a new OpenLink Database Agent For Progress instance or associates the new Progress Database session request with an existing OpenLink Database Agent For Progress instance.
5. OpenLink Database Agent For Progress instance uses your chosen Inter Process Communications (IPC) mechanism to initialize a session with a Progress Database Server that supports the chosen IPC mechanism. The default mechanism chosen by the OpenLink Database Agent For Progress is shared memory. You can opt to use TCP sockets by doing one of the following (as long as you have started a TCP sockets based Database Server for your Progress Database Engine by using the command:

   ```
   proserve &lt;dbname&gt; -S &lt;service name&gt; -N TCP  H localhost):
   ```

   Place the Progress Database Connection Parameters `- S <service name> -N TCP  H localhost` into the "Options" field of the OpenLink ODBC Data Source Configuration Utility, should you be using OpenLink ODBC Drivers.

   Place the Progress Database Connection Parameters `- S <service name> -N TCP  H localhost` as "/OPTIONS" JDBC URL attribute values, should you be using OpenLink Drivers for JDBC .Place the Progress Database Connection Parameters `- S <service name> -N TCP  H localhost` as "OPTIONS" connect string values, should you be using OpenLink UDBC or OpenLink s Unix based ODBC Drivers.

   The OpenLink Session Rules Book (oplrqb.ini) allows you to control this behavior centrally, alleviating yourself from the cost of maintaining numerous client based connection parameters. This is done by placing the Progress Database Connection Parameters `- S <service name> -N TCP  H localhost` as "ConnectOptions" section-key values within the Progress Database Agent configuration ("[generic_prox]") section of the Session Rules Book.
6. At this point your Progress Database Session is established

7. ODBC/JDBC/UDBC Driver binding and database session establishment is completed, and the ODBC/JDBC/UDBC Driver Manager is no longer required while the Progress Database Session remains open.
8. The OpenLink Request Broker like the Driver Managers in step 7 is no longer required (until session closure) while data is being exchanged directly between ODBC/JDBC/UDBC applications, the OpenLink Database Agents for Progress, and your Progress Database Engine.

10.6.3 Initial Connection Problems

The OpenLink Database Agent for Progress is a Hybrid of OpenLink & Progress ESQL/C components, indicating that the actual OpenLink Database Agent for Progress is for all intents and purposes a Progress ESQL/C client.

The files "proxxx_sv" or "proxx_sv.exe" are your OpenLink Database Agent for Progress executables (note: xxx represents Progress major and minor release numbers e.g. Progress 7.3C would be represented as pro73c_sv). These files are always situated in the "bin" sub-directory of your OpenLink installation directory.

When you attempt to make an ODBC/JDBC/UDBC connection to your Progress Database Engine(s), and you encounter a problem, one of the following error conditions would typically be presented to you via your ODBC/JDBC/UDBC application:

- "There is no server this database you are attempting to connect with", even though you know there is a server running for that database.

- "Shared memory of a particular version was expected, but shared memory of another version has been found"

The conditions above arise due to the fact that the OpenLink Database Agent for Progress not being built using Progress ESQL/C libraries that do not match your Progress Database Server to the exact minor release. This is to say that although the OpenLink Database Agent for Progress may have been built using ESQL/C libraries for Progress 7.3A, you will still have a version incompatibility problem of this nature should you attempt to connect this ESQL/C client to a Progress 7.3C Database Server process.

This error condition arises as a result of the fact that the OpenLink Database Agent for Progress defaults to using Shared Memory for its Inter Process Communications (IPC) , and that Shared Memory is very version sensitive nature albeit the fastest IPC mechanism .

There are two possible ways to work around this situation when it arises.

1. Change the default IPC mechanism of the OpenLink Database Agent for Progress, from Shared Memory to TCP sockets, as TCP sockets based IPC is version sensitive.
2. Rebuild/Relink your OpenLink Database Agent for Progress, achieved by running an OpenLink provided link script that produces a new OpenLink Database Agent for Progress executable with an exact Shared Memory match for your Progress Database Server.

10.6.4 Making OpenLink Database Agent Use a Sockets Based IPC

Pre-Configuration Check List:

- Progress Database Server installation must be licensed to use Progress Client Networking.
- TCP networking must be installed, configured and up and running on the Progress Database Server machine. (The command: ping localhost will quickly determine if TCP is configured correctly on your server machine).
- You need to create an entry in the "services" file (situated in the UNIX "/etc" directory) for one or more Progress TCP Sockets based Database Servers. The "services" file is standard TCP configuration file that is used to match Service Names to the TCP port number from which their services can be accessed. The entries in this file follow the format shown below:

  Table 8.19.

Service Name	Port/Protocol	Comment
netwall	533/udp	# -for emergency broadcasts
uucp	540/tcp	# uucp daemon
remotefs	556/tcp	# Brunhoff remote filesystem

Service Name	Port/Protocol	Comment
pppmsg	911/tcp	# PPP daemon
listen	1025/tcp	# listener RFS remote_file_sharing
nterm	1026/tcp	# remote_login network_terminal

You need to choose the name of a service and a port number that is not currently used by a tcp service (in the case illustrated above, for example, number 1026 is already being used).

Thus, if you chose to use the name *"pro8srv"* for your Progress TCP Sockets based Database Server s Service Name. And you chose to make its services available at TCP port # 9001 (and you have checked the file to make sure it is not being used) you would add the following entry (note that the third entry is simply a comment as it is preceded with a "#"):

```
pro8srv 9001/tcp # for Progress database sales
```

10.6.4.1 Configuration Steps

10.6.5 Note:

(Assuming a Progress Database named "sales.db" using as TCP Service "pro8srv")

- *Progress Server Components*
  - Assuming the database file "*sales.db"* is situated in directory "*/usr/data"*. Start your Progress TCP Sockets based Database Server process using the command:

    ```
    >proserve /usr/data/sales.db -S pro8srv  N TCP  H localhost
    ```
- *OpenLink Server Components (Database Agent for Progress)*
  - The OpenLink Session Rules Book (oplrqb.ini) allows you to control all OpenLink Data Access clients (ODBC/JDBC/UDBC) centrally, alleviating yourself from the time cost of maintaining numerous client based connection parameters.

    This is done by placing the Progress Database Connection Parameters `-S <service name> -N TCP  H localhost` as "ConnectOptions" section-key values within the Progress Database Agent configuration ("[generic_prox]") section of the Session Rules Book e.g.:

    ```
    [generic_pro8]
    Program = pro8_sv
    ServerOptions = -q  d mdy  TB 30  TM 30  e 25
    Database = /usr/data/sales.db
    ConnectOptions = -S pro8srv  N TCP  H localhost
    ```
- *OpenLink Client Components (Drivers for ODBC/JDBC /UDBC)*

  You only have to resort to the steps below if circumstances within your organization demand client side configuration, otherwise the server based approach using the OpenLink Session Rules Book (oprqb.ini) is much more effective and convenient.

  - *OpenLink ODBC Drivers* - Place the Progress Database Connection Parameters: `-S pro8srv  N TCP  H localhost` into the "Options" field of the OpenLink ODBC Data Source Configuration Utility
  - *OpenLink Drivers for JDBC* - Use the Progress Database Connection Parameters: `-S pro8srv  N TCP  H localhost` as "/OPTIONS" OpenLink JDBC URL attribute values e.g.:

    ```
    jdbc:openlink://dbservermachine/DSN=dsn_progress/OPTIONS=-S pro8srv  N TCP  H localh
    ```
  - *OpenLink UNIX based ODBC or UDBC Drivers* - Use the Progress Database Connection Parameters: `-S pro8srv  N TCP  H localhost` as "OPTIONS" connect string values e.g.:

    ```
    DSN=dsn_progress;OPTIONS=-S pro8srv  N TCP  H localhost
    ```

10.6.5.1 REBUILDING/RELINKING Your OpenLink Database Agent For Progress

As explained earlier an OpenLink Database Agent for Progress is a combination of OpenLink specific libraries (data access and network server modules) and Progress specific libraries provided situated in the "probuild" sub-directory under your Progress installations $DLC directory.

Should you desire the superior performance that Shared Memory has over TCP sockets, then rebuilding/relinking your OpenLink Database Agent for Progress provides an alternative solution to Shared Memory incompatibility problems that may be impeding your ability to establish an OpenLink session with your Progress Database Server.

Activity Prerequisites

- You must have a C library linker installed and running on your machine (part of your operating systems C compiler kit)
- You must have a "probuild" sub-directory situated under the $DLC (progress installation directory) directory.
- You must have a "lib" sub-directory situated under your OpenLink Server Components installation directory, and it must contain the following components:
    - A link script file named "mkdlcX" (where X represents your Progress installation s major release number e.g. for Progress 7.3C this would be lddlc7).
    - A link script named "ldproX" (where X represents your Progress installation s major release number e.g. for Progress 7.3C this would be ldpro7).
    - An OpenLink specific link library file named "pro7_sv.o"

Rebuild/Relink Steps

1. Create a UNIX environment variable named " *DLC"* and set its value your Progress installation directory. You do not need to do this as this environment variable is already set, the output from the command: echo $DLC will enable you determine if this has been set.
2. Create a UNIX environment variable named "*PROLOAD"* and set its value to *$DLC/probuild* From the UNIX command line prompt this is achieved by typing in the following:

   ```
   PROLOAD=$DLC/probuild ; export PROLOAD
   ```
3. Change directory to the lib sub-directory under your OpenLink Server Components installation directory
4. Run the script " *mkdlcX",* the aim here being the production of a file named *'dlcX.o'* (where X represents your Progress installations major release number)
5. If you receive an error messages from *step 4* stating that one or more Progress libraries cannot be found, please edit *mkdlcX* and remove any references to the file(s) identified in the error message.
6. Run the script ' *ldproX* ' . This will produce a new OpenLink Database Agent for Progress version X (proX_sv), and place the new executable file in the "bin" sub-directory of your OpenLink installation directory (the directory immediately adjacent to the "lib" directory from where you performed steps 1   5).

10.6.6 Distributed Databases, Array Fields, Database Triggers

Our ODBC Drivers facillate the support of the above through the use of a custom "Data Dictionary File", this file is built by running the OpenLink provided Progress Program File "*setup.p*" situated in the "bin" sub-directory below the OpenLink base installation directory.

Once the script has been run, modify the OpenLink "Session Rules Book" using the gudilines displayed when the " *setup.p*" program completes.

Note: To make use of Progress Database Triggers please ensure that the Logical Database name to which your Triggers have been compiled matches the value that you provide to the "-ld" parameter when setting your Progress Database Connect options during the execution of "setup.p" .

Example:

If you have compiled your triggers against a database physically identified as "/usr/progress/demo.db" but logically identified as "mydemo" your "setup.p" Database Connection String would be:

/usr/progress/demo -ld mydemo

10.6.7 Using OpenLink ODBC, Progress & Microsoft Access

The Mysterious "Admin" Account & Disabling the JET Scrollable Cursor (Dynaset) Facility

Please ensure that you have the following OpenLink Settings enabled :

Client

- OpenLink ODBC DSN "no login dialog box" check box is checked
- OPENLINK.INI settings `\Local Machine\Software\ODBC\OPENLINK.INI\Communications` ShowErrors Key is also set to "No" , this ensures that OpenLink Specific error messages aren't shown.

Server

- Ensure the following Mapping Rule exists as the first line of the [Mapping Rules] section of yor "Session Rules Book" :

```
[Mapping Rules]
;*:*:blank:*:*:*:rw = reject You should specify a username and password
*:*:Admin:*:*:*:rw = reject User Admin Account Does Not Exist In Your Progress Database
```

The above alleviates the need to depend of JetTryAuth or TryJetAuth.

10.6.8 Key Microsoft Access Jet Engines Setting That Can Affect Your OpenLink ODBC Experience

The following sections describe Windows registry settings settings for the Microsoft Jet database engine for connection to an ODBC database. These registry settings must be added by the user or by your application; the ODBC Driver Setup program does not write default values to the location:

```
\HKEY_LOCAL_MACHINE\Software\Microsoft\Access\7.0\Jet\3.5\Engines\ODBC
```

Initialization Settings for Jet-connected ODBC Databases, found at:

```
\HKEY_LOCAL_MACHINE\Software\Microsoft\Office\8.0\Access\Jet\3.5\Engines\ODBC
```

folder contains initialization settings for the Microsoft Jet database engine. Before you can add any of the following values to the registry, you must create the ODBC folder under the Jet\3.5\Engines folder.

To create the new folder, open the Jet\3.5\Engines folder in the Registry Editor and click Add Key on the Edit menu. In the Key Name: edit box, type ODBC and click OK. The Registry Editor creates a new folder below the Jet\3.5\Engines folder.

10.6.9 Note:

If you are using Windows NT, the Add Key dialog box also contains an edit box for the registry key class; you can leave this setting blank.

Typical settings for the entries in the Jet\3.5\Engines\ODBC folder are shown in the following example.

```
LoginTimeout=20
QueryTimeout=60
ConnectionTimeout=600
AsyncRetryInterval=500
AttachCaseSensitive=0
AttachableObjects=&#39;TABLE&#39;,&#39;VIEW&#39;,&#39;SYSTEM TABLE&#39;,&#39;ALIAS&#39;,&#39;SYNON
SnapshotOnly=0
TraceSQLMode=0
TraceODBCAPI=0
DisableAsync=0
JetTryAuth=1
PreparedInsert=0
PreparedUpdate=0
FastRequery=0
```

The Jet database engine uses the ODBC entries as follows:

LoginTimeout. The number of seconds a login attempt can continue before timing out. The default is 20 (values are of type REG_DWORD).

QueryTimeout. The number of seconds a query can run (total processing time) before timing out. The default is 60 (values are of type REG_DWORD).

ConnectionTimeout. The number of seconds a cached connection can remain idle before timing out. The default is 600 (values are of type REG_DWORD).

AsyncRetryInterval. The number of milliseconds between polls to determine if the server is done processing a query. This entry is used for asynchronous processing only. The default is 500 (values are of type REG_DWORD).

AttachCaseSensitive. An indicator of whether to match table names exactly when linking. Values are 0 (link the first table matching the specified name, regardless of case) and 1 (link a table only if the name matches exactly). The default is 0 (values are of type REG_DWORD).

AttachableObjects. A list of server object types to which linking will be allowed. The default is: 'TABLE', 'VIEW', 'SYSTEM TABLE', 'ALIAS', 'SYNONYM' (values are of type REG_SZ).

SnapshotOnly. An indicator of whether Recordset objects are forced to be of snapshot type. Values are 0 (allow dynasets) and 1 (force snapshots only). The default is 0 (values are of type REG_DWORD).

TraceSQLMode. An indicator of whether the Jet database engine will trace SQL statements sent to an ODBC data source in SQLOUT.txt. Values are 0 (no) and 1 (yes). The default is 0 (values are of type REG_DWORD). This entry is interchangeable with SQLTraceMode.

TraceODBCAPI . An indicator of whether to trace ODBC API calls in ODBCAPI.txt. Values are 0 (no) and 1 (yes). The default is 0 (values are of type REG_DWORD).

DisableAsync. An indicator of whether to force synchronous query execution. Values are 0 (use asynchronous query execution if possible) and 1 (force synchronous query execution). The default is 1 (values are of type REG_DWORD).

JetTryAuth. An indicator of whether to try using the Microsoft Access user name and password to log in to the server before prompting. Values are 0 (no) and 1 (yes). The default is 1 (values are of type REG_DWORD).

PreparedInsert. An indicator of whether to use a prepared INSERT statement that inserts data in all columns. Values are 0 (use a custom INSERT statement that inserts only non-Null values) and 1 (use a prepared INSERT statement). The default is 0 (values are of type REG_DWORD).Using prepared INSERT statements can cause Nulls to overwrite server defaults and can cause triggers to execute on columns that weren't inserted explicitly.

PreparedUpdate. An indicator of whether to use a prepared UPDATE statement that updates data in all columns. Values are 0 (use a custom UPDATE statement that sets only columns that have changed) and 1 (use a prepared UPDATE statement). The default is 0 (values are of type REG_DWORD).Using prepared UPDATE statements can cause triggers to execute on unchanged columns.

FastRequery. An indicator of whether to use a prepared SELECT statement for parameterized queries. Values are 0 (no) and 1 (yes). The default is 0 (values are of type REG_DWORD).

10.7 Application Server Architecture

Figure 8.44. Application Server Conceptual Architecture - Three Tier Distributed

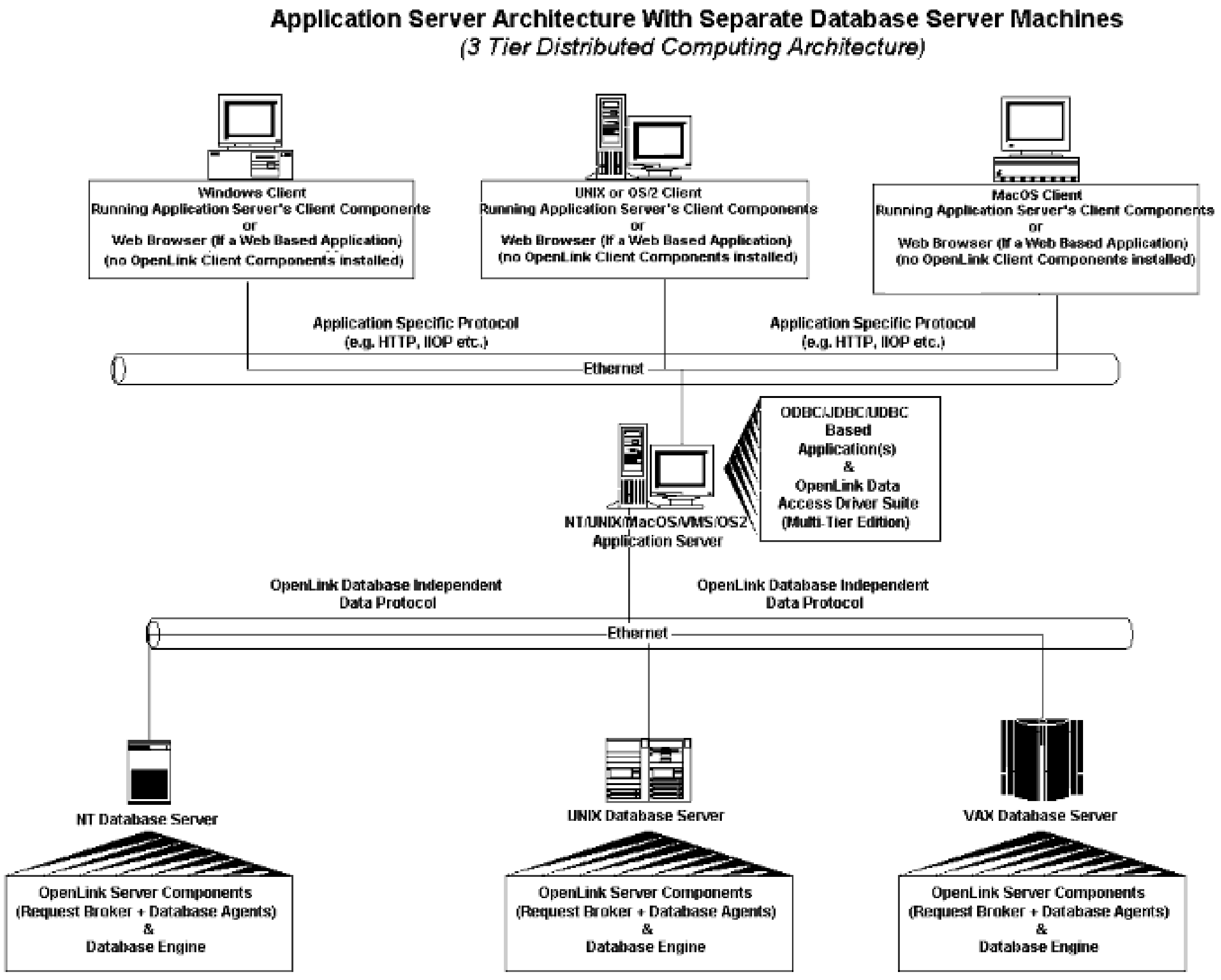

Application Server Architecture With Separate Database Server Machines
(3 Tier Distributed Computing Architecture)
Windows Client
Running Application Server's Client Components
or
Web Browser (If a Web Based Application)
(no OpenLink Client Components installed)
UNIX or OS/2 Client
Running Application Server's Client Components
or
Web Browser (If a Web Based Application)
(no OpenLink Client Components installed)
MacOS Client
Running Application Server's Client Components
or
Web Browser (If a Web Based Application)
(no OpenLink Client Components installed)
Application Specific Protocol
(e.g. HTTP, IIOP etc.)
Application Specific Protocol
(e.g. HTTP, IIOP etc.)
Ethernet
ODBC/JDBC/UDBC
Based
Application(s)
&
OpenLink Data
Access Driver Suite
(Multi-Tier Edition)
NT/UNIX/MacOS/VMS/OS2
Application Server
OpenLink Database Independent
Data Protocol
OpenLink Database Independent
Data Protocol
Ethernet
NT Database Server
UNIX Database Server
VAX Database Server
OpenLink Server Components
(Request Broker + Database Agents)
&
Database Engine
OpenLink Server Components
(Request Broker + Database Agents)
&
Database Engine
OpenLink Server Components
(Request Broker + Database Agents)
&
Database Engine

Figure 8.45. Application Server Conceptual Architecture - Avoiding DB Server Install

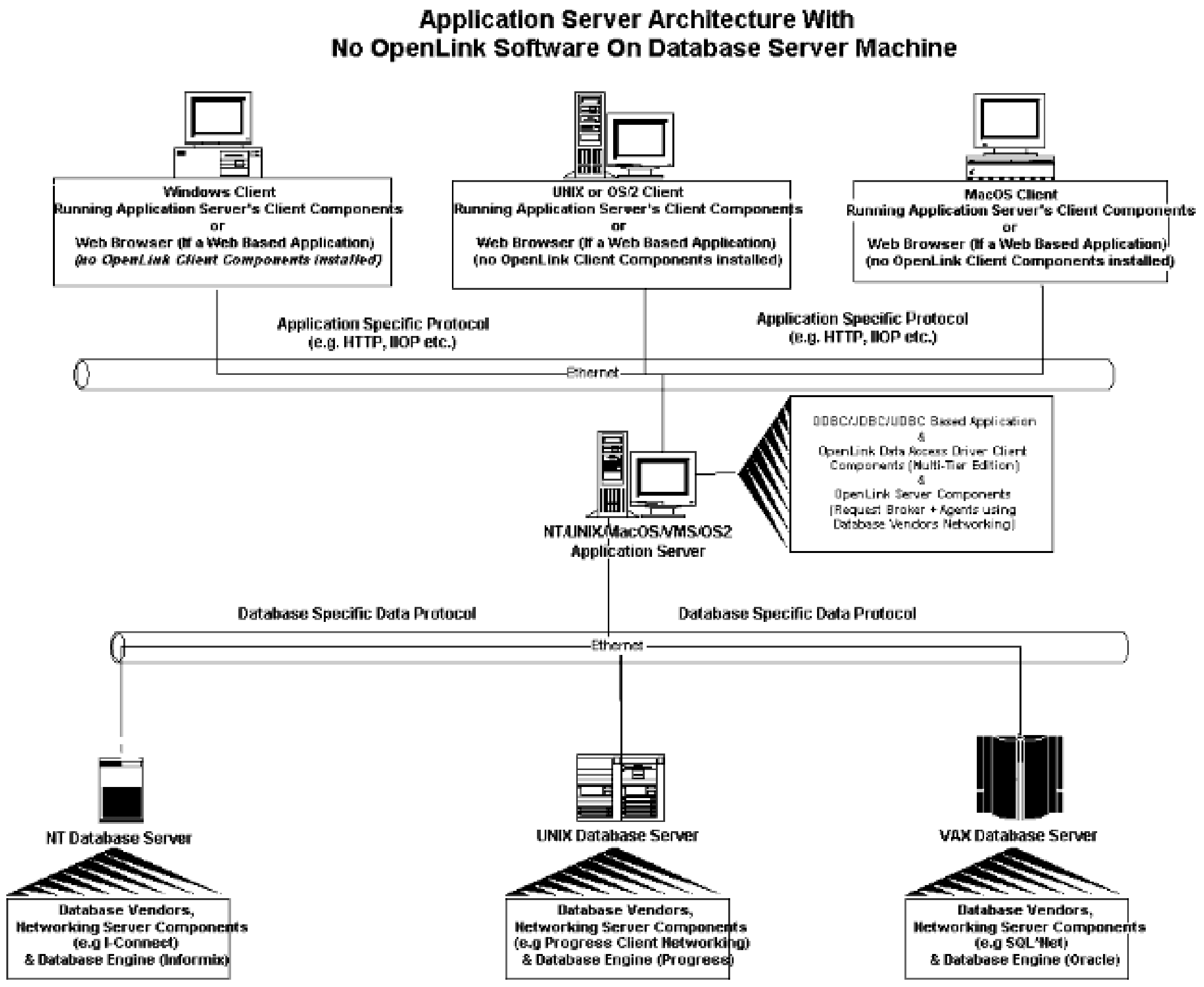
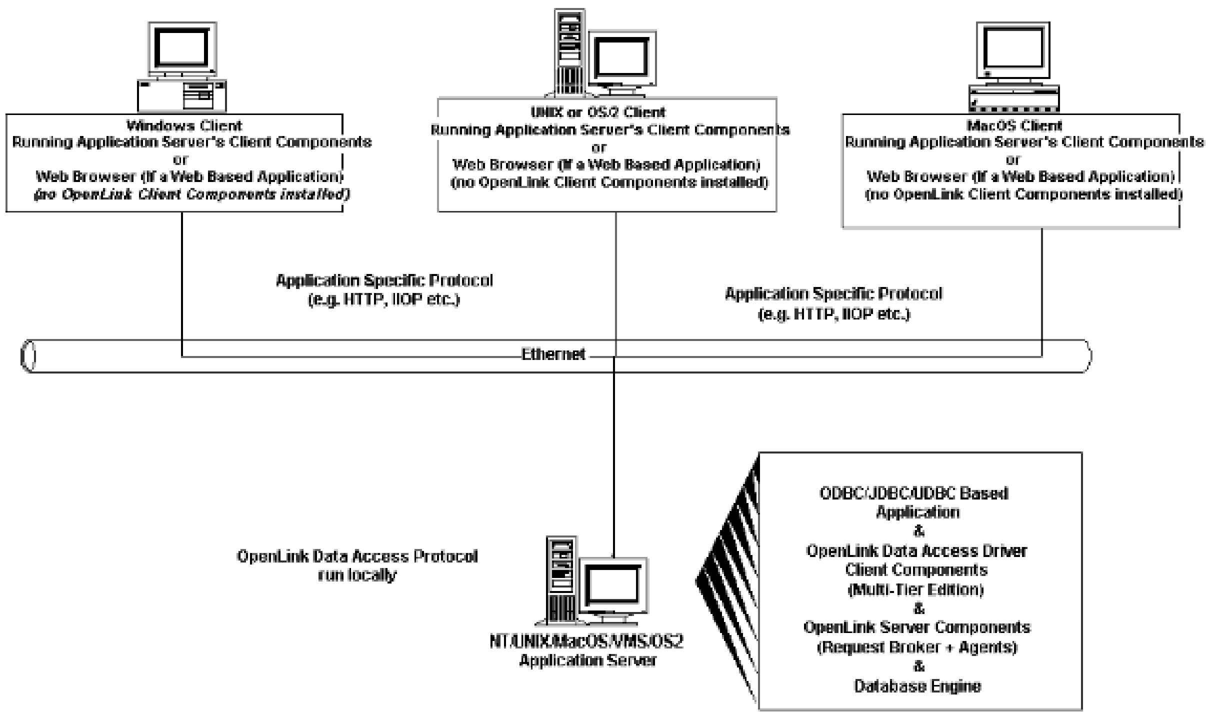

Application Server Architecture With
No OpenLink Software On Database Server Machine
Windows Client
Running Application Server's Client Components
or
Web Browser (If a Web Based Application)
(no OpenLink Client Components installed)
UNIX or OS/2 Client
Running Application Server's Client Components
or
Web Browser (If a Web Based Application)
(no OpenLink Client Components installed)
MacOS Client
Running Application Server's Client Components
or
Web Browser (If a Web Based Application)
(no OpenLink Client Components installed)
Application Specific Protocol
(e.g. HTTP, IIOP etc.)
Application Specific Protocol
(e.g. HTTP, IIOP etc.)
Ethernet
ODBC/JDBC/UDBC Based Application
&
OpenLink Data Access Driver Client Components (Multi-Tier Edition)
&
OpenLink Server Components
(Request Broker + Agents using Database Vendors Networking)
NT/UNIX/MacOS/VMS/OS2
Application Server
Database Specific Data Protocol
Database Specific Data Protocol
Ethernet
NT Database Server
UNIX Database Server
VAX Database Server
Database Vendors,
Networking Server Components
(e.g I-Connect)
& Database Engine (Informix)
Database Vendors,
Networking Server Components
(e.g Progress Client Networking)
& Database Engine (Progress)
Database Vendors,
Networking Server Components
(e.g SQL*Net)
& Database Engine (Oracle)

Figure 8.46. Application Server Conceptual Architecture - Same Machine

Application Server Architecture
Including Database Server On Same Machine


Windows Client
Running Application Server's Client Components
or
Web Browser (If a Web Based Application)
(no OpenLink Client Components installed)
UNIX or OS/2 Client
Running Application Server's Client Components
or
Web Browser (If a Web Based Application)
(no OpenLink Client Components installed)
MacOS Client
Running Application Server's Client Components
or
Web Browser (If a Web Based Application)
(no OpenLink Client Components installed)
Application Specific Protocol
(e.g. HTTP, IIOP etc.)
Application Specific Protocol
(e.g. HTTP, IIOP etc.)
Ethernet
OpenLink Data Access Protocol
run locally
NT/UNIX/MacOS/VMS/OS2
Application Server
ODBC/JDBC/UDBC Based
Application
&
OpenLink Data Access Driver
Client Components
(Multi-Tier Edition)
&
OpenLink Server Components
(Request Broker + Agents)
&
Database Engine

11 Chapter 9. Securing a Multi-Tier connection using SSL

Abstract
This document provides information on configuring OpenLink Multi-Tier for use with SSL, both on the client and server sides. SSL, implemented with OpenSSL, facilitates strong cryptographic encryption the whole network communication layer.

Table of Contents

11.1 Overview

As of version 5.2, OpenLink UDA now comes with optional support for using OpenSSL to encrypt the data layer between generic client and request broker / agent.

Features of the SSL implementation:

secures connections to both the broker and all database agents

encryption is optional, although it can be made mandatory

the same connection endpoint endpoint (port) handles both encrypted and non-encrypted clients

fully backward-compatible

per-agent certificates

all changes are in the OpenLink communications layer: there is no need to change any of the database agents to cater for SSL.

11.2 Implementing SSL Encryption

11.2.1 Server-side Configuration for SSL

In order to make use of this functionality, you need to create a certificate on the server running the request broker. On Unix platforms we have provided a bin/mkcert.sh script as part of the installation. On other platforms you may have to run openssl by hand or use another system to generate the PEM file (such as generating it on a unix server and copying the files across).

A sample run of the mkcert.sh script, showing the various questions it asks to identify your organization, follows:

```
bash$ cd /usr/openlink/bin/
bash$ ./mkcert.sh mykey
Checking OpenSSL: OpenSSL 0.9.7d 17 Mar 2004
Generating a 512 bit RSA private key
.++++++++++++
.......++++++++++++
writing new private key to 'mykey.key'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Host name (externally visible) [host.example.com]:purple
Organization (eg. company) [OpenLink Software]:
Organizational Unit [Research & Development]:Product Development
State or Province Name (full name) []:Surrey
Locality Name (eg. city) []:Croydon
Country Name (2 letter code) [US]:UK
1024 semi-random bytes loaded
```

```
Generating DH parameters, 512 bit long safe prime, generator 2
This is going to take a long time
..+.......................+.........................+.......+...........
[]
.+.....++*++*++*++*++*++*
subject= /CN=purple/O=OpenLink Software/OU=Product Development/ST=Surrey/L=Croydon/C=UK
notBefore=Nov 12 10:43:14 2004 GMT
notAfter=Nov 10 10:43:14 2014 GMT
MD5 Fingerprint=E0:DB:53:E7:E7:68:21:53:1C:2B:1E:8E:D9:BF:25:F0
Created private key parameters in mykey.key
Created public key parameters in mykey.cer
bash$
```

Having created the key-pair, you now need to tell the request broker to use them. This requires the following lines to be present in the rulebook (oplrqb.ini):

```
[Request Broker]
....
SSLKeyFile = mykey.pem
SSLRequired = No
```

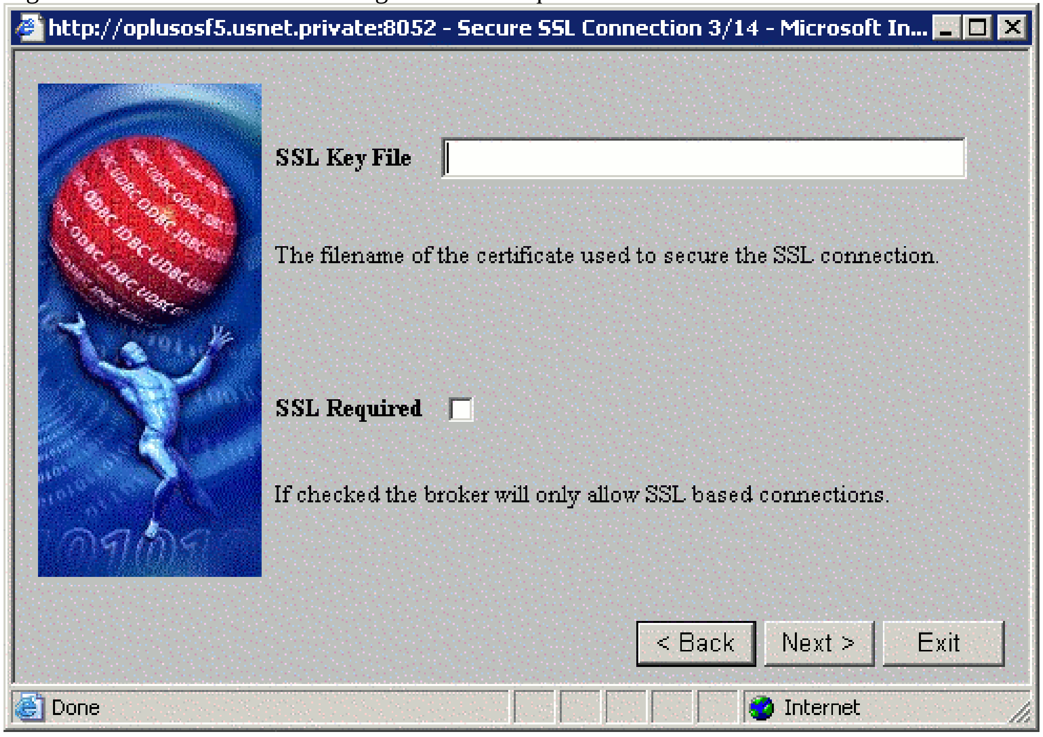
The SSLKeyFile is the filename of the key you just created; to deny connections without SSL, set SSLRequired=Yes instead.

You can also put these options in the individual database agent sections, such that one key is used for e.g. Ingres, and another for SQL Server, or one for your sales database and another for people coming in from VPN IP#s (by editing the mapping rules to match on a per-IP# basis). For example:

```
[generic_sqlserver]
.....
SSLKeyFile = mykey-sqlserver.pem
SSLRequired = Yes                  ; All SQLServer connections
MUST use SSL
```

As an alternative to editing oplrqb.ini by hand, you can use the web-based Administrator GUI, either the form or wizards, to set the SSL Key File and SSL Required fields, thus:

Figure 9.1. Server-side SSL Configuration of Request Broker via Web Assistant


http://oplusosf5.usnet.private:8052 - Secure SSL Connection 3/14 - Microsoft In...

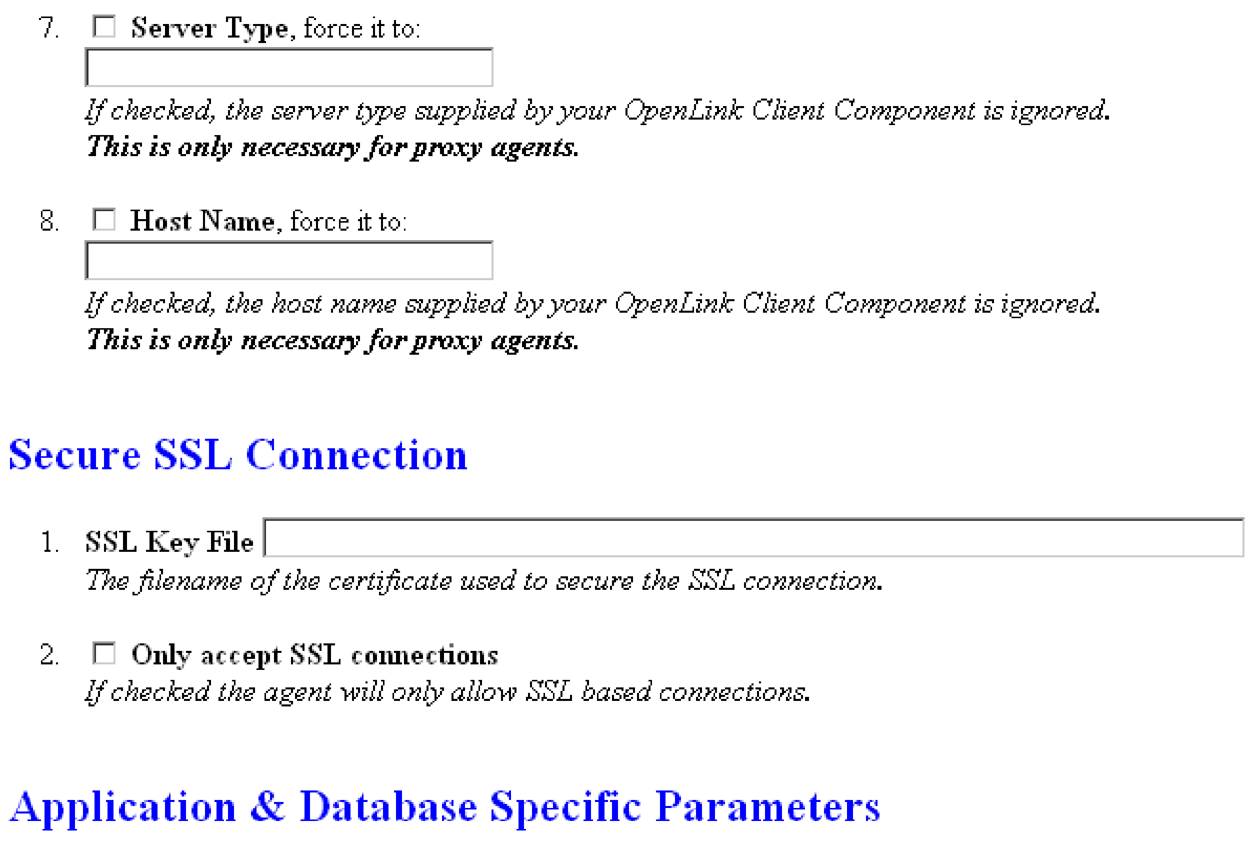
SSL Key File

The filename of the certificate used to secure the SSL connection.

SSL Required ☐

If checked the broker will only allow SSL based connections.

Figure 9.2. Server-side SSL Configuration at the database agent level via Web Assistant Form



7. ☐ **Server Type**, force it to:

   If checked, the server type supplied by your OpenLink Client Component is ignored.
   This is only necessary for proxy agents.

8. ☐ **Host Name**, force it to:

   If checked, the host name supplied by your OpenLink Client Component is ignored.
   This is only necessary for proxy agents.

Secure SSL Connection

1. **SSL Key File**
   The filename of the certificate used to secure the SSL connection.

2. ☐ **Only accept SSL connections**
   If checked the agent will only allow SSL based connections.

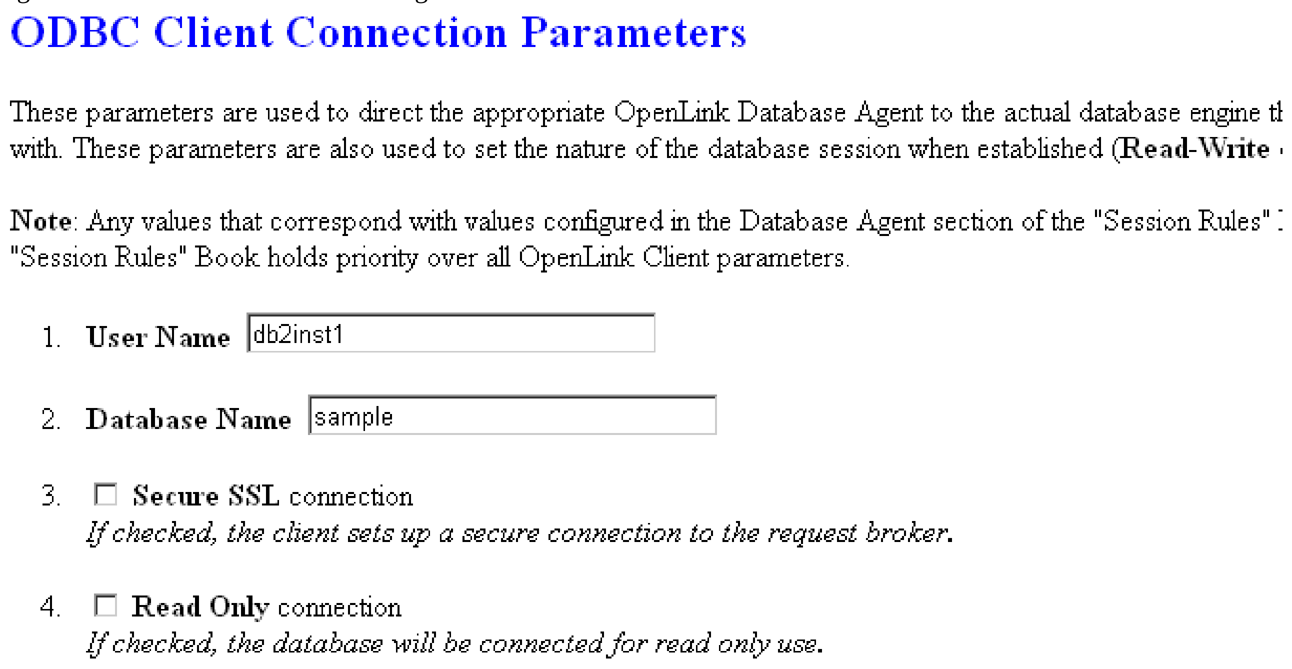
Application & Database Specific Parameters

11.2.2 Client-side Configuration for SSL

11.2.2.1 SSL Configuration for Windows Clients

On the client, when you configure a new DSN there is now a checkbox for "secure connection" adjacent to the username and password. Checking this will invoke use of SSL on the server for connections going through this DSN. This applies also to the web-based DSN configuration utility.

Figure 9.3. Client-side SSL DSN Configuration via Web Assistant Form



ODBC Client Connection Parameters

These parameters are used to direct the appropriate OpenLink Database Agent to the actual database engine tl
with. These parameters are also used to set the nature of the database session when established (**Read-Write**

Note: Any values that correspond with values configured in the Database Agent section of the "Session Rules"
"Session Rules" Book holds priority over all OpenLink Client parameters.

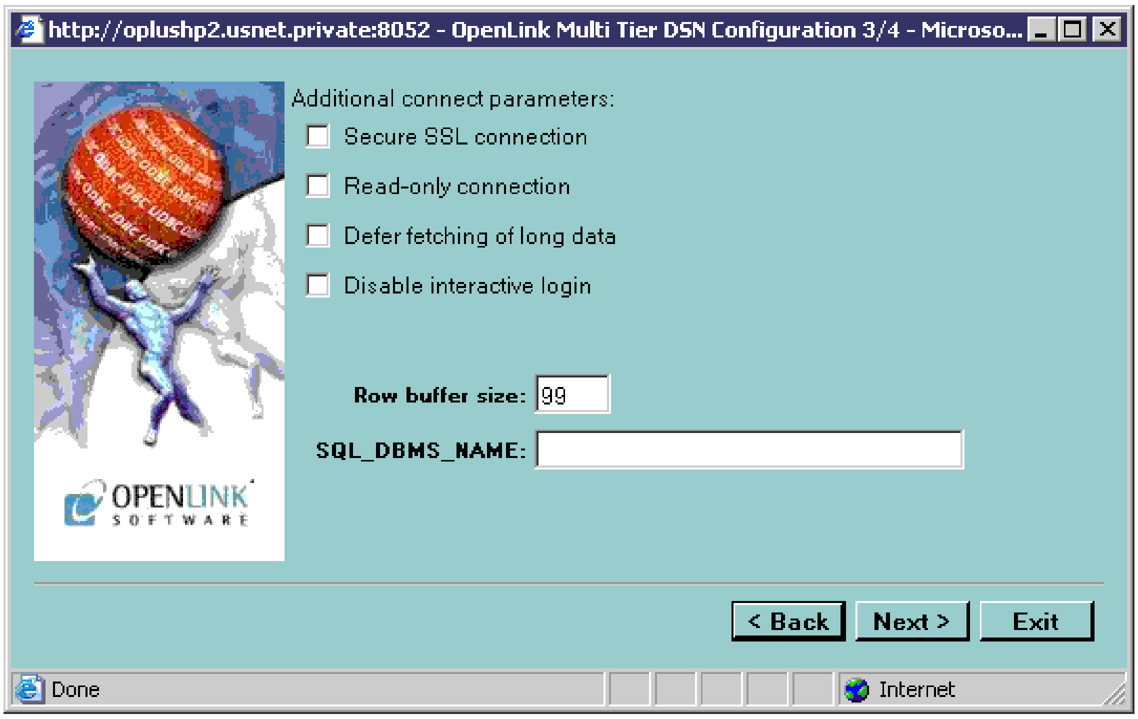
1. **User Name** db2inst1

2. **Database Name** sample

3. ☐ **Secure SSL** connection
   If checked, the client sets up a secure connection to the request broker.

4. ☐ **Read Only** connection
   If checked, the database will be connected for read only use.

Figure 9.4. Client-side SSL DSN Configuration via Web Assistant Form

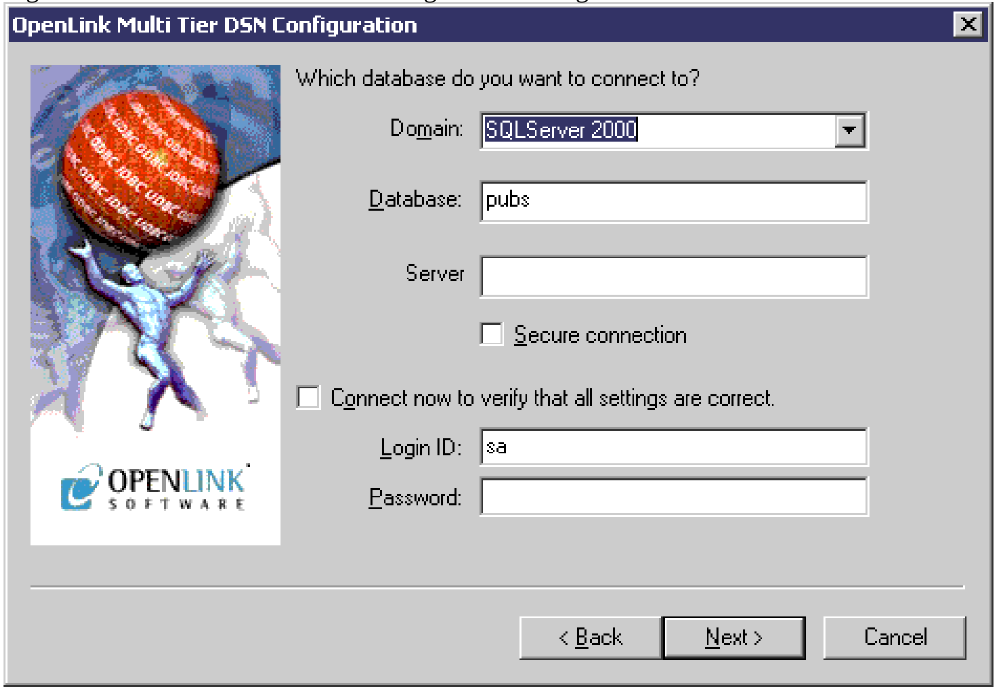

http://oplushp2.usnet.private:8052 - OpenLink Multi Tier DSN Configuration 3/4 - Microso...
OPENLINK SOFTWARE
< Back
Next >
Exit
Done
Internet

Additional connect parameters:

- [ ] Secure SSL connection
- [ ] Read-only connection
- [ ] Defer fetching of long data
- [ ] Disable interactive login

Row buffer size:	99
SQL_DBMS_NAME:	

Figure 9.5. Client-side SSL DSN Configuration through native Windows ODBC Administrator interface



Which database do you want to connect to?

Domain:	SQLServer 2000
Database:	pubs
Server	

- [ ] Secure connection
- [ ] Connect now to verify that all settings are correct.

Login ID:	sa
Password:	

11.2.2.2 SSL with JDBC Clients

The Java JDBC classes make use of J2SSE from Sun: as such, SSL is not supported with users of JDK1.1; it requires a separate download from Sun for JDK1.2 and 1.3; and these classes come as standard with JDK 1.4 and 1.5.

An SSL connection can be made by specifying the /UseSSL attribute in your JDBC connectstring, thus:

```
jdbc:openlink://localhost:5001/SVT=SQLServer/Database=pubs/UseSSL/UID=sa/PWD=
java -classpath .;.\opljdbc3.jar -Djavax.net.ssl.trustStore=<KeystoreFile> sample
```

where 'sample' is the name the JDBC Program. Note the use of the keystore file which is a unique feature of the JSSE implmentation, being the location where the certificate information is stored.

Should the JSSE classes not be located the JDBC client will return an error of the form:

```
"You need to add JSSE to your CLASSPATH in order to make a SSL connection
java.sql.SQLException: [OpenLink][OPLJDBC2]Connection failed:
java.io.IOException: java.lang.NoClassDefFoundError:javax/net/ssl/SSLSocket"
```

11.2.3 Error Messages

The introduction of SSL brings two new possible error messages to be seen at connect-time:

IM600 (ER_SSL_NOT_IN_CLIENT) "SSL initialization failure": this indicates that the client either has no SSL support built in, or that the underlying SSL library failed to initialize properly.

IM601 (ER_SSL_NOT_IN_SERVER) "The server does not support SSL": this indicates that the server has no SSL support built in, or that the SSL library failed to initialize properly. It could also mean that the required certificate could not be loaded on the server.

12 Chapter 10. Zero Configuration Networking

Abstract
Zero Configuration Networking is a server administered service that broadcasts configured available services over IP to allow clients to discover and used them with no prior knowledge or client specific configuration.

Table of Contents

12.1 Introduction

What is it? Zero Configuration Networking, also known as ZeroConf, is networking that needs nothing to be pre-configured and no administration to operate. Zero Configuration Networking uses industry standard IP protocols to allow devices to automatically find each other without the need to enter IP addresses or configure DNS servers.

Why is it Important? ZeroConf enables services on a network built with this support to be automatically discovered, thereby requiring no knowledge on the part of the end user attempting to use this service about the specifics of its configuration, just that it is available for use if selected.

Using ZeroConfig for installation rollouts or for site maintenance can significantly reduce administration overhead. Because the client ODBC Driver is bound to a service name only, should the details of that service change the client will automatically re-acquire the new details (this does not include changes to the name of the service). For example, 200 client installations configured to use the service "SalesDB1", moving the "Sales" database to another server, perhaps even a different database does not required any further client configuration, they will automatically connect to the new server.

ZeroConf configuration on the server is kept to a minimum. The host and port number of the Request Broker are automatically supplied to the ZeroConf service descriptions on the server when they are broadcast over the network. Furthermore, ZeroConf service descriptions can be based directly on a Server Type that maps to a Database Agent. If we consider the above example again, moving the "Sales" database to another machine will not need any ZeroConf adjustment on the server either, the new host and port and automatically discovered and broadcast with no administrator intervention other than to start the new Broker and Database.

12.2 Server Configuration

The OpenLink Multi-Tier Request Broker now has the ability to Broadcast ZeroConf services of available Database agent connections on the network. These can be discovered by the the OpenLink Multi-Tier ODBC client.

12.2.1 Configuration by Rule Book Edit

ZeroConf Support in the Request Broker is configured via the Rule Book. The Rule Book now contains a new section called [Zero Config]. This section contains a list of DSN="Connect String" pairs. The DSN is the name that will be presented to clients listing available ZeroConf data source, the Connect String is a string of connection attributes necessary for making a connection to a Database Agent as the example below shows:

```
[Zero Config]
ZJDBC3= "ServerType=JDBC13;Database=virtuoso.jdbc2.Driver;UID=dba"
ZSQL2K = "ServerType=SQLServer 2000;Database=pubs;UID=sa"
SQLServer 2000="Database=pubs;UID=sa"
```

The syntax is:

```
<Service_name>="Connect String"
```

or:

```
<Server Type>="Connect String"
```

In this second case, there is no Server Type defined within the connect string itself.

- <Service_name>. This is the service name you want this specific ZeroConf service to be known as when discovered by any Multi-Tier ZeroConf clients on the network.
- Connect String. A semi-colon (;) separated list of connection attributes that the Request Broker will supply to clients for automatically connecting the the requested DSN. Note that these attributes can be any of those used by a normal Multi-Tier client, since this is effectively making DSN-less connection string for clients to use. Host and Port cannot be specified, these are automatically supplied by the broker when the service is requested.

12.2.2 Note:

The Host and Port number of the Request Broker are not specifiable, the Broker will supply these to the services broadcasted automatically. This brings a huge saving on administration overhead; if the circumstances of the networking infrastructure change, the installation is simply moved to another machine or a backup server activated, no modifications to the ZeroConf details are required, and the new host:port will be automatically supplied from then on.

The Connect String must be enclosed in double-quotes.

The list of available attributes are:

- ServerType [SVT]. Specify agent domain. This is used by the broker to determine which agent section to connect the client request to using mapping rules.
- ServerOptions. Server startup options specific to agent/database.
- Database [DATABASE]. Database to use. Agent/database specific.
- Options [OPTIONS]. Database connect options. Agent/database specific.
- UserName [UID]. Username to connect as.
- Password [PWD]. Password for user.
- ReadOnly [READONLY]. Specify Y(es) for read-only (ro) or N(o) for read-write (rw) connections. Read only connections are sometimes faster, but can never modify any database.
- FetchBufferSize [FBS]. Number of rows (records) to be fetched per call from database agent. Values range from 1 to 99
- Persist. Controls whether the Zero Config DSN should be persisted on the client when select. If set to Yes/Y/T then it will, and if for set to No/N/F then it will only be used at connect time and not stored in the client which may be deemed a securty breach.

12.2.3 Configuration by Wizard

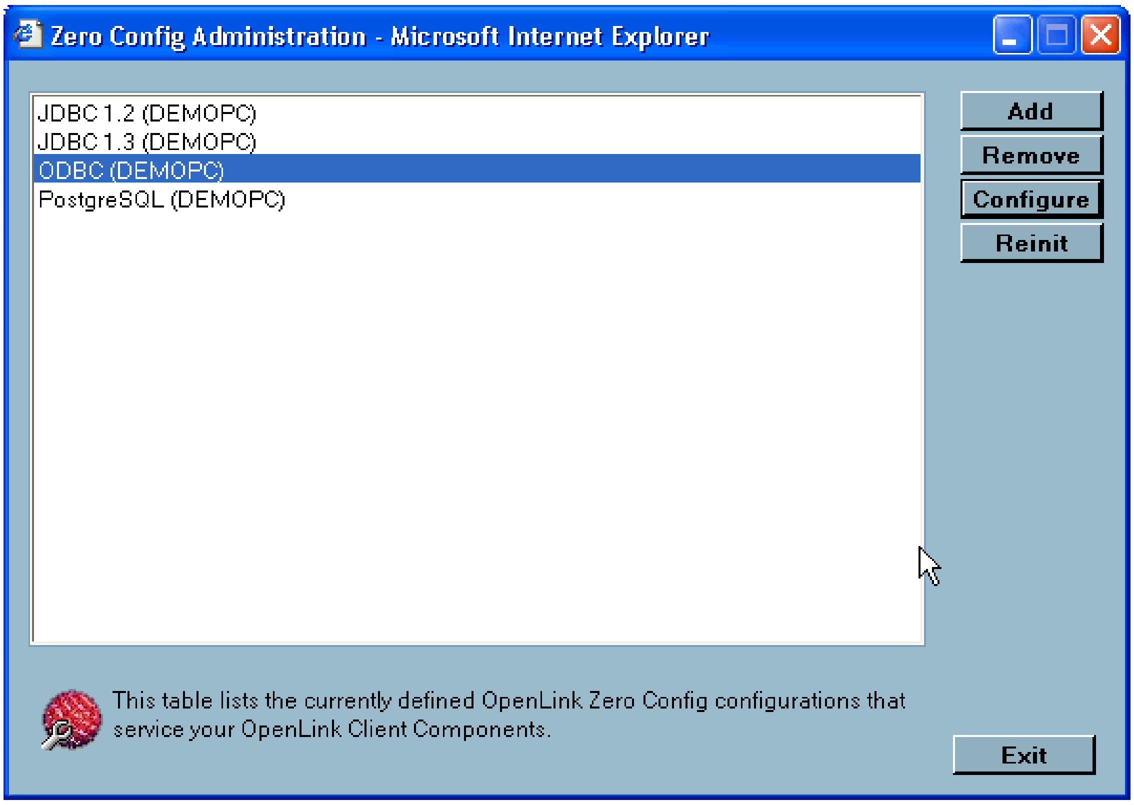
The OpenLink Universal Data Access Admin Assistant provides a graphical remote access interface to the server settings such as those found in the Rule Book. This interface has been extended to support configuration of the ZeroConf settings in the Server Component Administration Section, called Zero Config Administration.

The Zero Config Administration pages are very much like Client Data Source configuration pages, they have the same layout and descriptions as they are essentially data sources stored on the server.

To make a new Zero Configuration entry, follow this sequence:

1. From the Server Components Administration menu, chose the Zero Config Administration sub menu, and then Zero Config Administration by Wizard. A list of current confirguations is shown:

   Figure 10.1. Zero Config by Wizard - Admin Assistant Configuration

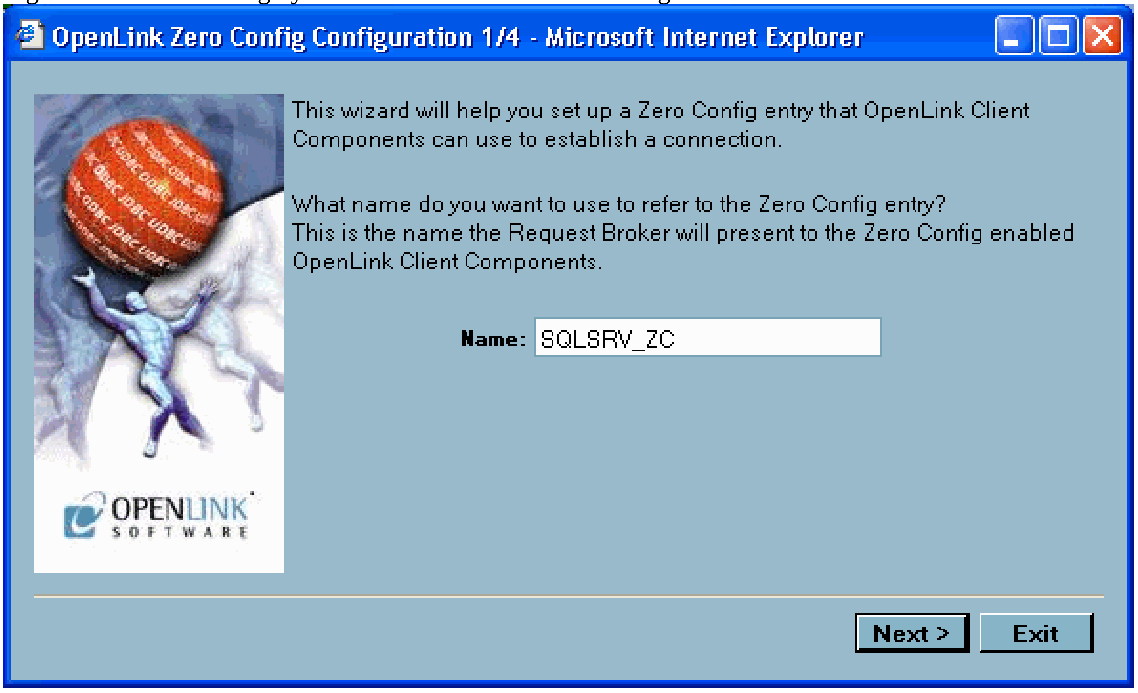


JDBC 1.2 (DEMOPC)
JDBC 1.3 (DEMOPC)
ODBC (DEMOPC)
PostgreSQL (DEMOPC)

This table lists the currently defined OpenLink Zero Config configurations that service your OpenLink Client Components.

2. Press the Add button. Enter a suitable name for the configuration:

Figure 10.2. Zero Config by Wizard - Admin Assistant Configuration


OpenLink Zero Config Configuration 1/4 - Microsoft Internet Explorer
OPENLINK SOFTWARE
Next >
Exit

This wizard will help you set up a Zero Config entry that OpenLink Client Components can use to establish a connection.

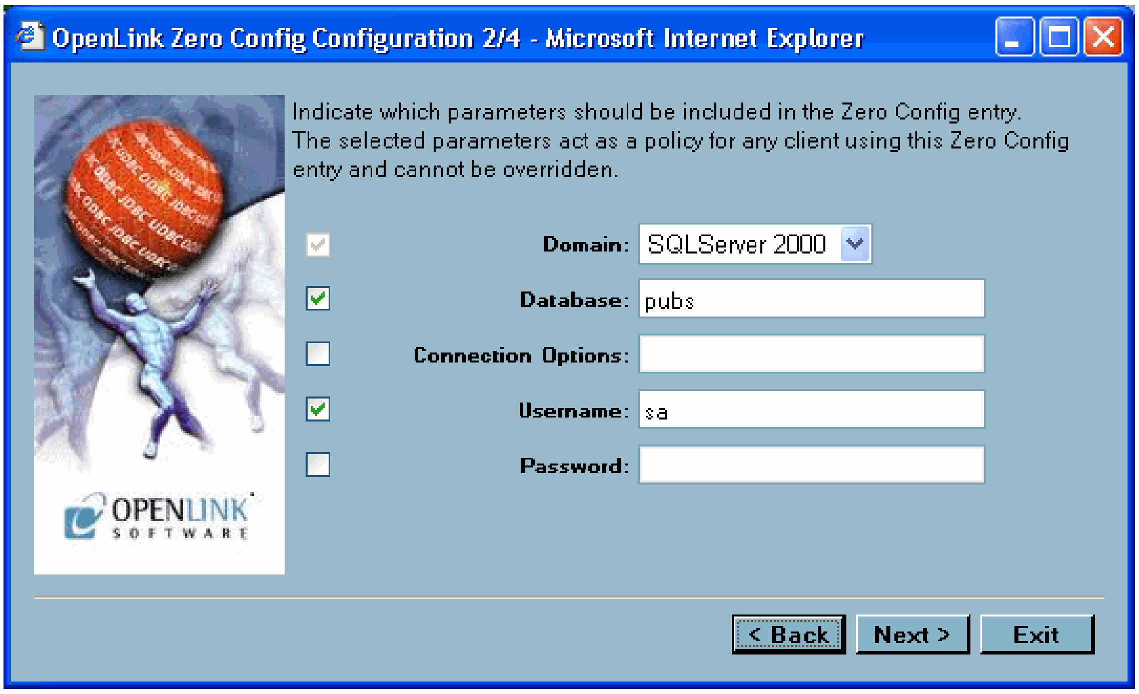
What name do you want to use to refer to the Zero Config entry?
This is the name the Request Broker will present to the Zero Config enabled OpenLink Client Components.

Name: SQLSRV_ZC

3. Enter the database details. The parameters are identical to those in the ODBC DSN configuration wizard.

   For each parameter you wish to configure, you need to enable it by checking the box on the left side. Each field is disabled until the check box is marked.

   Figure 10.3. Zero Config by Wizard - Admin Assistant Configuration

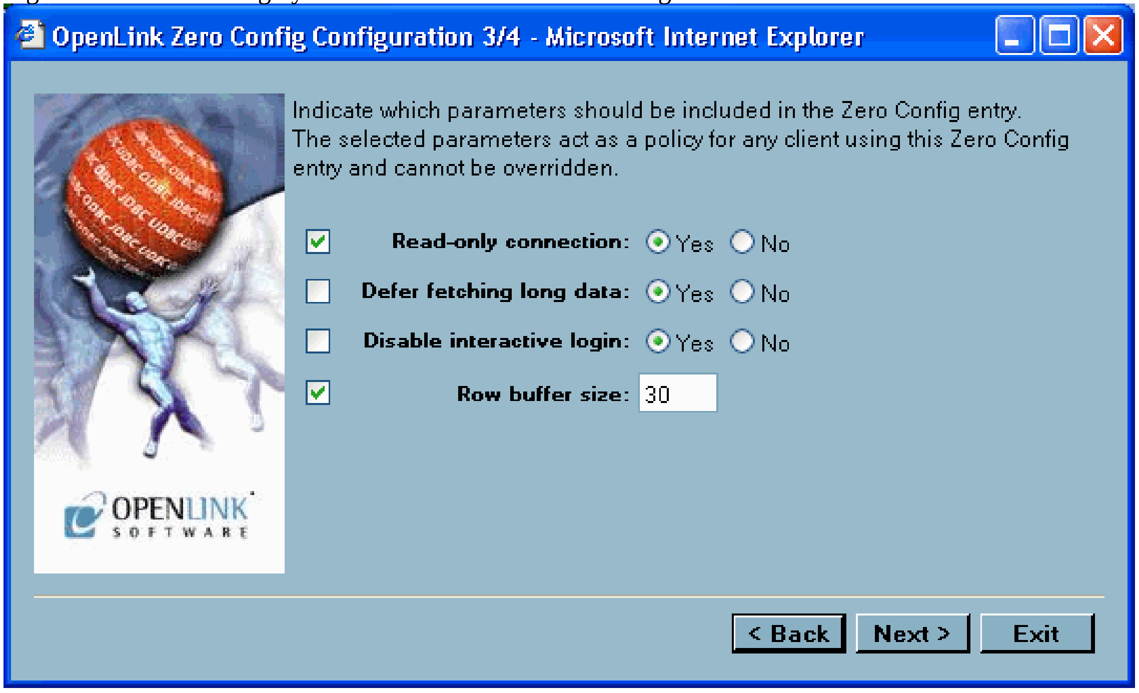

OpenLink Zero Config Configuration 2/4 - Microsoft Internet Explorer

Indicate which parameters should be included in the Zero Config entry.
The selected parameters act as a policy for any client using this Zero Config entry and cannot be overridden.

✓	Domain:	SQLServer 2000
✓	Database:	pubs
	Connection Options:	
✓	Username:	sa
	Password:	

4. Enter the additional connection parameters. In same way as previous page, each parameter field needs to be enabled by the check mark on the left side.

Figure 10.4. Zero Config by Wizard - Admin Assistant Configuration

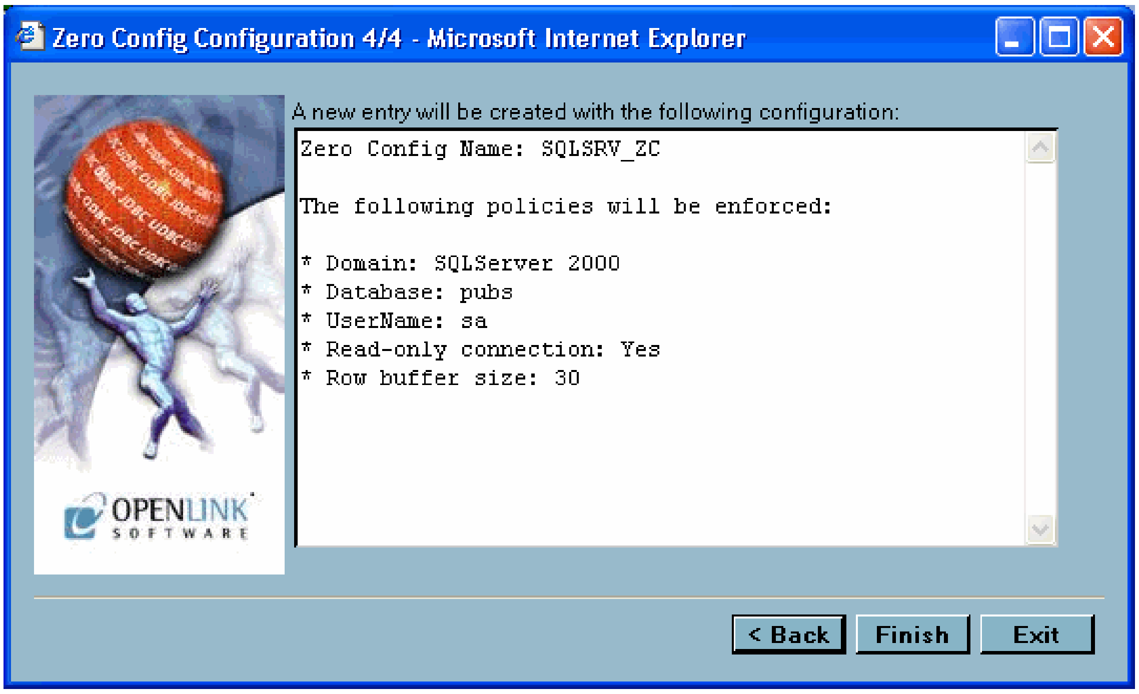

OpenLink Zero Config Configuration 3/4 - Microsoft Internet Explorer

Indicate which parameters should be included in the Zero Config entry.
The selected parameters act as a policy for any client using this Zero Config entry and cannot be overridden.

✓	Read-only connection:	[✓ Yes] No
	Defer fetching long data:	[✓ Yes] No
	Disable interactive login:	[✓ Yes] No
✓	Row buffer size:	30

5. The complete configuration parameters are now shown:

Figure 10.5. Zero Config by Wizard - Admin Assistant Configuration

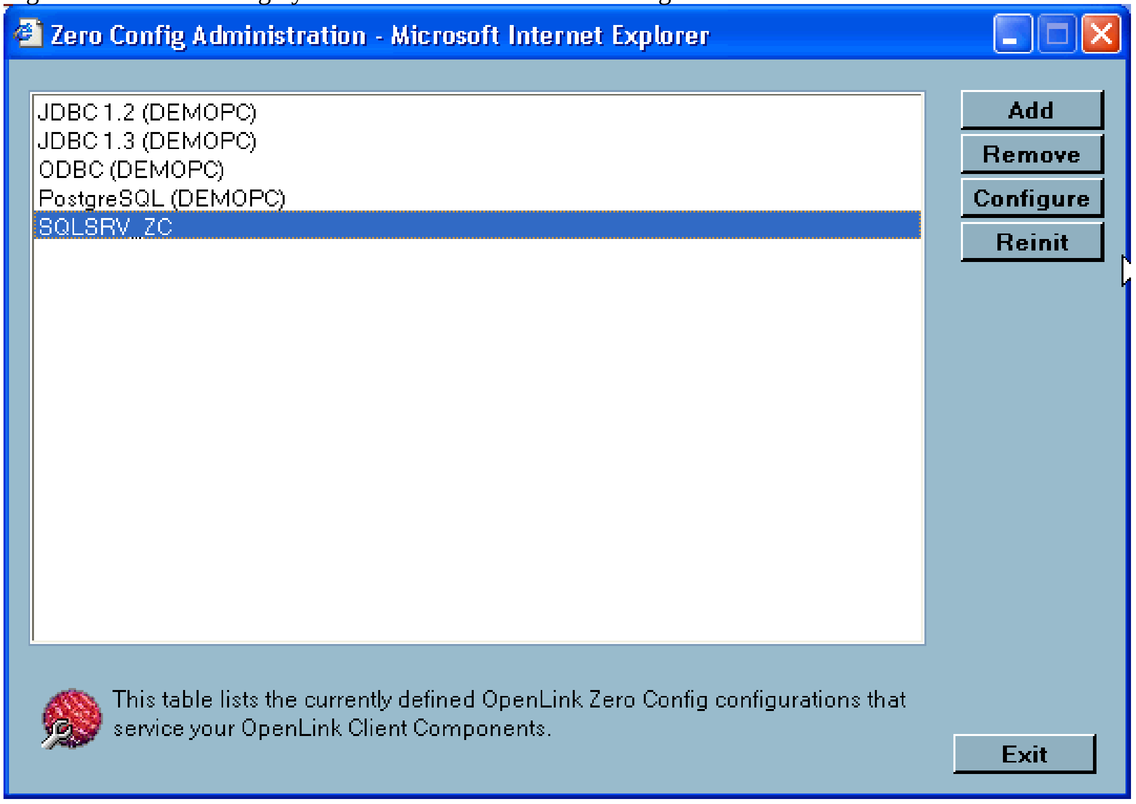


A new entry will be created with the following configuration:

```
Zero Config Name: SQLSRV_ZC

The following policies will be enforced:

* Domain: SQLServer 2000
* Database: pubs
* UserName: sa
* Read-only connection: Yes
* Row buffer size: 30
```

6. The Zero Config entry now appears in the list:

Figure 10.6. Zero Config by Wizard - Admin Assistant Configuration



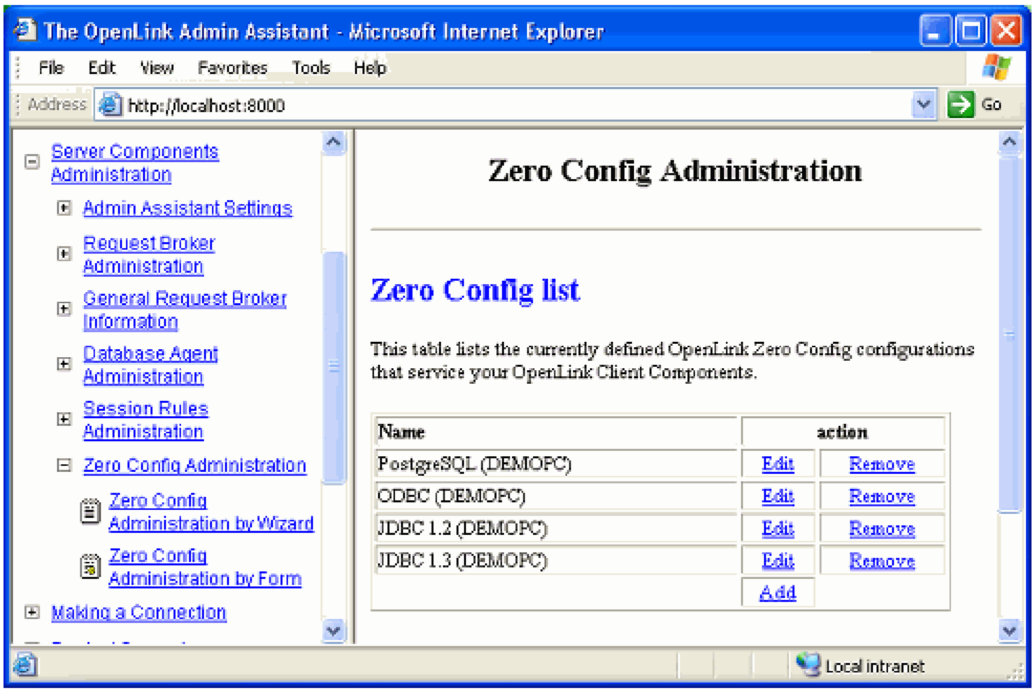
JDBC 1.2 (DEMOPC)
JDBC 1.3 (DEMOPC)
ODBC (DEMOPC)
PostgreSQL (DEMOPC)
SQLSRV_ZC

This table lists the currently defined OpenLink Zero Config configurations that service your OpenLink Client Components.

12.2.4 Configuration by Form

From the Server Components Administration menu, chose the Zero Config Administration sub menu, and then Zero Config Administration by Form. The list of current configurations is shown. Chose one to edit or delete, or select "add" to make a new configuration.

Figure 10.7. Zero Config by Forms - Admin Assistant Configuration


The OpenLink Admin Assistant - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://localhost:8000
Go
Server Components Administration
Admin Assistant Settings
Request Broker Administration
General Request Broker Information
Database Agent Administration
Session Rules Administration
Zero Config Administration
Zero Config Administration by Wizard
Zero Config Administration by Form
Making a Connection
Local intranet

Zero Config Administration

Zero Config list

This table lists the currently defined OpenLink Zero Config configurations that service your OpenLink Client Components.

Name	action	
PostgreSQL (DEMOPC)	Edit	Remove
ODBC (DEMOPC)	Edit	Remove
JDBC 1.2 (DEMOPC)	Edit	Remove
JDBC 1.3 (DEMOPC)	Edit	Remove
	Add	

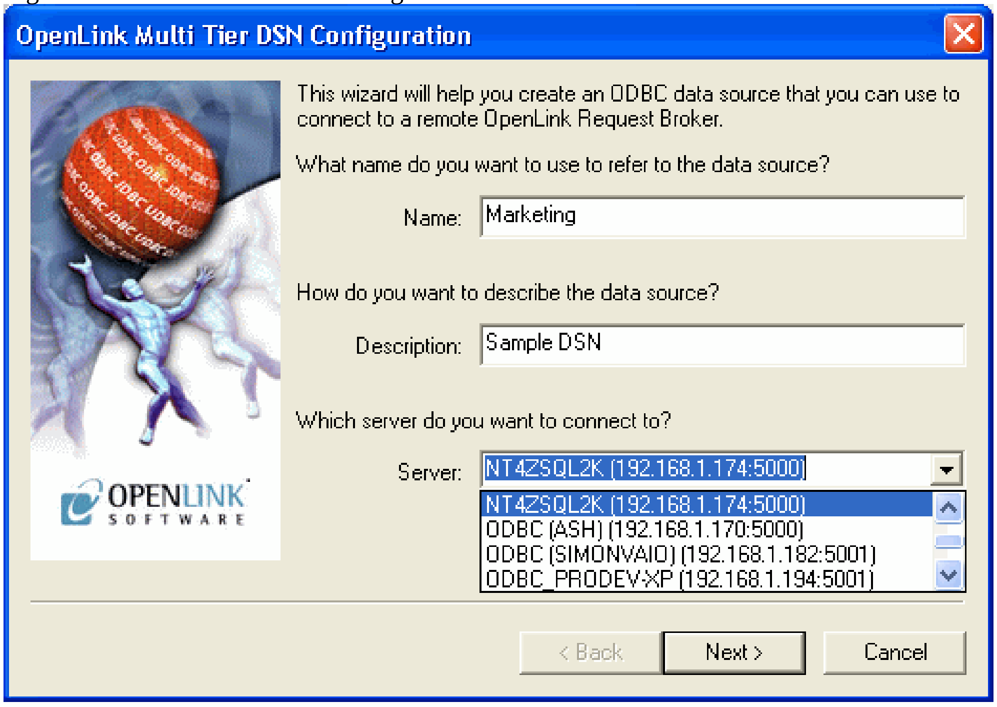
The parameters required in the form are the same as those detailed in the previous section on Rule Book Configuration. The Host and Port fields are not available since this information is supplied by the Request Broker hosting the service.

12.3 Client Configuration

12.3.1 Configuration by Windows ODBC Driver Manager

During the configuration of a DSN in the Windows ODBC Driver Manager the Server field offers a drop down list of available Zero Config servers:

Figure 10.8. Windows ODBC Configuration



This wizard will help you create an ODBC data source that you can use to connect to a remote OpenLink Request Broker.

What name do you want to use to refer to the data source?

Name: Marketing

How do you want to describe the data source?

Description: Sample DSN

Which server do you want to connect to?

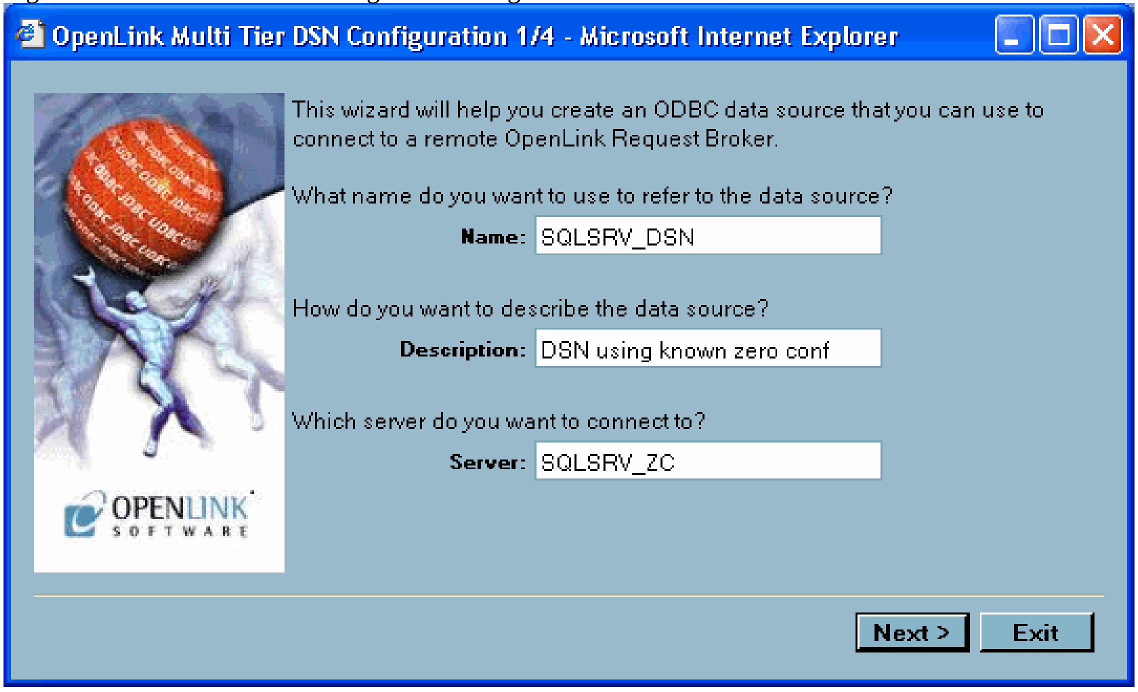
Server: NT4ZSQL2K (192.168.1.174:5000)

NT4ZSQL2K (192.168.1.174:5000)
ODBC (ASH) (192.168.1.170:5000)
ODBC (SIMONVAIO) (192.168.1.182:5001)
ODBC_PRODEV-XP (192.168.1.194:5001)

12.3.2 Configuration by Admin Assistant

During the configuration of a DSN in the Admin Assistant it is currently not possible to discover the Zero Config entries. However on Windows it is possible to put the name of a Zero Config entry in the Server field.

Figure 10.9. Windows Zero Config DSN Configuration



OpenLink Multi Tier DSN Configuration 1/4 - Microsoft Internet Explorer

This wizard will help you create an ODBC data source that you can use to connect to a remote OpenLink Request Broker.

What name do you want to use to refer to the data source?

Name: SQLSRV_DSN

How do you want to describe the data source?

Description: DSN using known zero conf

Which server do you want to connect to?

Server: SQLSRV_ZC

13 Chapter 11. JDBC Components

Abstract
Guide for successfully installing and running the OpenLink JDBC Client Classes.

The OpenLink Drivers for JDBC enable the development, deployment, and utilization of database independent Java Applications, Applets, Servlets, and Bean Components (collectively called JDBC Clients) that conform to the JDBC 1.1.x, or JDBC 2.0.x specifications from JavaSoft.

JDBC clients are built by importing the "java.sql.*" collection of classes known as the JDBC Driver Manager interface. The JDBC Driver Manager uses JDBC URLs to link JDBC clients with JDBC Drivers. It is important to note that JDBC URLs are JDBC Driver specific. Detailed information regarding JDBC is available from: java.sun.com.

Table of Contents

    - ♦ Class OPLJdbcRowSet
    - ♦ Class OPLPooledConnection
    - ♦ Class OPLPoolStatistic
    - ♦ Class OPLRowSetMetaData
- New Features

13.1 OpenLink JDBC Drivers

13.1.1 New JDBC Driver Packages

OpenLink now has a broader range of JDBC Driver types which also transcend JDBC versions.

The drivers are now categorized as follows:

- Generic Drivers for JDBC 1.1.x (compatible with Java Virtual Machine version 1.1.x)
- Generic Drivers for JDBC 2.0 (compatible with Java Virtual Machine version 1.2 & 1.3)
- Generic Drivers for JDBC 3.0 (compatible with Java Virtual Machine version 1.4)
- Megathin Drivers for JDBC 1.1.x (compatible with Java Virtual Machine version 1.1.x)
- Megathin Drivers for JDBC 2.0 (compatible with Java Virtual Machine version 1.2 & 1.3)
- Megathin Drivers for JDBC 3.0 (compatible with Java Virtual Machine version 1.4)

The following table depicts how each of these drivers is packaged, what JDBC Driver formats are supported, and whether they are certified 100% Pure Java or Not:

Table 11.1.

Product Name	Java Package	JDBC Driver Name	JDBC Driver Formats	100% Pure Java ?
Generic Drivers for JDBC 1.1.x	opljdbc.jar	openlink.jdbc.Driver	1,3	N
Generic Drivers for JDBC 2.0	opljdbc2.jar	openlink.jdbc2.Driver	1,3	N
Generic Drivers for JDBC 3.0	opljdbc3.jar	openlink.jdbc3.Driver	1,3	N
Megathin Drivers for JDBC 1.1.x	megathin.jar	openlink.jdbc.Driver	3	Y
Megathin Drivers for JDBC 2.0	megathin2.jar	openlink.jdbc2.Driver	3	Y
Megathin Drivers for JDBC 3.0	megathin3.jar	openlink.jdbc3.Driver	3	Y

13.1.2 New Features & Enhancements

1. Unicode Support. The OpenLink JDBC client is fully Unicode aware.
2. Enhanced Communications Layer. Prior to Release 4.1, the JDBC driver sent preformed database metadata calls in a manner considered to be redundant prior to SELECT statements. With this new release, each SELECT statement only sends and receives TCP packets associated with these metadata calls one time prior to execution.
3. Connection Pooling Support. The JDBC driver now supports connection pooling as defined in the JDBC 3.0 specification.
4.

13.1.2.1 Scrollable Cursors

A new OpenLink RowSet class enables JDBC applications to take advantage of ODBC-style scrollable cursors functionality, with the ability to: retrieve rowsets, go to any row in the resultset or rowset, add new rows to the database, refresh and update any row with a single method invocation, lock and unlock any row, retrieve the current row number, as well as use ODBC-style bookmarks. This is an OpenLink extension to JDBC.

The Drivers for JDBC 2.0 implement the Scrollable Cursor Interfaces introduced in JDBC 2.0. They also retain support for the OpenLink Scrollable Cursor extension so as to provide access to those Scrollable Cursor features that exist in ODBC but are missing from the JDBC 2.0 specification.

5.

13.1.2.2 Array Binding

As part of the new RowSet class. It is now possible to bind data arrays to the columns of the OpenLink RowSet object, and retrieve the data directly into the arrays with a single invocation of the RowSet.next() method. Please see the accompanying demonstration application for an example of its usage.

This approach enables you to retrieve multiple records with a reduced number of calls to the RowSet.next() method. It basically improves JDBC Application performance.

6.

13.1.2.3 openlink.sql.Statement

This is a new OpenLink interface that extends java.sql.Statement to provide additional methods for configuring the ODBC-Style Scrollable Cursors functionality.

You only need this functionality when working with the OpenLink Scrollable Cursor extensions. The Drivers for JDBC 2.0 implement similar features for the JDBC 2.0 Scrollable Cursor specification.

7. Security. The OpenLink client and server components for JDBC encrypt the data sent across the network between the client and server. This provides for enhanced security, particularly over a WAN. This is transparent to the application, and there are no driver specific properties for the application to set.
8. Version self-checking. The OpenLink client component for JDBC now cross checks version numbers with the server Agent for JDBC at connect time, this ensures that compatible components always in use. An exception will be thrown when incompatibilities are encountered, this exception will also contain information about the component versions relating to the exception.

9.

13.1.2.4 Easier Client Version checking

There is now an easier way of checking the OpenLink client for JDBC version. Make sure that the driver is in the classpath and then at your command prompt enter the command:

```
java openlink.jdbc.Driver
```

(for JDK 1.x Drivers)

or

```
java openlink.jdbc2.Driver
```

(for JDK 1.2 or 1.3/2.0 Drivers)

or

```
java openlink.jdbc3.Driver
```

(for JDK 1.4/3.0 Drivers)

or

```
java openlink.jdbc.Driver
```

(for 100% Pure Java Drivers for JDK 1.x)

or

```
java openlink.jdbc2.Driver
```

(for 100% Pure Java Drivers for JDK 1.2 or 1.32.0)

or

```
java openlink.jdbc3.Driver
```

(for 100% Pure Java Drivers for JDK 1.4/3.0)

The driver will respond with appropriate version and release number information.

13.2 Downloading Driver Software

The OpenLink Drivers for JDBC are packaged either as a bundle alongside the other OpenLink data access drivers (ODBC and OLE-DB) that make up the OpenLink Universal Data Access Driver Suite or as a separate release archive which contains only the OpenLink Megathin Drivers for JDBC.

The Megathin Driver for JDBC is a 100% pure Type 3 Network enabled Java Driver.

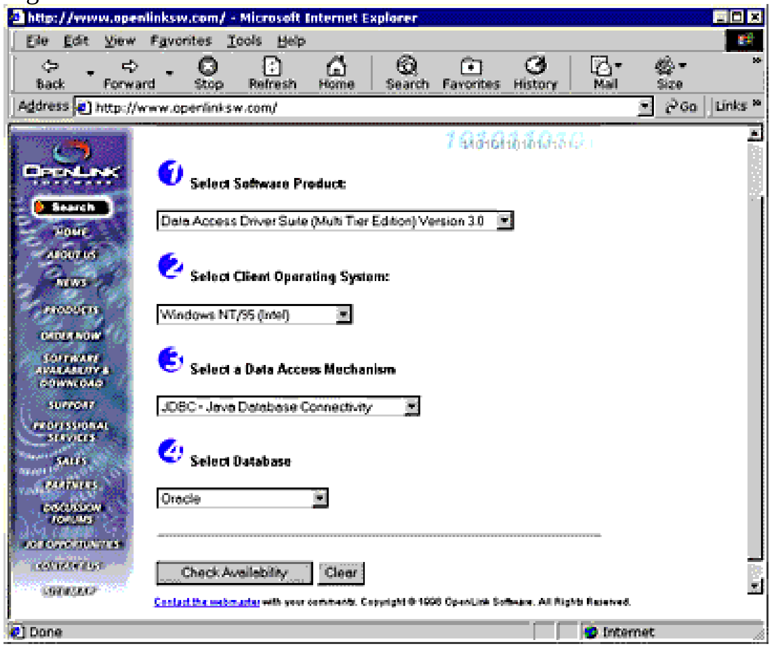
If you are not installing these Drivers from a CD you would have to visit the OpenLink Web Site's download page, http://www.openlinksw.com, to obtain these Drivers.

The screen shots that follow depict the OpenLink download wizard interaction that is required in order to download either the JDBC Driver bundle or the standalone Megathin Drivers.

13.2.1 OpenLink Web Download Wizard Interaction for obtaining OpenLink Drivers for JDBC Bundle

1. Select a Client Operating System from the "Select Client Operating System" listbox and then select a database engine that you will be connecting to via your Driver for JDBC using the "Select Database" listbox.

   Figure 11.1. Client OS


http://www.openlinksw.com/ - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail Size
Address http://www.openlinksw.com/
Go Links
Search
HOME
ABOUT US
NEWS
PRODUCTS
ORDER NOW
SOFTWARE AVAILABILITY & DOWNLOAD
SUPPORT
PROFESSIONAL SERVICES
SALES
PARTNERS
DISCUSSION FORUMS
JOB OPPORTUNITIES
CONTACT US
Check Availability
Clear
Done
Internet

Select Software Product: Data Access Driver Suite (Multi Tier Edition) Version 3.0

Select Client Operating System: Windows NT/95 (Intel)

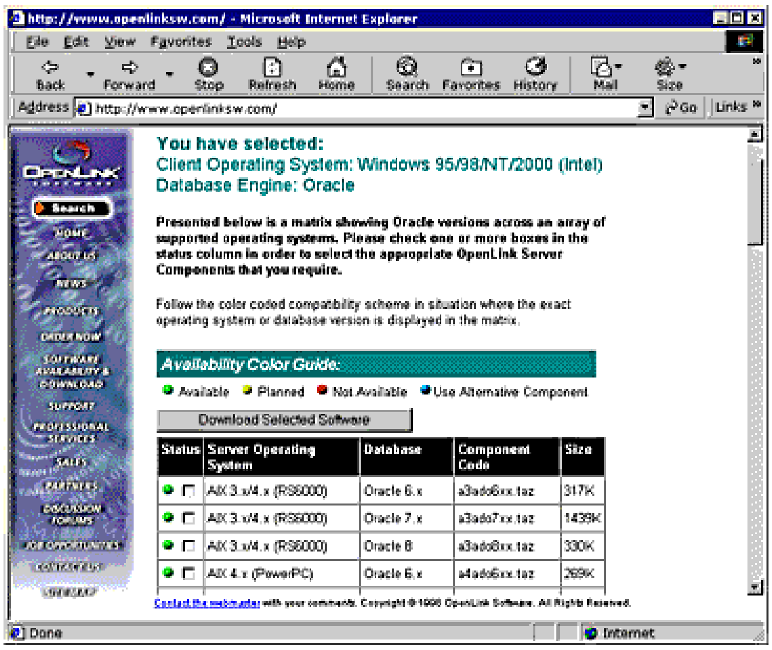
Select a Data Access Mechanism: JDBC - Java Database Connectivity

Select Database: Oracle

Contact the webmaster with your comments. Copyright © 1998 OpenLink Software. All Rights Reserved.

2. Pick one or more server components matching the server operating system that will host the OpenLink Server components required by the Drivers for JDBC. Then click on the "Download Selected Software" button.

   Figure 11.2. Client OS



You have selected:
Client Operating System: Windows 95/98/NT/2000 (Intel)
Database Engine: Oracle

Presented below is a matrix showing Oracle versions across an array of supported operating systems. Please check one or more boxes in the status column in order to select the appropriate OpenLink Server Components that you require.

Follow the color coded compatibility scheme in situation where the exact operating system or database version is displayed in the matrix.

Availability Color Guide:

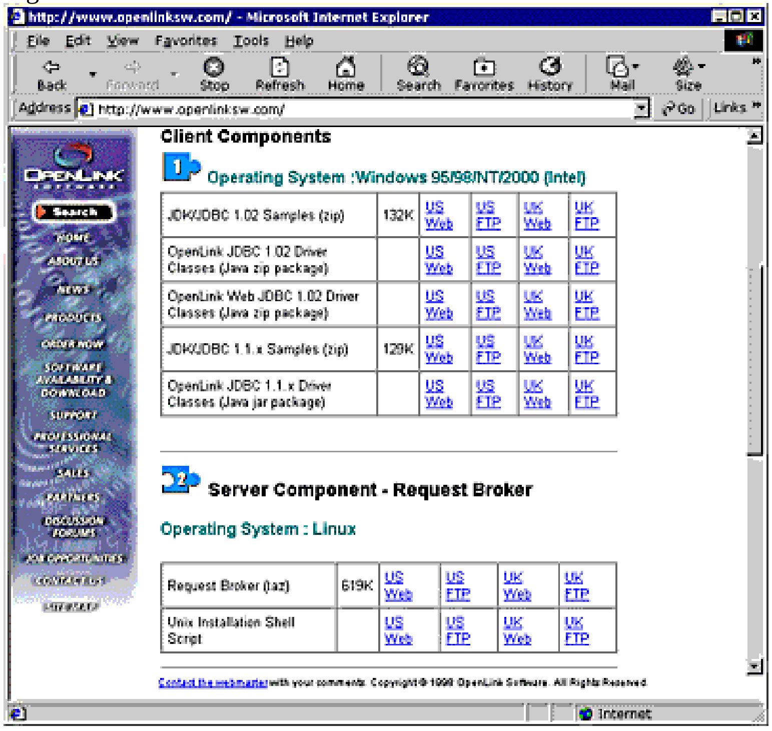
● Available ● Planned ● Not Available ● Use Alternative Component

Status	Server Operating System	Database	Component Code	Size
● ☐	AIX 3.x/4.x (RS6000)	Oracle 6.x	a3ado6xx.taz	317K
● ☐	AIX 3.x/4.x (RS6000)	Oracle 7.x	a3ado7xx.taz	1439K
● ☐	AIX 3.x/4.x (RS6000)	Oracle 8	a3ado8xx.taz	330K
● ☐	AIX 4.x (PowerPC)	Oracle 6.x	a4ado6xx.taz	269K

Contact the webmaster with your comments. Copyright © 1998 OpenLink Software. All Rights Reserved.

3. Download all the software components presented in the "Software Download" page.

Figure 11.3. Software Download



Client Components

Operating System :Windows 95/98/NT/2000 (Intel)

JDK/JDBC 1.02 Samples (zip)	132K	US Web	US FTP	UK Web	UK FTP
OpenLink JDBC 1.02 Driver Classes (Java zip package)		US Web	US FTP	UK Web	UK FTP
OpenLink Web JDBC 1.02 Driver Classes (Java zip package)		US Web	US FTP	UK Web	UK FTP
JDK/JDBC 1.1.x Samples (zip)	129K	US Web	US FTP	UK Web	UK FTP
OpenLink JDBC 1.1.x Driver Classes (Java jar package)		US Web	US FTP	UK Web	UK FTP

Server Component - Request Broker

Operating System : Linux

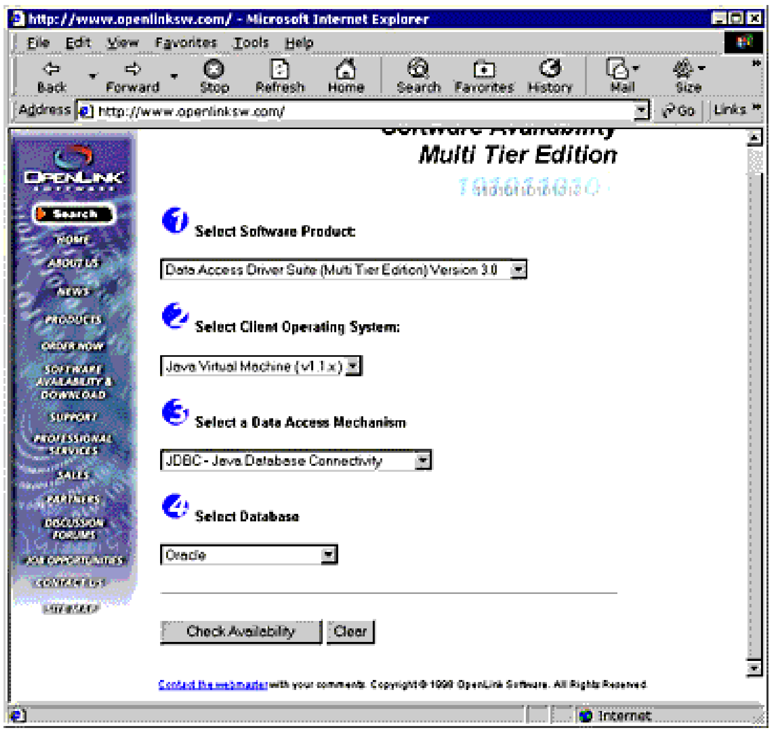
Request Broker (taz)	619K	US Web	US FTP	UK Web	UK FTP
Unix Installation Shell Script		US Web	US FTP	UK Web	UK FTP

Contact the webmaster with your comments. Copyright © 1999 OpenLink Software. All Rights Reserved.

13.2.2 Download Wizard Interaction for obtaining OpenLink Megathin Drivers for JDBC

1. Select a Java Virtual Machine version from the "Select Client Operating System" listbox and then select a database engine that you will be connecting to via your Driver for JDBC using the "Select Database" listbox.

Figure 11.4. Software Download



Multi Tier Edition

1. Select Software Product:

   Data Access Driver Suite (Multi Tier Edition) Version 3.0

2. Select Client Operating System:

   Java Virtual Machine (v1.1.x)

3. Select a Data Access Mechanism

   JDBC - Java Database Connectivity

4. Select Database

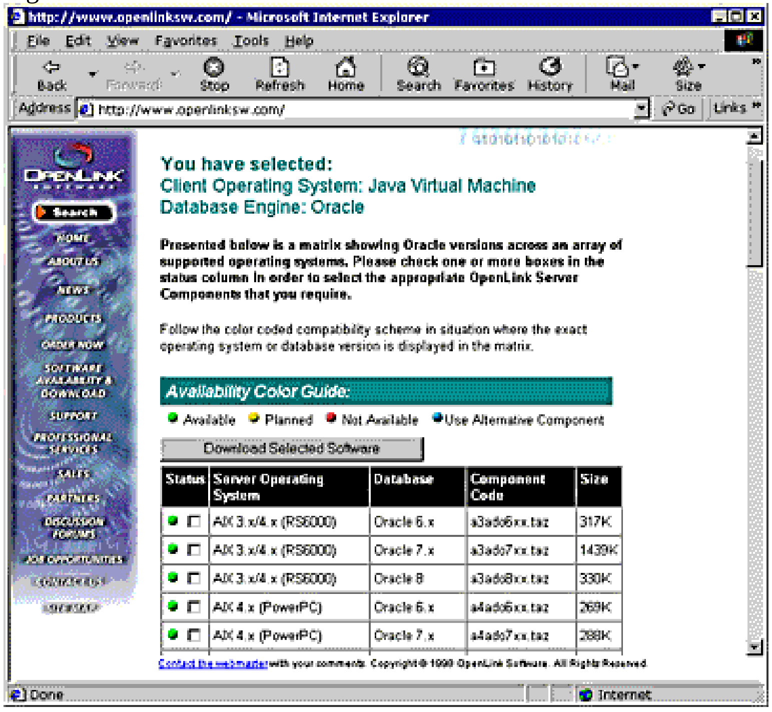
   Oracle

Check Availability | Clear

Contact the webmaster with your comments. Copyright © 1999 OpenLink Software. All Rights Reserved.

2. Pick one or more server components matching the server operating system that will host the OpenLink Server components required by the Drivers for JDBC. Then click on the "Download Selected Software" button.

   Figure 11.5. Software Download



You have selected:
Client Operating System: Java Virtual Machine
Database Engine: Oracle

Presented below is a matrix showing Oracle versions across an array of supported operating systems. Please check one or more boxes in the status column in order to select the appropriate OpenLink Server Components that you require.

Follow the color coded compatibility scheme in situation where the exact operating system or database version is displayed in the matrix.

Availability Color Guide:

● Available ● Planned ● Not Available ● Use Alternative Component

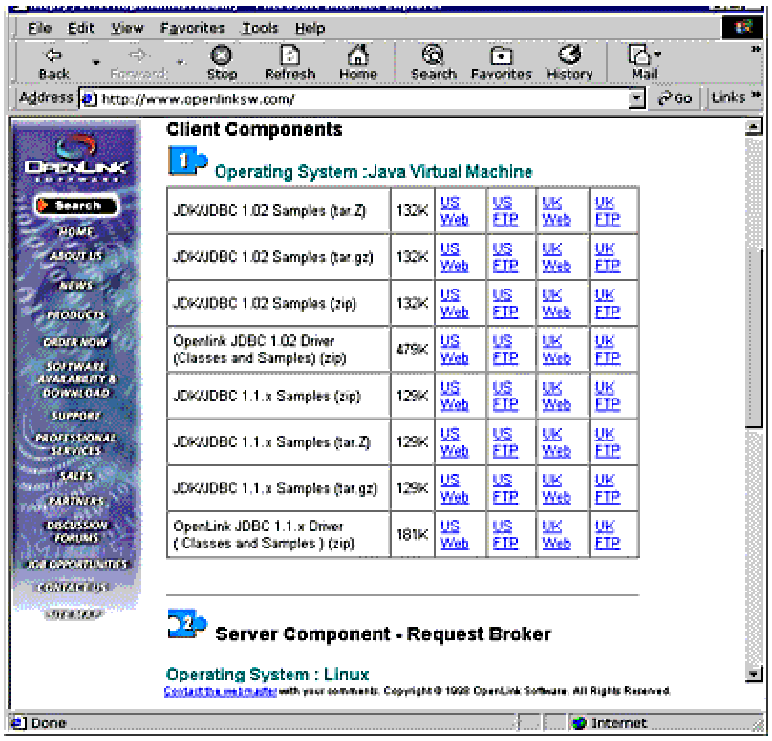
Download Selected Software

Status	Server Operating System	Database	Component Code	Size
● ☐	AIX 3.x/4.x (RS6000)	Oracle 6.x	a3ado6xx.taz	317K
● ☐	AIX 3.x/4.x (RS6000)	Oracle 7.x	a3ado7xx.taz	1439K
● ☐	AIX 3.x/4.x (RS6000)	Oracle 8	a3ado8xx.taz	330K
● ☐	AIX 4.x (PowerPC)	Oracle 6.x	a4ado6xx.taz	269K
● ☐	AIX 4.x (PowerPC)	Oracle 7.x	a4ado7xx.taz	288K

Contact the webmaster with your comments. Copyright © 1999 OpenLink Software. All Rights Reserved.

3. Download all the software components presented in the "Software Download" page.

   Figure 11.6. Software Download


File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail
Address http://www.openlinksw.com/
Go Links
Search HOME ABOUT US NEWS PRODUCTS ORDER NOW SOFTWARE AVAILABILITY & DOWNLOAD SUPPORT PROFESSIONAL SERVICES SALES PARTNERS DISCUSSION FORUMS JOB OPPORTUNITIES CONTACT US SITE MAP
Done
Internet

Client Components

Operating System :Java Virtual Machine

JDK/JDBC 1.02 Samples (tar.Z)	132K	US Web	US FTP	UK Web	UK FTP
JDK/JDBC 1.02 Samples (tar.gz)	132K	US Web	US FTP	UK Web	UK FTP
JDK/JDBC 1.02 Samples (zip)	132K	US Web	US FTP	UK Web	UK FTP
Openlink JDBC 1.02 Driver (Classes and Samples) (zip)	479K	US Web	US FTP	UK Web	UK FTP
JDK/JDBC 1.1.x Samples (zip)	129K	US Web	US FTP	UK Web	UK FTP
JDK/JDBC 1.1.x Samples (tar.Z)	129K	US Web	US FTP	UK Web	UK FTP
JDK/JDBC 1.1.x Samples (tar.gz)	129K	US Web	US FTP	UK Web	UK FTP
OpenLink JDBC 1.1.x Driver (Classes and Samples) (zip)	181K	US Web	US FTP	UK Web	UK FTP

Server Component - Request Broker

Operating System : Linux

Contact the webmaster with your comments. Copyright © 1998 OpenLink Software. All Rights Reserved.

13.3 OpenLink Drivers for JDBC Installation & Configuration

Once you have downloaded your OpenLink drivers for JDBC using the instructions provided above, the next step in the process is the actual configuration of these drivers for use within your operating environment.

Java is operating system independent by virtue of its core philosophy, but JDBC Drivers may or may not be operating system independent as this is JDBC Driver format and implementation specific. The sections that follow walk your through the OpenLink Driver for JDBC installation and configuration process.

13.3.1 Windows 95/98/NT/2000 Based Local Client-Server Environment

In this scenario your Windows machine is acting as the host machine for both your OpenLink client and server components, implying that you are going to install your OpenLink Client and Server components for JDBC on the same machine.

Installation Process

1. Download appropriate driver software installation archive using the instructions provided in the section that covers interaction with the OpenLink Software Download Wizard.
2. As Windows 95/98/NT/2000 is playing the dual role of both Client and Server machine for your OpenLink components, you would have downloaded a ZIP archive that contains both the OpenLink Client & Server components for this platform. Extract the contents of this ZIP archive to a temporary installation folder and then run the "Setup.exe" program
3. The archive you have downloaded will contain the entire suite of Data Access Drivers for this platform. If you do not require the OpenLink ODBC or OLE-DB Drivers simply uncheck these components using the installers component list dialog when presented during the install process.
4. The installer will automatically determine what version of the Java Virtual Machine is installed on your machine and then automatically checks which OpenLink Drivers for JDBC java classes should be checked for installation by default. You can override this settings during the installation process so as to match your specific requirements should they differ from those derived by the installer.
5. The installer will also add the OpenLink Driver for JDBC class files that you have selected in step 3 to the CLASSPATH environment variable on your system
6. Reboot your system.
7. Verify your OpenLink Driver for JDBC client components installation by running one of the following commands (depending on your choice of driver for JDBC) from a DOS Window's command prompt:

Table 11.2. JDBC Driver Version Commands

OpenLink Driver for JDBC Type	Verification Command
Generic Driver for JDBC 1.1.x	java openlink.jdbc.Driver
Generic Driver for JDBC 1.2.x or 2.x	java openlink.jdbc2.Driver
Generic Driver for JDBC 1.3.x	java openlink.jdbc2.Driver
Megathin Driver for JDBC 1.1.x	java openlink.jdbc.Driver
Megathin Driver for JDBC 1.2.x or 2.x	java openlink.jdbc2.Driver
Megathin Driver for JDBC 1.3.x	java openlink.jdbc2.Driver

If you receive output indicating the relevant OpenLink component branding then this indicates that the drivers have been installed correctly and are ready for use with you Java environment, anything else indicates something is wrong. Typically this would be a mismatch between the Java Virtual machine (your default Java environment) and the OpenLink Driver for JDBC classes. Correcting your PATH or CLASSPATH environment variable entries will typically resolve these problems.

8. Verify your OpenLink Driver for JDBC server components installation by starting a Web Browser session and then entering the following URL:

   http://localhost:8000/

   If you are presented with the Home Page of the OpenLink Admin Assistant then this confirms that your OpenLink Server environment is also correctly setup.
9. An additional but non compulsory check that you may perform is to actually verify the existence and state of the OpenLink Database server components called the OpenLink Database agents. Do this by changing into the "bin" sub-directory of the OpenLink base directory. Then run an agent with the --help parameter. For example:

```
       ingii_sv --help   (this will verify the Ingres II Database Agent)
       db2_sv --help     (this will verify the IBM DB2 Database Agent)
```

   See the detailed section about OpenLink Database Agents for additional information.

13.3.2 Windows 95/98/NT/2000 Based Client-Server (2-Tier Configuration) Environment

In this scenario one or more Windows machines act as the host machine for your OpenLink client components, while a separate Windows server machine hosts your OpenLink server components. This Windows server machine also hosts the database engine that you will be connecting to via JDBC, this machine is typically referred to as your Database Server machine.

13.3.2.1 Client Components Installation

1. Download appropriate driver software using the instructions provided in the section that covers interaction with the OpenLink Software Download Wizard on to your designated client machine
2. As Windows 95/98/NT/2000 is playing the single role of Client machine for your OpenLink Drivers for JDBC, you would have downloaded a ZIP archive that contains only the OpenLink Client components. Extract the contents of this ZIP archive to a temporary installation folder and then run the "Setup.exe" program
3. The archive you have downloaded will contain the entire suite of Data Access Drivers for this platform. If you do not require the OpenLink ODBC or OLE-DB Drivers simply uncheck these components using the installers component list dialog when presented during the install process
4. The installer will automatically determine what version of the Java Virtual Machine is installed on your machine and then automatically checks which OpenLink Drivers for JDBC java classes should be checked for installation by default. You can override this settings during the installation process so as to match your specific requirements should they differ from those derived by the installer
5. The installer will also add the OpenLink Driver for JDBC class files that you have selected in step 3 to the CLASSPATH environment variable on your system
6. Reboot your system
7. Verify your OpenLink Driver for JDBC client components installation by running one of the following commands (depending on your choice of driver for JDBC) from a DOS Window's command prompt:

Table 11.3. JDBC Driver Version Commands

OpenLink Driver for JDBC Type	Verification Command
Generic Driver for JDBC 1.1.x	java openlink.jdbc.Driver
Generic Driver for JDBC 1.2.x or 2.x	java openlink.jdbc2.Driver
Generic Driver for JDBC 1.3.x	java openlink.jdbc2.Driver
Megathin Driver for JDBC 1.1.x	java openlink.jdbc.Driver
Megathin Driver for JDBC 1.2.x or 2.x	java openlink.jdbc2.Driver
Megathin Driver for JDBC 1.3.x	java openlink.jdbc2.Driver

If you receive output indicating the relevant OpenLink component branding then this indicates that the drivers have been installed correctly and are ready for use with you Java environment, anything else indicates something is wrong. Typically this would be a mismatch between the Java Virtual machine (your default Java environment) and the OpenLink Driver for JDBC classes. Correcting your PATH or CLASSPATH environment variable entries will typically resolve these problems.

13.3.2.2 Server Components Installation

1. Download appropriate server components software using the instructions provided in the section that covers interaction with the OpenLink Software Download Wizard on to your designated server machine
2. As Windows 95/98/NT/2000 is playing the single role of Server machine for your OpenLink Drivers for JDBC, you would have downloaded a ZIP archive that contains only the OpenLink Server components. Extract the contents of this ZIP archive to a temporary installation folder and then run the "Setup.exe" program
3. The archive you have downloaded will contain both OpenLink client and Server components for this platform. Since you are setting up a Server machine simply uncheck the OpenLink Client components (ODBC, JDBC, OLE-DB) using the installers component list dialog when presented during the install process, this ensures that you only install OpenLink Server components on your Server machine(s)
4. If you are an existing OpenLink user please ensure that you do not have an OpenLink Request Broker process running (check your services control panel item), if there is a Request Broker process running please shut it down at this point
5. Run the "setup.exe" program
6. Start the OpenLink Request Broker, you this by either going into your "Services" control panel (for Windows NT) or to the "OpenLink Data Access Drivers" Windows Start Menu, and then click on the "Broker Startup" menu item
7. Verify your OpenLink Driver for JDBC server components installation by starting a Web Browser session from either your OpenLink Client or Server machine and then enter one of the following URLs:

   From Client Machine:

   ```
   http://<server name or IP address>:8000
   ```

   From Server Machine:

   ```
   http://localhost:8000
   ```

   If you are presented with the Home Page of the OpenLink Admin Assistant then this confirms that your OpenLink Server environment is also correctly setup.
8. An additional but non compulsory check that you may perform is to actually verify the existence and state of the OpenLink Database server components called the OpenLink Database agents. Do this by changing into the "bin" sub-directory of the OpenLink base directory. Then run an agent with the --help parameter. For example:

   ```
   ingii_sv --help   (this will verify the Ingres II Database Agent)
   db2_sv --help     (this will verify the IBM DB2 Database Agent)
   ```

   See the detailed section about OpenLink Database Agents for additional information.

13.3.3 Windows 95/98/NT/2000 Based Application-Server (3-Tier Configuration) Environment

In this scenario your OpenLink Client and Server components for JDBC are installed on an Application Server, as this is where your JDBC based application will be hosted and developed (if you are building a JDBC based 3-Tier solution). Thus, the installation process is broken down into two parts, Application Server, and Database Server components installation. You will not need to install any software on the client machines being used by your JDBC solutions end-users. Application Server Components Installation

1. Download appropriate server components software using the instructions provided in the section that covers interaction with the OpenLink Software Download Wizard on to your designated server machine
2. As this machine needs to host both Client and Server components (by virtue of this machine playing the role of Application Server), you would have downloaded a ZIP archive that contains both the OpenLink Client & Server components for this platform. Extract the contents of this ZIP archive to a temporary installation folder and then run the "Setup.exe" program
3. The archive you have downloaded will contain the entire suite of Data Access Drivers for this platform. If you do not require the OpenLink ODBC or OLE-DB Drivers simply uncheck these components using the installers component list dialog when presented during the install process.
4. If you choose to use OpenLink's Database Independent Networking to connect to remote database engines hosted on one or more dedicated Database Server machine, then ensure that an OpenLink Database Agent checkbox for each Database Engine type is checked from the component list presented by the installer. If on the other hand you choose to use Database Specific Networking provided by your database vendor(s) when connecting to your remote Database Engine(s) hosted on your dedicated Database Server machines, then simply leave all the OpenLink Database Agent checkboxes unchecked.

   SQL*Net, Open Client, Progress Client. I-Connect, Ingres Net, and Netlib are database specific networking products for Oracle, Sybase, Progress, Informix, Ingres, and Microsoft SQL Server respectively.
5. The installer will automatically determine what version of the Java Virtual Machine is installed on your machine and then automatically checks which OpenLink Drivers for JDBC java classes should be checked for installation by default. You can override this settings during the installation process so as to match your specific requirements should they differ from those derived by the installer
6. The installer will also add the OpenLink Driver for JDBC class files that you have selected in step 3 to the CLASSPATH environment variable on your system
7. Reboot your system
8. Verify your OpenLink Driver for JDBC client components installation by running one of the following commands (depending on your choice of driver for JDBC) from a DOS Window's command prompt:

   Table 11.4. JDBC Driver Version Commands

OpenLink Driver for JDBC Type	Verification Command
Generic Driver for JDK 1.1.x	java openlink.jdbc.Driver
Generic Driver for JDK 1.2.x	java openlink.jdbc2.Driver
Generic Driver for JDK 1.3.x	java openlink.jdbc2.Driver
Generic Driver for JDK 1.4.x	java openlink.jdbc3.Driver
Megathin Driver for JDK 1.1.x	java openlink.jdbc.Driver
Megathin Driver for JDK 1.2.x	java openlink.jdbc2.Driver
Megathin Driver for JDK 1.3.x	java openlink.jdbc2.Driver
Megathin Driver for JDK 1.4.x	java openlink.jdbc3.Driver

   If you receive output indicating the relevant OpenLink component branding then this indicates that the drivers have been installed correctly and are ready for use with you Java environment, anything else indicates something is wrong. Typically this would be a mismatch between the Java Virtual machine (your default Java environment) and the OpenLink Driver for JDBC classes. Correcting your PATH or CLASSPATH environment variable entries will typically resolve these problems.
9. Verify your OpenLink Driver for JDBC server components installation by starting a Web Browser session from either your OpenLink Client or Server machine and then enter one of the following URLs:

   From Client Machine:

   ```
   http://<server name or IP address>:8000
   ```

   From Server Machine:

```
http://localhost:8000
```

If you are presented with the Home Page of the "OpenLink Admin Assistant" then this confirms that your OpenLink Server environment is also correctly setup.

10. If you are going to be connecting to your remote database servers using database specific networking provided by one or more database vendors then you need to perform an additional check to ensure that your database agents have been installed properly. Do this by changing into the "bin" sub-directory of the OpenLink base directory. Then run an agent with the --help parameter. For example:

    ```
    ingii_sv --help   (this will verify the Ingres II Database Agent)
    db2_sv --help     (this will verify the IBM DB2 Database Agent)
    ```

    See the detailed section about OpenLink Database Agents for additional information.

13.3.3.1 Database Server Components Installation

This step is only required if your are connecting your Application Server components installed in the prior section to a remote database engine hosted on a dedicated Database Server machine using OpenLink's Database Independent Networking.

1. Download appropriate server components software using the instructions provided in the section that covers interaction with the OpenLink Software Download Wizard on to your designated server machine
2. As Windows 95/98/NT/2000 is playing the single role of a dedicated Database Server machine for your OpenLink Application Server components for JDBC, you would have downloaded a ZIP archive that contains only the OpenLink Server components. Extract the contents of this ZIP archive to a temporary installation folder and then run the "Setup.exe" program
3. The archive you have downloaded will contain both OpenLink client and Server components for this platform. Since you are setting up a Server machine simply uncheck the OpenLink Client components (ODBC, JDBC, OLE-DB) using the installers component list dialog when presented during the install process, this ensures that you only install OpenLink Server components on your Server machine(s)
4. Ensure that an OpenLink Database Agent checkbox for each Database Engine type is checked from the component list presented by the installer. Uncheck all JDBC component related checkboxes unless you anticipate using this dedicated Database Server as an Application Server at a later date.
5. If you are an existing OpenLink user please ensure that you do not have an OpenLink Request Broker process running (check your services control panel item), if there is a Request Broker process running please shut it down at this point
6. Run the "setup.exe" program
7. Start the OpenLink Request Broker, you do this by either going into your "Services" control panel (for Windows NT) or to the "OpenLink Data Access Drivers" Windows Start Menu, and then click on the "Broker Startup" menu item
8. Verify your OpenLink Database server components installation by starting a Web Browser session from either your OpenLink Client, Application or Server machine and then enter one of the following URLs:

    From Client or Application Server Machine:

    ```
    http://<server name or IP address>:8000
    ```

    From Database Server Machine:

    ```
    http://localhost:8000
    ```

    If you are presented with the Home Page of the "OpenLink Admin Assistant" then this confirms that your OpenLink Server environment is also correctly setup.

    An additional but non compulsory check that you may perform is to actually verify the existence and state of the OpenLink Database server components called the OpenLink Database agents. Do this by changing into the "bin" sub-directory of the OpenLink base directory. Then run an agent with the --help parameter. For example:

    ```
    ora8_sv --help    (this will verify the Oracle Database Agent)
    db2_sv --help     (this will verify the IBM DB2 Database Agent)
    ```

    See the detailed section about OpenLink Database Agents for additional information.

13.3.3.2 Linux or UNIX Based Local Client-Server Configuration

In this scenario your Linux or UNIX machine is acting as the host machine for both your OpenLink client and server components, implying that you are going to install your OpenLink Client and Server components for JDBC on the same machine.

Installation Process

1. Download appropriate driver software installation archive using the instructions provided in the section that covers interaction with the OpenLink Software Download Wizard. Ensure that you hatched a checkbox for each Database Engine type that you will be connecting to via JDBC.
2. Move the Request Broker and Database Agent archives into a temporary installation folder on your Linux or UNIX machine then run the following command from the command line prompt:

   ```
   sh install.sh
   ```
3. Follow the instructions presented by the installer for configuring your OpenLink Database Agents.
4. The installer creates an OpenLink environment setup script named "openlink.sh" in the openlink installation's base installation directory. This files contains the following entries which you can modify so as to match the OpenLink Drivers for JDBC to the appropriate Java environment on your machine:

   ```
   CLASSPATH=$CLASSPATH:/dbs/openlink/v40/openlink/jdk1.1.x/opljdbc.jar
   #CLASSPATH=$CLASSPATH:/dbs/openlink/v40/openlink/jdk1.2.x/opljdbc2.zip
   ```
5. Run the script "openlink.sh" (you may also want to add a reference to this in your .profile file) by executing the following command from your Linux or UNIX command line prompt:

   ```
   . openlink.sh
   ```
6. Verify your OpenLink Driver for JDBC client components installation by running one of the following commands (depending on your choice of driver for JDBC) from a DOS Window's command prompt:

   Table 11.5. JDBC Driver Version Commands

OpenLink Driver for JDBC Type	Verification Command
Generic Driver for JDK 1.1.x	java openlink.jdbc.Driver
Generic Driver for JDK 1.2.x	java openlink.jdbc2.Driver
Generic Driver for JDK 1.3.x	java openlink.jdbc2.Driver
Generic Driver for JDK 1.4.x	java openlink.jdbc3.Driver
Megathin Driver for JDK 1.1.x	java openlink.jdbc.Driver
Megathin Driver for JDK 1.2.x	java openlink.jdbc2.Driver
Megathin Driver for JDK 1.3.x	java openlink.jdbc2.Driver
Megathin Driver for JDK 1.4.x	java openlink.jdbc3.Driver

   If you receive output indicating the relevant OpenLink component branding then this indicates that the drivers have been installed correctly and are ready for use with you Java environment, anything else indicates something is wrong. Typically this would be a mismatch between the Java Virtual machine (your default Java environment) and the OpenLink Driver for JDBC classes. Correcting your PATH or CLASSPATH environment variable entries will typically resolve these problems.
7. Verify your OpenLink Driver for JDBC server components installation by starting a Web Browser session and then entering the following URL:

   ```
   http://localhost:8000
   ```

   or

   ```
   http://<hostname of current machine>:8000
   ```

   If you are presented with the Home Page of the OpenLink Admin Assistant then this confirms that your OpenLink Server environment is also correctly setup.
8. An additional but non compulsory check that you may perform is to actually verify the existence and state of the OpenLink Database server components called the OpenLink Database agents. Do this by changing into the "bin" sub-directory of the OpenLink base directory. Then run an agent with the --help parameter. For example:

```
ora8_sv --help   (this will verify the Oracle Database Agent)
db2_sv --help    (this will verify the IBM DB2 Database Agent)
```

See the detailed section about OpenLink Database Agents for additional information.

13.3.3.3 Linux or UNIX Based Client-Server (2-Tier Configuration) Installation

In this scenario one or more Linux or UNIX machines act as the host machine for your OpenLink client components, while a separate Linux or UNIX server machine hosts your OpenLink server components. This Linux or UNIX server machine also hosts the database engine that you will be connecting to via JDBC, this machine is typically referred to as your Database Server machine.

Client Components Installation

1. Download appropriate driver software installation archive using the instructions provided in the section that covers interaction with the OpenLink Software Download Wizard
2. Although Linux or UNIX is only playing role of both Client machine for your OpenLink components, you still need to download a compressed TAR archive containing the OpenLink Request Broker (the download page clearly identifies this archive), this contains both the OpenLink Request Broker and the OpenLink Driver for JDBC components. Move this archive to a temporary installation folder and then run the following installation program:

   ```
   sh install.sh
   ```
3. The installer creates an OpenLink environment setup script named "openlink.sh" in the openlink installation base installation directory. This files contains the following entry which you can modify so as to match the OpenLink Drivers for JDBC to the appropriate Java environment on your machine:

   ```
   CLASSPATH=$CLASSPATH:/dbs/openlink/v40/openlink/jdk1.1.x/opljdbc.jar
   #CLASSPATH=$CLASSPATH:/dbs/openlink/v40/openlink/jdk1.2.x/opljdbc2.zip
   ```
4. Run the script "openlink.sh" (you may also want to add a reference to this in your .profile file) by executing the following command from your Linux or UNIX command line prompt:

   ```
   . openlink.sh
   ```
5. Verify your OpenLink Driver for JDBC client components installation by running one of the following commands (depending on your choice of driver for JDBC) from a DOS Window's command prompt:

   Table 11.6. JDBC Driver Version Commands

OpenLink Driver for JDBC Type	Verification Command
Generic Driver for JDK 1.1.x	java openlink.jdbc.Driver
Generic Driver for JDK 1.2.x	java openlink.jdbc2.Driver
Generic Driver for JDK 1.3.x	java openlink.jdbc2.Driver
Generic Driver for JDK 1.4.x	java openlink.jdbc3.Driver
Megathin Driver for JDK 1.1.x	java openlink.jdbc.Driver
Megathin Driver for JDK 1.2.x	java openlink.jdbc2.Driver
Megathin Driver for JDK 1.3.x	java openlink.jdbc2.Driver
Megathin Driver for JDK 1.4.x	java openlink.jdbc3.Driver

   If you receive output indicating the relevant OpenLink component branding then this indicates that the drivers have been installed correctly and are ready for use with you Java environment, anything else indicates something is wrong. Typically this would be a mismatch between the Java Virtual machine (your default Java environment) and the OpenLink Driver for JDBC classes. Correcting your PATH or CLASSPATH environment variable entries will typically resolve these problems.
6. Verify your OpenLink Driver for JDBC server components installation by starting a Web Browser session and then entering the following URL:

   ```
   http://localhost:8000
   ```

   or

```
        http://<hostname of current machine>:8000
```

If you are presented with the Home Page of the OpenLink Admin Assistant then this confirms that your OpenLink Server environment is also correctly setup.

7. An additional but non compulsory check that you may perform is to actually verify the existence and state of the OpenLink Database server components called the OpenLink Database agents. Do this by changing into the "bin" sub-directory of the OpenLink base directory. Then run an agent with the --help parameter. For example:

```
        ora80_sv --help  (this will verify the Oracle Database Agent)
        db2_sv --help    (this will verify the IBM DB2 Database Agent)
```

   See the detailed section about OpenLink Database Agents for additional information.

Database Server Components Installation

Only perform these steps if you are connecting to database engines hosted on your dedicated Database Server using OpenLink's Database Independent Networking:

1. Download appropriate server components installation archive using the instructions provided in the section that covers interaction with the OpenLink Software Download Wizard. Ensure that you hatched a checkbox for each Database Engine type that you will be connecting to via JDBC.
2. Move the Request Broker and Database Agent archives into a temporary installation folder on your Database Server machine then run the following command from the command line prompt:

```
        sh install.sh
```

3. Follow the instructions presented by the installer for configuring your OpenLink Database Agents
4. An additional but non compulsory check that you may perform is to actually verify the existence and state of the OpenLink Database server components called the OpenLink Database agents. Do this by changing into the "bin" sub-directory of the OpenLink base directory. Then run an agent with the --help parameter. For example:

```
        ora80_sv --help  (this will verify the Oracle Database Agent)
        db2_sv --help    (this will verify the IBM DB2 Database Agent)
```

   See the detailed section about OpenLink Database Agents for additional information.

13.3.3.4 Linux or UNIXBased Application-Server (3-Tier Configuration) Installation

In this scenario your OpenLink Client machine plays the role of an Application Server, as this is where your JDBC based application will be hosted and developed (if you are building a 3-Tier JDBC solution). Thus, the installation process is broken down into two parts, Application Server, and Database Server components installation. You will not need to install any software on the machines being used by your JDBC solution's end-users.

Application Server Components Installation

1. Download appropriate driver software installation archive using the instructions provided in the section that covers interaction with the OpenLink Software Download Wizard. Ensure that you hatched a checkbox for each Database Engine type that you will be connecting to via JDBC.
2. Although Linux or UNIX is only playing role of Client machine for your OpenLink components, you still need to download Linux RPMs or a UNIX compressed TAR archives containing the OpenLink Request Broker and the Database Agents for each database engine that you will be connecting to via JDBC (the download page clearly identifies these archives).
3. Move the Request Broker and this archive to a temporary installation folder, if you choose to use OpenLink's Database Independent Networking to connect to remote database engines hosted on one or more dedicated Database Server machines, do not move the Database Agent archives into the temporary installation directory on the Application Server. Run the following installation programs from the temporary installation directory on your Application Server machine:

   Linux:

```
        rpm -ivh openlink-4.0-2.rpm
```

   Linux (if your Linux system does not have the RPM facility) and UNIX:

```
sh install.sh
```

Linux:

```
rpm -ivh openlink-4.0-2.rpm
```

Linux (if your Linux system does not have the RPM facility) and UNIX:

```
sh install.sh
```

* Ignore the Database Agent configuration menu when presented to you by the installer. *

If on the other hand you choose to use Database Specific Networking provided by your database vendor(s) when connecting to your remote Database Engine(s) hosted on your dedicated Database Server machines, then then move each Database Agent archive into a temporary installation directory alongside the Request Broker archive and then run the following installation programs:

Linux:

```
rpm -ivh openlink-4.0-2.rpm

rpm -ivh openlink-agents-4.0-2.i386-glibc2.rpm
```

(for glibc2 based Linux Environments) or

```
rpm -ivh openlink-agents-4.0-2.i386-libc5.rpm
```

(for libc5 based Linux Environments)

Linux (if your Linux system does not have the RPM facility) and UNIX:

```
sh install.sh
```

4. The installer creates an OpenLink environment setup script named "openlink.sh" in the openlink installation base installation directory. This files contains the following entry which you can modify so as to match the OpenLink Drivers for JDBC to the appropriate Java environment on your machine:

```
CLASSPATH=$CLASSPATH:/dbs/openlink/v40/openlink/jdk1.1.x/opljdbc.jar
#CLASSPATH=$CLASSPATH:/dbs/openlink/v40/openlink/jdk1.2.x/opljdbc2.zip
```

5. Run the script "openlink.sh" (you may also want to add a reference to this in your .profile file) by executing the following command from your Linux or UNIX command line prompt:

```
. openlink.sh
```

6. Verify your OpenLink Driver for JDBC client components installation by running one of the following commands (depending on your choice of driver for JDBC) from a DOS Window's command prompt:

Table 11.7. JDBC Driver Version Commands

OpenLink Driver for JDBC Type	Verification Command
Generic Driver for JDK 1.1.x	java openlink.jdbc.Driver
Generic Driver for JDK 1.2.x	java openlink.jdbc2.Driver
Generic Driver for JDK 1.3.x	java openlink.jdbc2.Driver
Generic Driver for JDK 1.4.x	java openlink.jdbc3.Driver
Megathin Driver for JDK 1.1.x	java openlink.jdbc.Driver
Megathin Driver for JDK 1.2.x	java openlink.jdbc2.Driver
Megathin Driver for JDK 1.3.x	java openlink.jdbc2.Driver
Megathin Driver for JDK 1.4.x	java openlink.jdbc3.Driver

If you receive output indicating the relevant OpenLink component branding then this indicates that the drivers have been installed correctly and are ready for use with you Java environment, anything else indicates something is wrong. Typically this would be a mismatch between the Java Virtual machine (your default Java environment) and the OpenLink Driver for JDBC classes. Correcting your PATH or CLASSPATH environment variable entries will typically resolve these problems.

7. Verify your OpenLink Driver for JDBC server components installation by starting a Web Browser session and then entering the following URL:

   ```
               http://localhost:8000 or http://<hostname of current machine>:8000
   ```

   If you are presented with the Home Page of the OpenLink Admin Assistant then this confirms that your OpenLink Server environment is also correctly setup.
8. Verify your OpenLink Database server components installation by starting a Web Browser session from either your OpenLink Client, Application or Server machine and then enter one of the following URLs:

   From Client Machine:

   ```
       http://<server name or IP address>:8000
   ```

   From Database Server Machine:

   ```
       http://localhost:8000
   ```

   If you are presented with the Home Page of the "OpenLink Admin Assistant" then this confirms that your OpenLink Server environment is also correctly setup.

   An additional but non compulsory check that you may perform is to actually verify the existence and state of the OpenLink Database server components called the OpenLink Database agents. Do this by changing into the "bin" sub-directory of the OpenLink base directory. Then run an agent with the --help parameter. For example:

   ```
       ora80_sv --help  (this will verify the Oracle Database Agent)
       db2_sv --help    (this will verify the IBM DB2 Database Agent)
   ```

   See the detailed section about OpenLink Database Agents for additional information.

13.3.3.5 Database Server Components Installation

Only perform these steps if you are connecting to database engines hosted on your dedicated Database Server using OpenLink's Database Independent Networking:

1. Download appropriate server components installation archive using the instructions provided in the section that covers interaction with the OpenLink Software Download Wizard. Ensure that you hatched a checkbox for each Database Engine type that you will be connecting to via JDBC.
2. Move the Request Broker and Database Agent archives into a temporary installation folder on your Database Server machine then run the following commands from the command line prompt:

   Linux:

   ```
       rpm -ivh openlink-4.0-2.rpm

       rpm -ivh openlink-agents-4.0-2.i386-glibc2.rpm
   ```

   (for glibc2 based Linux Environments) or

   ```
       rpm -ivh openlink-agents-4.0-2.i386-libc5.rpm
   ```

   (for libc5 based Linux Environments)

   Linux (if your Linux system does not have the RPM facility) and UNIX:

   ```
       sh install.sh
   ```
3. Follow the instructions presented by the installer for configuring your OpenLink Database Agents
4. The installer creates an OpenLink environment setup script named "openlink.sh" in the openlink installation's base installation directory.
5. Run the script "openlink.sh" (you may also want to add a reference to this in your .profile file) by executing the following command from your Linux or UNIX command line prompt:

```
. openlink.sh
```

6. Verify your OpenLink Database server components installation by starting a Web Browser session from either your OpenLink Client, Application or Server machine and then enter one of the following URLs:

   From Client or Application Server Machine:

   ```
   http://<server name or IP address>:8000
   ```

   From Database Server Machine:

   ```
   http://localhost:8000
   ```

   If you are presented with the Home Page of the "OpenLink Admin Assistant" then this confirms that your OpenLink Server environment is also correctly setup.

   An additional but non compulsory check that you may perform is to actually verify the existence and state of the OpenLink Database server components called the OpenLink Database agents. Do this by changing into the "bin" sub-directory of the OpenLink base directory. Then run an agent with the --help parameter. For example:

   ```
   ora80_sv --help  (this will verify the Oracle Database Agent)
   db2_sv --help    (this will verify the IBM DB2 Database Agent)
   ```

   See the detailed section about OpenLink Database Agents for additional information.

13.4 Java Based Local Client-Server

In this scenario the Java Virtual Machine is acting as the host of your OpenLink client component for JDBC (a 100% Pure Java Driver for JDBC). The operating system hosting your Java Virtual Machine, also hosts the OpenLink Server server components for JDBC. Thus, you are going to install your OpenLink Client and Server components for JDBC on the same machine.

13.4.1 Client Components Installation Process

1. Download appropriate driver software installation archive using the instructions provided in the section that covers interaction with the OpenLink Software Download Wizard. You would have selected "Java Virtual Machine" as you client operating system when interacting with the OpenLink download Wizard and then have the files "megathin.jar" or "megathin2.jar" presented in the download results page depending on the version of the Java Virtual Machine selected
2. Place the "megathin.jar" or "megathin2.jar" file into directory of your choice then add the directory and reference to the JAR file to your CLASSPATH environment variable. See example below:

   Windows 95/98/NT/2000

   Presuming you place the "megathin.jar" file in the "\program files\openlink\jdk11" on your Windows machine, you would add the following line to your "autoexec.bat" if you are running Windows 95/98:

   ```
           set CLASSPATH=%CLASSPATH%;"c:\program files\openlink\jdk11\megathin.jar":.
   ```

   If you are using NT or Windows 2000 " then you need to open the "System Environment" properties of the "System" Control Panel applet and then add the same entry to the "System Variables" section if you want the driver to be accessible to all users, if not place the entry in the "User Variables" section.

   Linux or UNIX. Presuming you place the "megathin.jar" file in the "/opt/openlink/jdk11" on your Linux or UNIX machine, you would need to modify the following line in the file "openlink.sh" so that they match what is listed below:

   ```
   CLASSPATH=$CLASSPATH:/opt/openlink/jdk1.1.x/megathin.jar
   ```

13.4.2 Server Components Installation

Windows 95/98/NT/2000

1. As Windows 95/98/NT/2000 is playing the dual role of both Client and Server machine for your OpenLink components, you would have downloaded a ZIP archive that contains both the OpenLink Client & Server components for this platform. Extract the contents of this ZIP archive to a temporary installation folder and then run the "Setup.exe" program
2. The archive you have downloaded will contain the entire suite of Data Access Drivers for this platform. If you do not require the OpenLink ODBC or OLE-DB Drivers simply uncheck these components using the installers component list dialog when presented during the install process.
3. Reboot your system
4. Verify your OpenLink Driver for JDBC client components installation by running one of the following commands (depending on your choice of driver for JDBC) from a DOS Window's command prompt:

   Table 11.8. JDBC Driver Version Commands

OpenLink Driver for JDBC Type	Verification Command
Megathin Driver for JDK 1.1.x	java openlink.jdbc.Driver
Megathin Driver for JDK 1.2.x	java openlink.jdbc2.Driver
Megathin Driver for JDK 1.3.x	java openlink.jdbc2.Driver
Megathin Driver for JDK 1.4.x	java openlink.jdbc3.Driver

   If you receive output indicating the relevant OpenLink component branding then this indicates that the drivers have been installed correctly and are ready for use with you Java environment, anything else indicates something is wrong. Typically this would be a mismatch between the Java Virtual machine (your default Java environment) and the OpenLink Driver for JDBC classes. Correcting your PATH or CLASSPATH environment variable entries will typically resolve these problems.
5. Verify your OpenLink Driver for JDBC server components installation by starting a Web Browser session and then entering the following URL:

   ```
   http://localhost:8000
   ```

   If you are presented with the Home Page of the OpenLink Admin Assistant then this confirms that your OpenLink Server environment is also correctly setup.
6. An additional but non compulsory check that you may perform is to actually verify the existence and state of the OpenLink Database server components called the OpenLink Database agents. Do this by changing into the "bin" sub-directory of the OpenLink base directory. Then run an agent with the --help parameter. For example:

   ```
   inf73_sv --help  (this will verify the Informix Database Agent)
   ingii_sv --help  (this will verify the Ingres II Database Agent)
   ```

   See the detailed section about OpenLink Database Agents for additional information.

13.4.3 Linux or UNIX Server Components Installation

1. Download appropriate driver software installation archive using the instructions provided in the section that covers interaction with the OpenLink Software Download Wizard. Ensure that you hatched a checkbox for each Database Engine type that you will be connecting to via JDBC.
2. Move the Request Broker and Database Agent archives into a temporary installation folder on your Linux or UNIX machine then run the following commands from the command line prompt:

   Linux:

   ```
   rpm -ivh openlink-4.0-2.rpm

   rpm -ivh openlink-agents-4.0-2.i386-glibc2.rpm
   ```

   (for glibc2 based Linux Environments) or

   ```
   rpm -ivh openlink-agents-4.0-2.i386-libc5.rpm
   ```

   (for libc5 based Linux Environments)

   Linux (if your Linux system does not have the RPM facility) and UNIX:

```
        sh install.sh
```

3. Follow the instructions presented by the installer for configuring your OpenLink Database Agents.
4. The installer creates an OpenLink environment setup script named "openlink.sh" in the openlink installation's base installation directory. This files contains the following entries which you can modify so as to match the OpenLink Drivers for JDBC to the appropriate Java environment on your machine:

```
        CLASSPATH=$CLASSPATH:/openlink/openlink/jdk1.1.x/megathin.jar
```

13.4.4 Note:

This step is only required because the Linux and UNIX installer archives automatically install all the OpenLink Driver types for JDBC, and also perform the default CLASSPATH entry configuration.

5. Run the script "openlink.sh" (you may also want to add a reference to this in your ".profile" file) by executing the following command from your Linux or UNIX command line prompt:

```
        . openlink.sh
```

6. Verify your OpenLink Driver for JDBC client components installation by running one of the following commands (depending on your choice of driver for JDBC) from a Linux or UNIX command prompt:

Table 11.9. JDBC Driver Version Commands

OpenLink Driver for JDBC Type	Verification Command
Megathin Driver for JDK 1.1.x	java openlink.jdbc.Driver
Megathin Driver for JDK 1.2.x	java openlink.jdbc2.Driver
Megathin Driver for JDK 1.3.x	java openlink.jdbc2.Driver
Megathin Driver for JDK 1.4.x	java openlink.jdbc3.Driver

If you receive output indicating the relevant OpenLink component branding then this indicates that the drivers have been installed correctly and are ready for use with you Java environment, anything else indicates something is wrong. Typically this would be a mismatch between the Java Virtual machine (your default Java environment) and the OpenLink Driver for JDBC classes. Correcting your PATH or CLASSPATH environment variable entries will typically resolve these problems.

7. Verify your OpenLink Driver for JDBC server components installation by starting a Web Browser session and then entering the following URL:

```
        http://localhost:8000
```

or

```
        http://<hostname of current machine>:8000
```

If you are presented with the Home Page of the OpenLink Admin Assistant then this confirms that your OpenLink Server environment is also correctly setup.

8. An additional but non compulsory check that you may perform is to actually verify the existence and state of the OpenLink Database server components called the OpenLink Database agents. Do this by changing into the "bin" sub-directory of the OpenLink base directory. Then run an agent with the --help parameter. For example:

```
        syb11_sv --help  (this will verify the Sybase Database Agent)
        inf73_sv --help  (this will verify the Informix Database Agent)
```

See the detailed section about OpenLink Database Agents for additional information.

13.5 Java Based Client-Server (2-Tier) Installation

In this scenario the Java Virtual Machine and OpenLink Drivers for JDBC reside on separate to OpenLink Server server components for JDBC and Database Connectivity. Thus, you are going to install your OpenLink Client and Server components for JDBC on separate machines, one acting as the Client and the other the Server. The Server also hosts the actual database engine that you will be connecting to via JDBC.

13.5.1 Windows 95/98/NT/2000 Client Components Installation Process

1. Download appropriate driver software installation archive using the instructions provided in the section that covers interaction with the OpenLink Software Download Wizard. You would have selected "Java Virtual Machine" as you client operating system when interacting with the OpenLink download Wizard and then have the files "megathin.jar" or "megathin2.jar" presented in the download results page depending on the version of the Java Virtual Machine selected
2. Place the "megathin.jar" or "megathin2.jar" file into directory of your choice then add the directory and reference to the JAR file to your CLASSPATH environment variable. See example below:

   Windows 95/98/NT/2000

   Presuming you place the "megathin.jar" file in the "\program files\openlink\jdk11" on your Windows machine, you would add the following line to your "autoexec.bat" if you are running Windows 95/98:

   ```
   set CLASSPATH=%CLASSPATH%;"c:\program files\openlink\jdk11\megathin.jar":.
   ```

   If you are using NT or Windows 2000 " then you need to open the "System Environment" properties of the "System" Control Panel applet and then add the same entry to the "System Variables" section if you want the driver to be accessible to all users, if not place the entry in the "User Variables" section.
3. Reboot your machine
4. Verify your OpenLink Driver for JDBC client components installation by running one of the following commands (depending on your choice of driver for JDBC) from a DOS Window's command prompt:

   Table 11.10. JDBC Driver Version Commands

OpenLink Driver for JDBC Type	Verification Command
Megathin Driver for JDK 1.1.x	java openlink.jdbc.Driver
Megathin Driver for JDK 1.2.x	java openlink.jdbc2.Driver
Megathin Driver for JDK 1.3.x	java openlink.jdbc2.Driver
Megathin Driver for JDK 1.4.x	java openlink.jdbc3.Driver

   If you receive output indicating the relevant OpenLink component branding then this indicates that the drivers have been installed correctly and are ready for use with you Java environment, anything else indicates something is wrong. Typically this would be a mismatch between the Java Virtual machine (your default Java environment) and the OpenLink Driver for JDBC classes. Correcting your PATH or CLASSPATH environment variable entries will typically resolve these problems.

13.5.2 Linux or UNIX Client Components Installation

1. Presuming you place the "megathin.jar" file in the "/opt/openlink/jdk11" on your Linux or UNIX machine, you would need to modify the following line in the file "openlink.sh" so that they match what is listed below:

   ```
   CLASSPATH=$CLASSPATH:/opt/openlink/jdk1.1.x/megathin.jar
   ```
2. Run the script "openlink.sh" (you may also want to add a reference to this in your ".profile" file) by executing the following command from your Linux or UNIX command line prompt:

   ```
   . openlink.sh
   ```
3. Verify your OpenLink Driver for JDBC client components installation by running one of the following commands (depending on your choice of driver for JDBC) from a Linux or UNIX command prompt:

   Table 11.11. JDBC Driver Version Commands

OpenLink Driver for JDBC Type	Verification Command
Megathin Driver for JDK 1.1.x	java openlink.jdbc.Driver
Megathin Driver for JDK 1.2.x	java openlink.jdbc2.Driver
Megathin Driver for JDK 1.3.x	java openlink.jdbc2.Driver
Megathin Driver for JDK 1.4.x	java openlink.jdbc3.Driver

If you receive output indicating the relevant OpenLink component branding then this indicates that the drivers have been installed correctly and are ready for use with you Java environment, anything else indicates something is wrong. Typically this would be a mismatch between the Java Virtual machine (your default Java environment) and the OpenLink Driver for JDBC classes. Correcting your PATH or CLASSPATH environment variable entries will typically resolve these problems.

13.5.3 Server Components Installation

Only perform these steps if you are connecting to database engines hosted on your dedicated Database Server using OpenLink's Database Independent Networking:

Windows 95/98/NT/2000

1. As a separate Windows 95/98/NT/2000 is playing the role of Server machine, you would have downloaded a ZIP archive that contains both the OpenLink Client & Server components for this platform. Extract the contents of this ZIP archive to a temporary installation folder on the Windows Server machine and then run the "Setup.exe" program
2. The archive you have downloaded will contain the entire suite of Data Access Drivers for this platform. If you do not require the OpenLink ODBC or OLE-DB Drivers simply uncheck these components using the installers component list dialog when presented during the install process.
3. Reboot your system
4. Verify your OpenLink Driver for JDBC server components installation by starting a Web Browser session and then entering the following URL:

   ```
   http://localhost:8000/
   ```

   If you are presented with the Home Page of the OpenLink Admin Assistant then this confirms that your OpenLink Server environment is also correctly setup.
5. An additional but non compulsory check that you may perform is to actually verify the existence and state of the OpenLink Database server components called the OpenLink Database agents. Do this by changing into the "bin" sub-directory of the OpenLink base directory. Then run an agent with the --help parameter. For example:

   ```
   pro83a_sv --help (this will verify the Progress Database Agent)
   syb11_sv --help  (this will verify the Sybase Database Agent)
   ```

   See the detailed section about OpenLink Database Agents for additional information.

13.5.4 Linux or UNIX Server Components Installation

1. Download appropriate driver software installation archive using the instructions provided in the section that covers interaction with the OpenLink Software Download Wizard. Ensure that you hatched a checkbox for each Database Engine type that you will be connecting to via JDBC.
2. Move the Request Broker and Database Agent archives into a temporary installation folder on your Linux or UNIX machine then run the following commands from the command line prompt:

   Linux:

   ```
   rpm -ivh openlink-4.0-2.rpm

   rpm -ivh openlink-agents-4.0-2.i386-glibc2.rpm
   ```

   (for glibc2 based Linux Environments) or

   ```
   rpm -ivh openlink-agents-4.0-2.i386-libc5.rpm
   ```

   (for libc5 based Linux Environments)

   Linux (if your Linux system does not have the RPM facility) and UNIX:

   ```
   sh install.sh
   ```
3. Follow the instructions presented by the installer for configuring your OpenLink Database Agents.

4. The installer creates an OpenLink environment setup script named "openlink.sh" in the openlink installation's base installation directory. This files contains the following entries which you can modify so as to match the OpenLink Drivers for JDBC to the appropriate Java environment on your machine:

```
CLASSPATH=$CLASSPATH:/openlink/openlink/jdk1.1.x/megathin.jar
```

13.5.5 Note:

This step is only required because the Linux and UNIX installer archives automatically install all the OpenLink Driver types for JDBC, and also perform the default CLASSPATH entry configuration.

5. Run the script "openlink.sh" (you may also want to add a reference to this in your ".profile" file) by executing the following command from your Linux or UNIX command line prompt:

```
. openlink.sh
```

6. Verify your OpenLink Driver for JDBC server components installation by starting a Web Browser session and then entering the following URL:

```
http://localhost:8000
```

or

```
http://<hostname of current machine>:8000
```

If you are presented with the Home Page of the OpenLink Admin Assistant then this confirms that your OpenLink Server environment is also correctly setup.

7. An additional but non compulsory check that you may perform is to actually verify the existence and state of the OpenLink Database server components called the OpenLink Database agents. Do this by changing into the "bin" sub-directory of the OpenLink base directory. Then run an agent with the --help parameter. For example:

```
ingii_sv --help  (this will verify the Ingres II Database Agent)
db2_sv --help    (this will verify the IBM DB2 Database Agent)
```

See the detailed section about OpenLink Database Agents for additional information.

13.6 Java Based Application-Server (3-Tier) Installation

In this scenario the Java Virtual Machine and OpenLink Drivers for JDBC and the OpenLink Server server components for JDBC reside on the same machine which is known as the Application Server. The OpenLink Database Server components reside on a separate Database Server machine (if required) which hosts the database that you will be connecting to via JDBC.

13.6.1 Windows 95/98/NT/2000 Client Components Installation Process.

Windows 95/98/NT/2000 Application Server Components Installation

1. Download appropriate driver software installation archive using the instructions provided in the section that covers interaction with the OpenLink Software Download Wizard. You would have selected "Java Virtual Machine" as your client operating system when interacting with the OpenLink download Wizard and then have the files "megathin.jar" or "megathin2.jar" presented in the download results page depending on the version of the Java Virtual Machine selected
2. Place the "megathin.jar" or "megathin2.jar" file into directory of your choice then add the directory and reference to the JAR file to your CLASSPATH environment variable. See example below:

   Windows 95/98/NT/2000

   Presuming you place the "megathin.jar" file in the "\program files\openlink\jdk11" on your Windows machine, you would add the following line to your "autoexec.bat" if you are running Windows 95/98:

```
set CLASSPATH=%CLASSPATH%;"c:\program files\openlink\jdk11\megathin.jar":.
```

If you are using NT or Windows 2000 " then you need to open the "System Environment" properties of the "System" Control Panel applet and then add the same entry to the "System Variables" section if you want the driver to be accessible to all users, if not place the entry in the "User Variables" section.

3. Reboot your machine
4. Verify your OpenLink Driver for JDBC client components installation by running one of the following commands (depending on your choice of driver for JDBC) from a DOS Window's command prompt:

   Table 11.12. JDBC Driver Version Commands

OpenLink Driver for JDBC Type	Verification Command
Megathin Driver for JDK 1.1.x	java openlink.jdbc.Driver
Megathin Driver for JDK 1.2.x	java openlink.jdbc2.Driver
Megathin Driver for JDK 1.3.x	java openlink.jdbc2.Driver
Megathin Driver for JDK 1.4.x	java openlink.jdbc3.Driver

   If you receive output indicating the relevant OpenLink component branding then this indicates that the drivers have been installed correctly and are ready for use with you Java environment, anything else indicates something is wrong. Typically this would be a mismatch between the Java Virtual machine (your default Java environment) and the OpenLink Driver for JDBC classes. Correcting your PATH or CLASSPATH environment variable entries will typically resolve these problems.
5. Verify your OpenLink Driver for JDBC server components installation by starting a Web Browser session and then entering the following URL:

   http://localhost:8000 or http://<hostname of current machine>:8000

   If you are presented with the Home Page of the OpenLink Admin Assistant then this confirms that your OpenLink Server environment is also correctly setup.
6. Verify your OpenLink Database server components. Do this by changing into the "bin" sub-directory of the OpenLink base directory. Then run an agent with the --help parameter. For example:

   ```
   ora80_sv --help  (this will verify the Oracle Database Agent)
   pro83a_sv --help (this will verify the Progress Database Agent)
   syb10_sv --help  (this will verify the Sybase Database Agent)
   inf73_sv --help  (this will verify the Informix Database Agent)
   ingii_sv --help  (this will verify the Ingres II Database Agent)
   db2_sv --help    (this will verify the IBM DB2 Database Agent)
   ```

   See the detailed section about OpenLink Database Agents for additional information.

13.6.2 Linux or UNIX Application Server Components Installation

1. Presuming you place the "megathin.jar" file in the "/opt/openlink/jdk11" on your Linux or UNIX machine, you would need to modify the following line in the file "openlink.sh" so that they match what is listed below:

   ```
   CLASSPATH=$CLASSPATH:/opt/openlink/jdk1.1.x/megathin.jar
   ```
2. Run the script "openlink.sh" (you may also want to add a reference to this in your ".profile" file) by executing the following command from your Linux or UNIX command line prompt:

   ```
   . openlink.sh
   ```
3. Verify your OpenLink Driver for JDBC client components installation by running one of the following commands (depending on your choice of driver for JDBC) from a DOS Window's command prompt:

   Table 11.13. JDBC Driver Version Commands

OpenLink Driver for JDBC Type	Verification Command
Megathin Driver for JDK 1.1.x	java openlink.jdbc.Driver
Megathin Driver for JDK 1.2.x	java openlink.jdbc2.Driver
Megathin Driver for JDK 1.3.x	java openlink.jdbc2.Driver
Megathin Driver for JDK 1.4.x	java openlink.jdbc3.Driver

If you receive output indicating the relevant OpenLink component branding then this indicates that the drivers have been installed correctly and are ready for use with you Java environment, anything else indicates something is wrong. Typically this would be a mismatch between the Java Virtual machine (your default Java environment) and the OpenLink Driver for JDBC classes. Correcting your PATH or CLASSPATH environment variable entries will typically resolve these problems.

13.6.3 Database Server Components Installation

Only perform these steps if you are connecting to database engines hosted on your dedicated Database Server using OpenLink's Database Independent Networking:

13.6.3.1 Windows 95/98/NT/2000

1. As a separate Windows 95/98/NT/2000 is playing the role of Server machine, you would have downloaded a ZIP archive that contains both the OpenLink Client & Server components for this platform. Extract the contents of this ZIP archive to a temporary installation folder on the Windows Server machine and then run the "Setup.exe" program
2. The archive you have downloaded will contain the entire suite of Data Access Drivers for this platform. If you do not require the OpenLink ODBC or OLE-DB Drivers simply uncheck these components using the installers component list dialog when presented during the install process.
3. Reboot your system.
4. Verify your OpenLink Driver for JDBC server components installation by starting a Web Browser session and then entering the following URL:

   http://localhost:8000/

   If you are presented with the Home Page of the OpenLink Admin Assistant then this confirms that your OpenLink Server environment is also correctly setup.
5. Verify the existence and state of the OpenLink Database server components called the OpenLink Database agents. Do this by changing into the "bin" sub-directory of the OpenLink base directory. Then run an agent with the --help parameter. For example:

   ```
   ingii_sv --help  (this will verify the Ingres II Database Agent)
   db2_sv --help    (this will verify the IBM DB2 Database Agent)
   ```

   See the detailed section about OpenLink Database Agents for additional information.

13.6.3.2 Linux or UNIX Server

1. Download appropriate driver software installation archive using the instructions provided in the section that covers interaction with the OpenLink Software Download Wizard. Ensure that you hatched a checkbox for each Database Engine type that you will be connecting to via JDBC.
2. Move the Request Broker and Database Agent archives into a temporary installation folder on your Linux or UNIX machine then run the following commands from the command line prompt:

   Linux:

   ```
   rpm -ivh openlink-4.0-2.rpm

   rpm -ivh openlink-agents-4.0-2.i386-glibc2.rpm
   ```

   (for glibc2 based Linux Environments) or

   ```
   rpm -ivh openlink-agents-4.0-2.i386-libc5.rpm
   ```

   (for libc5 based Linux Environments)

   Linux (if your Linux system does not have the RPM facility) and UNIX:

   ```
   sh install.sh
   ```
3. Follow the instructions presented by the installer for configuring your OpenLink Database Agents.

4. The installer creates an OpenLink environment setup script named "openlink.sh" in the openlink installation's base installation directory. This files contains the following entries which you can modify so as to match the OpenLink Drivers for JDBC to the appropriate Java environment on your machine:

   ```
   CLASSPATH=$CLASSPATH:/openlink/openlink/jdk1.1.x/megathin.jar
   ```

   ### 13.6.4 Note:

   This step is only required because the Linux and UNIX installer archives automatically install all the OpenLink Driver types for JDBC, and also perform the default CLASSPATH entry configuration.
5. Run the script "openlink.sh" (you may also want to add a reference to this in your ".profile" file) by executing the following command from your Linux or UNIX command line prompt:

   ```
   . openlink.sh
   ```
6. Verify your OpenLink Driver for JDBC server components installation by starting a Web Browser session and then entering the following URL:

   ```
   http://localhost:8000
   ```

   or

   ```
   http://<hostname of current machine>:8000
   ```

   If you are presented with the Home Page of the OpenLink Admin Assistant then this confirms that your OpenLink Server environment is also correctly setup.
7. Verify your OpenLink Database server components, you do this by moving into the "bin" sub-directory of your OpenLink installation's base installation directory. Then run one of the following commands (depending on what database(s) you will be connecting to via JDBC):

   ```
   ora80_sv --help  (this will verify the Oracle Database Agent)
   pro83a_sv --help (this will verify the Progress Database Agent)
   syb11_sv --help  (this will verify the Sybase Database Agent)
   inf73_sv --help  (this will verify the Informix Database Agent)
   ingii_sv --help  (this will verify the Ingres II Database Agent)
   db2_sv --help    (this will verify the IBM DB2 Database Agent)
   ```

   See the detailed section about OpenLink Database Agents for additional information.

13.6.5 Mixed Environment Installations

It is important to note that the client and server operating systems hosting your OpenLink Software do not have to be the same. The Installation instructions have only taken this approach in order to simplify understanding of the installation process. A Linux or UNIX machine can act as a client to a Windows machine and vice versa, all you have to do is follow the steps for installing either the client or server components on the appropriate platform.

13.7 OpenLink Server Components Configuration

The OpenLink JDBC agent and OpenLink Database agents form the OpenLink server components, In the prior section you would have installed these components on the appropriate server machine.

Following installation you have to configure these server components in order to enable connectivity between your backend database(s) and your OpenLink Driver for JDBC. Both of these components are exposed to your OpenLink Driver for JDBC via the OpenLink Request Broker.

This new OpenLink release enables the direct connection to an OpenLink Database agent from your OpenLink Java client classes, as JDBC support in now built in to the database agents. Thus you no longer have to connect via the OpenLink JDBC Agent proxy, as in the past, although connection via this method is still supported. Obviously connecting directly to the Database agent will provide better performance, and is now the recomended connection method whenever possible.

13.7.1 Database Agents

OpenLink Database Agents are the OpenLink data access server components that actually provide database connectivity services to your OpenLink Driver for JDBC. A Database Agent exists for each database engine supported by OpenLink, the supported database list currently includes: Oracle, DB2, Informix, Sybase, Ingres, Progress, Microsoft SQL Server, OpenLink Virtuoso, Solid, PostgresSQL, and other ODBC based databases.

Please follow the instructions provided in the OpenLink Database Agents configuration guide prior to attempting to use your OpenLink Drivers for JDBC (if you haven't already done so as part of the installation process).

13.8 OpenLink Drivers for JDBC Utilization

OpenLink Drivers for JDBC are available in three different JDBC Driver formats.

- JDBC Type 1. Driver for JDBC is implemented as a bridge to ODBC Drivers, thereby implementing the JDBC Driver classes through native methods, this is due to the fact that ODBC is a 'C' language based data access application programming interface. Thus, this driver format is inherently part Java and part Native, implying that it is inherently platform specific rather than independent.
- JDBC Type 3. Driver for JDBC is implemented in Java sitting atop a database independent networking layer bridge also implemented in Java. Thus, the entire driver is Pure Java and thereby operating system independent.

JDBC Applets, Applications, Bean Components, and Servlets communicate with JDBC drivers through JDBC Uniform Resource Locators (URLs). Theses URLs are service request and binding formats implemented slightly differently for each OpenLink Driver for JDBC format. The general JDBC URL format is:

jdbc:<jdbc-subprotocol>:[jdbc implementation specific URL attributes]

The "sub-protocol" component of the URL above identifies each JDBC implementation and typically identifies the JDBC driver vendor, the actual URL attributes are vendor specific. Each OpenLink Driver for JDBC type has a different JDBC URL format, the sections that follow depict and provides examples of these formats.

13.8.1 OpenLink Driver for JDBC Type 1

This driver format connects you to ODBC Data Source Names (DSNs) via JDBC. The URL format is as follows:

```
jdbc:openlink://ODBC[/DSN][/UID][/PWD][/READONLY]
```

URL Format.

/DSN	- ODBC Data Source Name
/UID	- Username
/PWD	- Password
/READONLY	- Determines session mode, read-write or read-only.

Example. If you were attempting to connect to an ODBC DSN on your machine named "CustomerDatabase" in read-only mode then you would enter the following JDBC URL:

```
jdbc:openlink://ODBC/DSN=CustomerDatabase/UID=test/PWD=test/READONLY=Y
```

13.8.2 Note:

In the case of OpenLink ODBC DSNs you do not have to provide values for the /UID and /PWD attributes since these can be controlled and configured on the database or application server using the OpenLink Session Rules Book.

13.8.3 OpenLink Driver for JDBC Type 3

URL Format

This driver format connects you to remote database using remote ODBC DSNs. It also supports direct DSN-Less connections to remote databases. The URL format is as follows:

```
jdbc:openlink://<Hostname>:[portnumber] [/UID] [/PWD] [/READONLY] [/SVT]
     [/APPLICATION] [/FBS|FETCHBUFFERSIZE] [/ENCRYPTED] [/CHARSET] [/UNICODE]
     [/DLF] [/DATABASE] [/OPTIONS] [/DRIVER]
```

URL Attributes.

Hostname - Network Alias or IP address of server machine running an OpenLink Request Broker instance

Port Number - Port number that identifies location of OpenLink JDBC Agent Service, the default value is 5000

/UID - Username

/PWD - Password

/READONLY - Determines session mode, read-write or read-only

/SVT - Determines OpenLink Database Agent type (Oracle, Informix, Sybase, Progress, Ingres, SQL Sever, Sybase etc.)

/APPLICATION - Set the Application name with which connections have to be obtained.

/FBS - (also FETCHBUFFERSIZE) Sets the Fetch Buffer Size, which is the number of JDBC resultset rows that get packed into a single network packet

/ENCRYPTED - Set the Encrypted flag for outgoing packets with which connections have to be obtained. The encryption isn't used by default. The value may be '1', '0', 'Y', 'N', 'y', 'n'

/CHARSET - Set the charset of remote database with which connections have to be obtained. The default value is get from System.getProperty("file.encoding")

/UNICODE - Set the Unicode attribute with which connections have to be obtained. The unicode connection isn't used by default. The value may be '1', '0', 'Y', 'N', 'y', 'n'

/DLF - Set the DeferLongFetch attribute with which connections have to be obtained. The defer Long etch isn't used by default. The value may be '1', '0', 'Y', 'N', 'y', 'n'

/DATABASE - Actual database name within a particular database environment

/OPTIONS - Values used to connect to OpenLink Database Agents to remote database servers using database vendors networking

/DRIVER - IS the Driver name in {}, used when making a DSN-Less connection to a remote ODBC Driver

13.8.4 Examples

13.8.4.1 Connecting To Remote Database

If you were attempting to connect to a remote Database Server Type "SQLServer 2000", hosted on a database server machine with the network alias "pluto", with an OpenLink JDBC server listening at port 5001 (rather than default of 5000), and you wanted this session to be in read-only mode then you would enter the following JDBC URL:

```
jdbc:openlink://pluto:5001/SVT=SQLServer 2000/UID=test/PWD=test/
                READONLY=YES/FBS=55
```

13.8.5 Note:

1. In the case of OpenLink ODBC DSNs you do not have to provide values for the /UID and /PWD attributes since these can be controlled and configured on the database or application server using the OpenLink Session Rules Book
2. "/FBS" ensures that each iteration of a JDBC Resultset fetch loop returns 55 records or less until all records have been retrieved from a remote database server

13.8.5.1 Connecting directly via your OpenLink Database agent

As your OpenLink Database Agent now has built-in support for JDBC, you can make a direct connection to the database using the Type-3 format by specifying the OpenLink Database Type and Database Name attributes as part of your JDBC URL. This is the recommended connection method for a JDBC Type-3 connection whenever possible.

To connect to a remote Microsoft SQL Server 6 or 7 database using this format you would construct the following URL:

```
jdbc:openlink://saturn:5001/SVT=SQLServer 6/DATABASE=pubs/UID=sa/
        PWD=/FBS=55/READONLY=Y
```

13.8.6 Notes:

1. You do not necessarily have to provide values for the /UID and /PWD attributes since these can be controlled and configured on the database or application server using the OpenLink Session Rules Book
2. Since we are using the direct Database Agent connection type, no JDBC agent reference will be needed in the URL string.

13.8.6.1 Connecting To Database via ODBC Driver Without A DSN (DSN-Less Connection)

DSN-Less connections require you to determine the ODBC connect string attributes for the ODBC Driver that you are using. For OpenLink ODBC Drivers these values are:

Table 11.14. Connect String Attributes

Attribute	Description
Driver	Full driver name as defined in the ODBC control panel
ServerType	Database Agent Type
Host	Machine hosting the Database Agent serving an OpenLink ODBC Driver
Username	Valid Database Username
Password	Valid Password for Username
FetchBufferSize	Number of resultset records fetched during each ODBC fetch loop
Database	Actual database name within database server environment
NoLoginBox	Disables OpenLink ODBC Drivers attempt to present dialog when ODBC Driver determines an incomplete ODBC connect string (collection of ODBC attributes passed at connect time) good examples being missing or blank "Username" and "Password" attributes.

If you were connecting a remote Oracle database on a machine called "pluto" and you wanted this session to be read-only, your URL formal would be as follows:

```
jdbc:openlink://pluto/DRIVER={OpenLink Generic ODBC Driver}/Database=ORCL/Username=test/
            PWD=test/ReadOnly=Yes/FBS=55/ServerType=Oracle 8/Host=pluto
```

13.8.6.2 Connecting To Remote Databases on a Separate Server Machine (OpenLink 3-Tier Architecture)

You may choose to install your OpenLink Drivers for JDBC on an Application Server and then install your OpenLink Data Access Server components (Request Broker and Database Agents) on your dedicated database server machine. In such a scenario, you will be connecting to your remote database engine using OpenLink Database Independent networking as opposed to your Database vendor's database specific networking middleware. Please note that you would need the OpenLink server components installed on both the application server and the database server in this scenario.

If you were connecting from an Application Server called "pluto" to a dedicated Database Server machine named "ora_server", hosting an Oracle 8i database identified as "ORCL" you would construct the following JDBC URL:

```
jdbc:openlink://pluto/SVT=JDBC/HOST=ora_server/DATABASE=ORCL/UID=test/PWD=test
```

13.8.7 Notes:

1. This URL string assumes that you have already configured the Generic JDBC client to point to the "Oracle 8.1.x" Server Type. This is done via the Admin Assistant, under "Database Agent Administration".
2. You could also have used the "/DSN" attribute to point to an ODBC DSN which has been configured to connect to the Database Server machine. This requires you to have configured an ODBC DSN on your Application Server machine, but it removes the necessity to embed connection attributes such as Database name and Username/Password in your URL string.

13.8.7.1 Connecting To Remote Databases On Separate Server Machine Using Database Vendor's Networking (Mixed 3-Tier Architecture)

Organizational standards or individual preference may present you with a scenario in which you have two server machines in use: one acting as an Application Server hosting your OpenLink Drivers for JDBC and OpenLink Data Access Server components (Request Broker & Database Agents), and the other acting as a dedicated Database Server. You may not have the necessary authority to install the OpenLink Data Access Server components on the Database Server, or you simply prefer to use your database vendor's networking software which is already configured on your Application Server. This scenario can be described as a "Mixed 3-Tier" architecture, as you are going to use your OpenLink Database Agents atop database-vendor-provided networking, rather than connecting to an OpenLink Database Agent using OpenLink Database-independent networking. (Thus, no OpenLink server components would need to be installed on your Application Server.)

If you were connecting to a remote Oracle 8i database somewhere on your network from an application server called "pluto", you would construct the following JDBC URL (assuming an existing TNS alias / Net*8 server alias, called "ora_pluto"):

```
jdbc:openlink://pluto/SVT=Oracle 8.1.x/UID=test/PWD=test/OPTIONS=ora_pluto
```

13.8.8 Notes:

1. The "/OPTIONS" JDBC URL attribute provides the entry or bind point for connecting OpenLink Database agents to a Database vendor's networking products. This applies to all supported OpenLink databases. See the OpenLink Database Agents configuration guide for additional information relating to the database-specific formats of values passed to the "/OPTIONS" JDBC URL attribute.
2. You could also have used the "/DSN" attribute to point to an ODBC DSN which has been configured to connect to the Database Server machine, this simply reduces the size of your JDBC URL, but imposes the use of DSNs upon you.

13.9 OpenLink Demonstration Programs

To assist you further during your utilization or evaluation of OpenLink's Drivers for JDBC a number of demonstration JDBC compliant Applets and Applications are bundled with your OpenLink Driver for JDBC installation, these programs are provided in both binary and source code format for your free use. The sections that follow guide you through the process of using these programs

13.9.1 JDBC Compliant Applet Demos

Three JDBC applet samples are bundled with your OpenLink Driver for JDBC installation, each one of these demonstrating practical use of JDBC applets and highlighting OpenLink specific functionality. Each of these demos reside in the "samples\jdbc\jdk[10 or 11 or 12]" sub-directory below the directory into which you installed your OpenLink software. Each applet is accessible from the OpenLink Admin Assistant (an OpenLink agent that provides HTTP services like any Web Server does). The programs are:

1. JDBCDemo. demonstrates basic JDBC functionality via an Applet
2. ScrollDemo. demonstrates JDBC functionality via an Applet. It also demonstrates the additional Resultset navigation functionality provided by OpenLink's Scrollable Resultset & RowSet Extensions for JDBC on a Record by Record Basis.
3. ScrollDemo2. demonstrates JDBC functionality via an Applet. It also demonstrates the additional Resultset navigation functionality provided by JDBC 2.0
4. RowSetDemo. demonstrates JDBC functionality via an Applet. It also demonstrates the additional Resultset navigation functionality provided by OpenLink's Scrollable Resultset & RowSet Extensions for JDBC on a RowSet by RowSet Basis.

13.9.2 JDBCDemo

This applet demonstrates basic JDBC functionality via an Applet.

Utilization Steps:

1. Start the OpenLink Request Broker
2. Start up your Web Browser
3. Enter one of the following URLs into your browser depending on the location of your OpenLink Request Broker:

   Local To you:

   ```
   http://localhost:8000
   ```

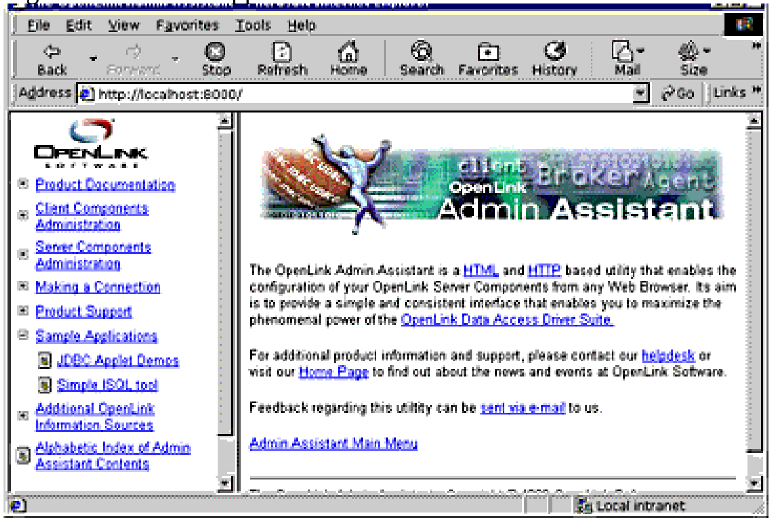
   Remote Server:

   ```
   http://<hostname or IP address of remote server>:8000
   ```

   ### 13.9.3 Note:

   Port "8000" presumes that you provide this value when prompted during your OpenLink Sever components installation.
4. Follow the Admin Assistant's Menu tree to the location of the "Sample Applications->JDBC Applet Demos" menu item. The graphic below depicts this process.

   Figure 11.7. JDBC Applet Demos

   The OpenLink Admin Assistant is a HTML and HTTP based utility that enables the configuration of your OpenLink Server Components from any Web Browser. Its aim is to provide a simple and consistent interface that enables you to maximize the phenomenal power of the OpenLink Data Access Driver Suite.

   For additional product information and support, please contact our helpdesk or visit our Home Page to find out about the news and events at OpenLink Software.

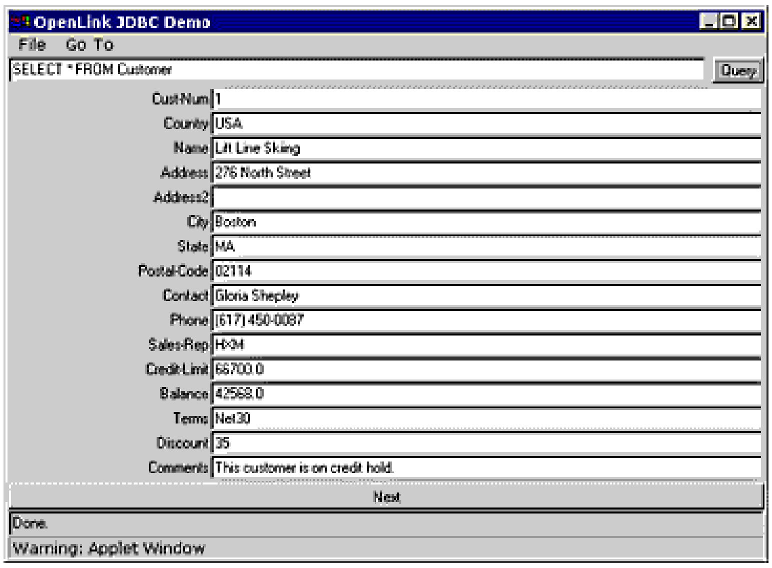
   Feedback regarding this utility can be sent via e-mail to us.

5. Click on the hyperlink that reads "Applet demonstration with OpenLink Software JDBC Driver"
6. Use the Applet's File->Set Connection URL menu item to enter a URL to the data source. If uncertain follows the instructions laid out in the section covering OpenLink JDBC URL formats which shows you how to construct Type 1, 2, and 3 URL formats for your OpenLink Drivers for JDBC. This applet will run with non OpenLink Drivers for JDBC but you will need to obtain URL construction information from the relevant driver vendor.

   The line below depicts the URL construction dialog presented:

   ```
   jdbc:openlink://localhost/SVT=Oracle 7/DATABASE=ORCL/UID=scott/PWD=tiger
   ```
7. Enter a valid SQL statement for the backend database that you are connecting to via JDBC and then click on the "Query" button. The screen shot below depicts this process:

   Figure 11.8. WebJDBC Applet Demo



SELECT * FROM Customer

Cust-Num	1
County	USA
Name	Lift Line Skiing
Address	276 North Street
Address2	
City	Boston
State	MA
Postal-Code	02114
Contact	Gloria Shepley
Phone	(617) 450-0087
Sales-Rep	HXM
Credit-Limit	66700.0
Balance	42568.0
Terms	Net30
Discount	35
Comments	This customer is on credit hold.

Done.

Warning: Applet Window

8. Basic JDBC 1.1 functionality provides Forward-Only as opposed to Bi-Directional record Scrolling, this is why the basic JDBC applet on has a "Next" button. When you click on the "Next" button you are moved to the next record in your JDBC resultset, unfortunately you have to hit the "Query" button again and re-start the Forward-Only resultset navigation if you need to see the First or Prior resultset records from your current position. The examples that follow show how OpenLink and the new release of JDBC (version 2.0) address the Bi-Directional Scrolling Limitation demonstrated by this Applet.

13.9.4 ScrollDemo

This program demonstrates JDBC functionality via an Applet. It also demonstrates the additional Resultset navigation functionality provided by OpenLink's Scrollable Resultset & RowSet Extensions for JDBC on a Record by Record Basis.

Utilization Steps:

1. Start the OpenLink Request Broker.
2. Start up your Web Browser
3. Enter one of the following URLs into your browser depending on the location of your OpenLink Request Broker:

   Local To you:

   http://localhost:8000

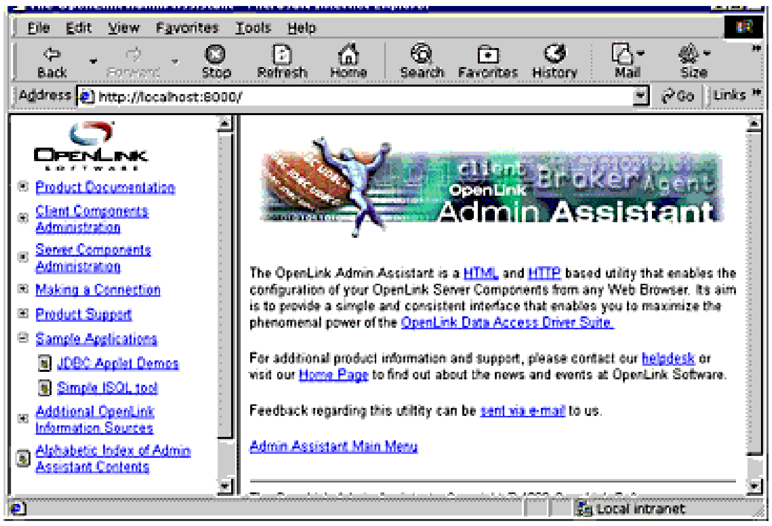
   Remote Server:

   http://<hostname or IP address of remote server>:8000

   ### 13.9.5 Note:

   Port "8000" presumes that you provide this value when prompted during your OpenLink Sever components installation.
4. Follow the Admin Assistant's Menu tree to the location of the "JDBC Applet Demos" menu item. The graphic below depicts this process.

   Figure 11.9. WebJDBC Applet Demo


File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail Size
Address http://localhost:8000/
Go Links
Product Documentation
Client Components Administration
Server Components Administration
Making a Connection
Product Support
Sample Applications
JDBC Applet Demos
Simple ISQL tool
Additional OpenLink Information Sources
Alphabetic Index of Admin Assistant Contents
OpenLink Client Broker Agent Admin Assistant
Local intranet

The OpenLink Admin Assistant is a HTML and HTTP based utility that enables the configuration of your OpenLink Server Components from any Web Browser. Its aim is to provide a simple and consistent interface that enables you to maximize the phenomenal power of the OpenLink Data Access Driver Suite.

For additional product information and support, please contact our helpdesk or visit our Home Page to find out about the news and events at OpenLink Software.

Feedback regarding this utility can be sent via e-mail to us.

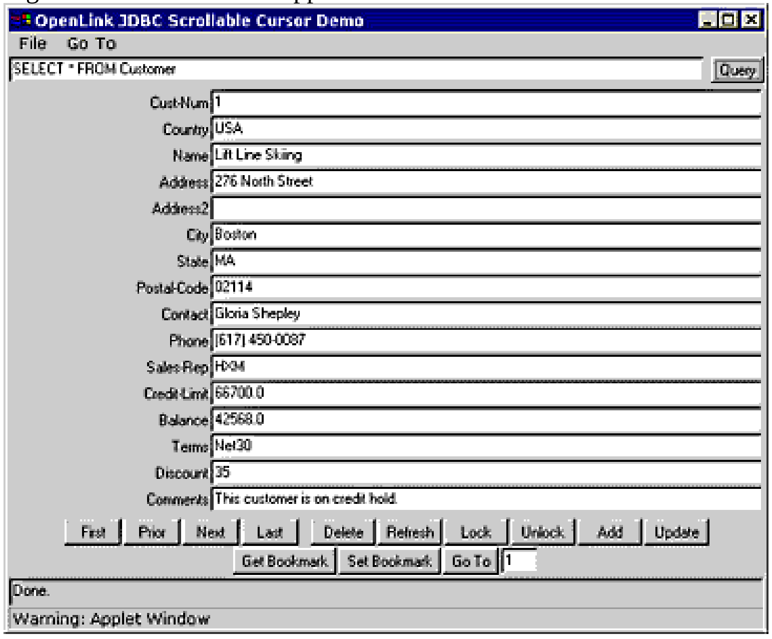
Admin Assistant Main Menu

5. Click on the hyperlink that reads "Applet demonstration with OpenLink Software JDBC Scrollable Cursor extensions"
6. Use the Applet's File->Set Connection URL menu item to enter a URL to the data source. If uncertain follows the instructions laid out in the section covering OpenLink JDBC URL formats which shows you how to construct Type 1, 2, and 3 URL formats for your OpenLink Drivers for JDBC.

   The line below depicts the URL construction dialog presented:

   ```
   jdbc:openlink://localhost/SVT=Oracle 7/DATABASE=ORCL/UID=scott/PWD=tiger
   ```
7. Enter a valid SQL statement for the backend database that you are connecting to via JDBC and then click on the "Query" button. The screen shot below depicts this process:

   Figure 11.10. WebScroll Applet Demo


OpenLink JDBC Scrollable Cursor Demo
File Go To
Query
First Prior Next Last Delete Refresh Lock Unlock Add Update
Get Bookmark Set Bookmark Go To

SELECT * FROM Customer

Cust-Num	1
Country	USA
Name	Lift Line Skiing
Address	276 North Street
Address2	
City	Boston
State	MA
Postal-Code	02114
Contact	Gloria Shepley
Phone	(617) 450-0087
Sales-Rep	HXM
Credit-Limit	66700.0
Balance	42568.0
Terms	Net30
Discount	35
Comments	This customer is on credit hold.

Go To: 1

Done.

Warning: Applet Window

8. JDBC 1.1 functionality provides Forward-Only as opposed to Bi-Directional Resultset Scrolling, OpenLink's Scrollable Resultset Extensions for JDBC enable Bi-Directional Resultset Scrolling. This is why this applet has an additional set of Resultset Navigation buttons: "First","Next", "Prior","Last", "Lock", "Unlock", "Add", "Update", "Get Bookmark", "Set Bookmark", "and Go To" . The existence of Bi-directional Scrollable Resultsets (or Cursors) is often presumed by end-users and developers alike, its importance rarely understood prior to embarking upon JDBC application development or product selection, the unfortunate consequence being complex application re-writes or implementation of sub par JDBC solutions. Each of the button in the applet demo is explained below so as to understand the magnitude of this issue:

   Table 11.15. Scroll Demo Keys Explained

Button	Explanation
First	takes you to first record in the Resultset
Next	takes you to the next record in the Resultset from your current position
Prior	takes you to the previous record in the Resultset from your current position
Last	takes you to the last record in the Resultset
Lock	locks the current record
Unlock	unlocks the current record
Add	add a new record to database
Update	change current record
Delete	remove current record from database
Get Bookmark	mark current record position for future revisit
Set Bookmark	revisit previous marked position in current ResultSet
Go To	go directly to a specific record number within the current ResultSet
Refresh	Reopen current resultset

13.9.6 ScrollDemo2

This applet demonstrates JDBC functionality via an Applet. It also demonstrates the additional Resultset navigation functionality provided by JDBC 2.0

This Applet require a browser that is Java Virtual Machine version 1.2.x or 2.x compliant. If you do not have such a Browser, you can simply run the JDBC Application version of this program.

Utilization Steps:

1. Start the OpenLink Request Broker
2. Start up your Web Browser
3. Enter one of the following URLs into your browser depending on the location of your OpenLink Request Broker:

   Local To you:

   http://localhost:8000/

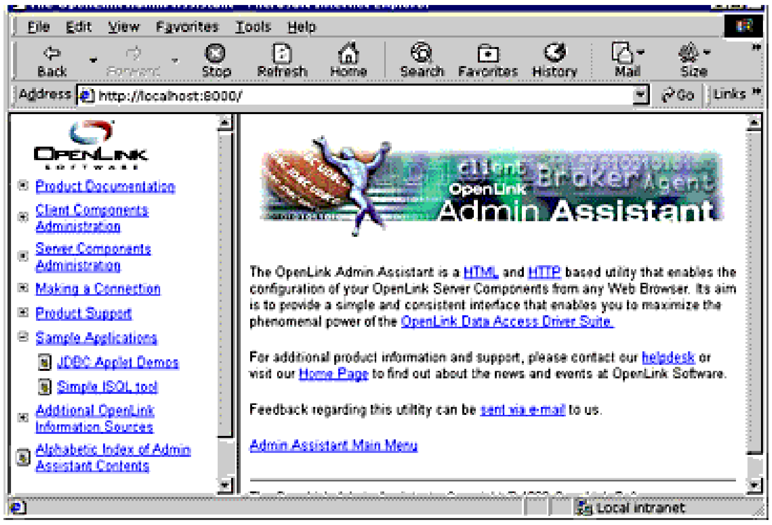
   Remote Server:

   http://<hostname or IP address of remote server>:8000

   ### 13.9.7 Note:

   Port "8000" presumes that you provide this value when prompted during your OpenLink Sever components installation.
4. Follow the Admin Assistant's Menu tree to the location of the "Sample Applications->JDBC Applet Demos" menu item. The graphic below depicts this process.

   Figure 11.11. Scroll2 Applet Demo


File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail Size
Address http://localhost:8000/
Go Links
OpenLink Software
Product Documentation
Client Components Administration
Server Components Administration
Making a Connection
Product Support
Sample Applications
JDBC Applet Demos
Simple ISQL tool
Additional OpenLink Information Sources
Alphabetic Index of Admin Assistant Contents
Client Server Broker Agent OpenLink Admin Assistant
Local intranet

The OpenLink Admin Assistant is a HTML and HTTP based utility that enables the configuration of your OpenLink Server Components from any Web Browser. Its aim is to provide a simple and consistent interface that enables you to maximize the phenomenal power of the OpenLink Data Access Driver Suite.

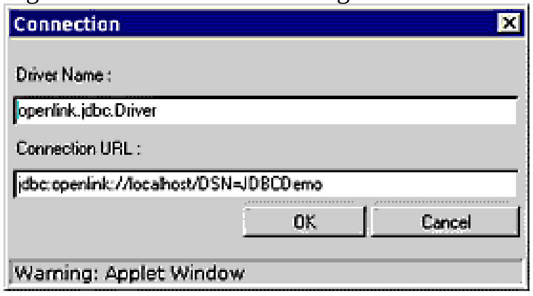
For additional product information and support, please contact our helpdesk or visit our Home Page to find out about the news and events at OpenLink Software.

Feedback regarding this utility can be sent via e-mail to us.

Admin Assistant Main Menu

5. Click on the hyperlink that reads "Applet demonstration with OpenLink Software JDBC 2.0 Scrollable Cursors"
6. Use the Applet's File->Set Connection URL menu item set register your Driver for JDBC 2.0 and then enter a URL pointing to an ODBC DSN. If uncertain follow the instructions laid out in the section covering OpenLink JDBC URL formats which shows you how to construct Type 1, and 3 URL formats for your OpenLink Drivers for JDBC. This applet will run with non OpenLink Drivers for JDBC but you will need to obtain Driver registration and JDBC URL construction information from the relevant Driver vendor.

   The screen shot below depicts the URL construction dialog presented:

   Figure 11.12. Connection dialog
   

   Driver Name :
   openlink.jdbc.Driver
   Connection URL :
   jdbc:openlink://localhost/DSN=JDBCDemo

7. Enter a valid SQL statement for the backend database that you are connecting to via JDBC and then click on the "Query" button.
8. JDBC 1.1 functionality provides Forward-Only as opposed to Bi-Directional Resultset Scrolling, JDBC 2.0 on the other hand supports Bi-Directional Resultset Scrolling. As a result this applet has an additional set of Resultset Navigation buttons: "First","Next", "Previous","Last", "Insert", "Update", "Absolute", "Relative". Unfortunately JDBC 2.0 does not provide Bookmarking or Row Level concurrency control hence the exclusion of the "Lock", "UnLock", "Go To", "Set Bookmark", "Get Bookmark" navigation buttons provided in the "ScrollDemo" applet. To use this functionality in a JDBC 2.0 environment you simply make use of the OpenLink Scrollable ResultSet & RowSet Extensions.

   Each navigation button is described below so as to shed more light on the Scrollable ResultSet functionality provided by JDBC 2.0.

   Table 11.16. Scroll Demo Keys Explained

Button	Explanation
First	takes you to first record in the Resultset
Next	takes you to the next record in the Resultset from your current position
Previous	takes you to the previous record in the Resultset from your current position
Last	takes you to the last record in the Resultset
Add	add a new record to database
Update	change current record
Delete	remove current record from database

Button	Explanation
Relative	moves N number of records forward from the current record where N represents a value entered into the field beside the "Relative" button. IF the field contains a negative number then it indicates a backwards move.
Absolute	go directly to record number N within the current ResultSet where N represents a value entered into the field beside the "Relative" button, the actual direction of Resultset navigation depends on the actual location of the record in question
Refresh	Reopen current resultset

13.9.8 RowSetDemo

This applet demonstrates JDBC functionality via an Applet. It also demonstrates the additional Resultset navigation functionality provided by OpenLink's Scrollable Resultset & RowSet Extensions for JDBC on a RowSet by RowSet Basis.

Utilization Steps:

1. Start the OpenLink Request Broker
2. Start up your Web Browser
3. Enter one of the following URLs into your browser depending on the location of your OpenLink Request Broker:

   Local To you:

   ```
   http://localhost:8000
   ```

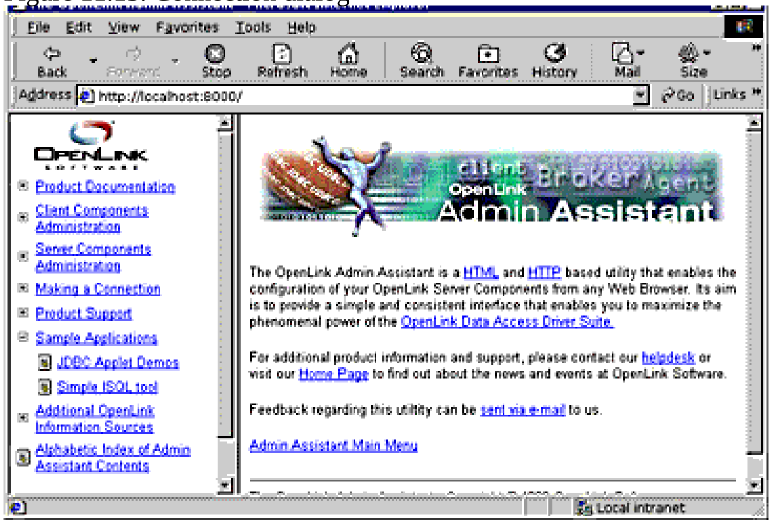
   Remote Server:

   ```
   http://<hostname or IP address of remote server>:8000
   ```

   ### 13.9.9 Note:

   Port "8000" presumes that you provide this value when prompted during your OpenLink Sever components installation.
4. Follow the Admin Assistant's Menu tree to the location of the "Sample Applications->JDBC Applet Demos" menu item. The graphic below depicts this process.

   Figure 11.13. Connection dialog

   
   File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail Size
Address http://localhost:8000/
Go Links
Product Documentation
Client Components Administration
Server Components Administration
Making a Connection
Product Support
Sample Applications
JDBC Applet Demos
Simple ISQL tool
Additional OpenLink Information Sources
Alphabetic Index of Admin Assistant Contents
OpenLink Admin Assistant
Admin Assistant Main Menu
Local intranet

   The OpenLink Admin Assistant is a HTML and HTTP based utility that enables the configuration of your OpenLink Server Components from any Web Browser. Its aim is to provide a simple and consistent interface that enables you to maximize the phenomenal power of the OpenLink Data Access Driver Suite.

   For additional product information and support, please contact our helpdesk or visit our Home Page to find out about the news and events at OpenLink Software.

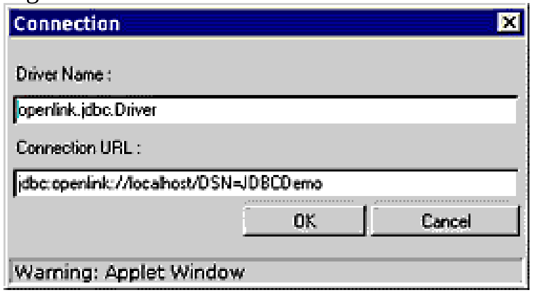
   Feedback regarding this utility can be sent via e-mail to us.

5. Click on the hyperlink that reads "Applet demonstration with OpenLink Software JDBC Scrollable Cursor RowSet Extensions"
6. Use the Applet's File->Set Connection URL menu item set a URL pointing to an ODBC DSN. If uncertain follows the instructions laid out in the section covering OpenLink JDBC URL formats which shows you how to

construct Type 1 and 3 URL formats for your OpenLink Drivers for JDBC. This applet will run with non OpenLink Drivers for JDBC but you will need to obtain URL construction information from the relevant driver vendor.

The screen shot below depicts the URL construction dialog presented:

Figure 11.14. URL Construction



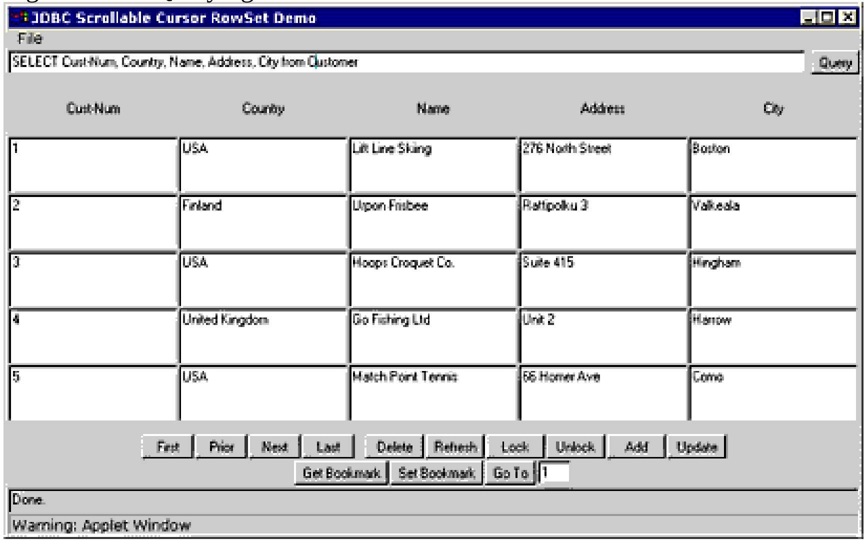
Connection

Driver Name : openlink.jdbc.Driver

Connection URL : jdbc:openlink://localhost/DSN=JDBCDemo

7. Enter a valid SQL statement for the backend database that you are connecting to via JDBC and then click on the "Query" button. The screen shot below depicts this process:

Figure 11.15. Querying



JDBC Scrollable Cursor RowSet Demo

SELECT Cust-Num, Country, Name, Address, City from Customer

Cust-Num	Country	Name	Address	City
1	USA	Lift Line Skiing	276 North Street	Boston
2	Finland	Urpon Frisbee	Rattipolku 3	Valkeala
3	USA	Hoops Croquet Co.	Suite 415	Hingham
4	United Kingdom	Go Fishing Ltd	Unit 2	Harrow
5	USA	Match Point Tennis	66 Homer Ave	Como

Done.

8. JDBC 1.1 functionality provides Forward-Only as opposed to Bi-Directional Resultset Scrolling, JDBC 2.0 provides Scrollable Resultsets but does not provide Bookmarking or Attached RowSets (transient RowSets located in the same process space as the ResultSet). OpenLink's Scrollable Resultset Extensions for JDBC address these issues irrespective of JDBC version. As a result this applet has an additional set of Resultset Navigation buttons when compared to the basic JDBC 2.0 Applet in the prior section, the buttons are: "First","Next", "Prior","Last", "Lock", "Unlock", "Add", "Update", "Get Bookmark", "Set Bookmark", "and Go To" . The existence of transient RowSets due to Bi-directional Scrollable Resultsets (or Cursors) in JDBC 2.0 is more than likely presumed to exist by end-users and developers alike, it is important that you take note of this before embarking upon JDBC 2.0 application development or product selection. Each of the buttons in the Applet demo is explained below so as to assist in the understanding of these matters:

Table 11.17. Scroll Demo Keys Explained

Button	Explanation
First	takes you to first RowSet in the Resultset
Next	takes you to the next RowSet in the Resultset from your current position
Prior	takes you to the previous RowSet in the Resultset from your current position
Last	takes you to the last RowSet in the Resultset
Lock	locks the current RowSet
Unlock	unlocks the current RowSet
Add	add a new record to database

Button	Explanation
Update	change current record with the current RowSet of
Delete	remove current record from within the current RowSet from the database
Get Bookmark	mark current RowSet within Resultset for future revisit
Set Bookmark	revisit previous marked RowSet position in current ResultSet
Go To	go directly to a specific record number within the current ResultSet
Refresh	Reopen current resultset

13.9.10 JDBC compliant Application Demos

All the JDBC compliant Applet demos described in the previous sections have also been implemented as JDBC compliant Applications Demos, you can run these programs in a number of ways depending on operating system hosting your Java Virtual Machine.

13.9.10.1 Windows 95/98/NT/2000

1. Click on your Windows Start Menu Button
2. Select the "OpenLink Data Access Drivers" Start Menu Group
3. Locate the "JDBC Samples" Menu Item
4. Choose from the list of JDBC Applications presented

13.9.10.2 Linux or UNIX

1. Move into your OpenLink base installation directory
2. The move to the following directory listing output maps out the location of the various JDBC Application demos by Java Virtual Machine version:

```
JDBC/jdk1.1.x/Applications:
JDBCDemo RowSetDemo ScrollDemo
JDBC/jdk1.1.x/Applications/JDBCDemo:
DialogConnection.class JDBCDemo.class
DialogConnection.java JDBCDemo.java
JDBC/jdk1.1.x/Applications/RowSetDemo:
DataTextField.class DialogConnection.java readme.txt
DataTextField.java RowSetDemo.class
DialogConnection.class RowSetDemo.java
JDBC/jdk1.1.x/Applications/ScrollDemo:
DialogConnection.class ScrollDemo.class readme.txt
DialogConnection.java ScrollDemo.java
JDBC/jdk1.2.x/Applications:
JDBCDemo RowSetDemo ScrollDemo ScrollDemo2
JDBC/jdk1.2.x/Applications/JDBCDemo:
DialogConnection.class JDBCDemo.class
DialogConnection.java JDBCDemo.java
JDBC/jdk1.2.x/Applications/RowSetDemo:
DataTextField.class DialogConnection.java readme.txt
DataTextField.java RowSetDemo.class
DialogConnection.class RowSetDemo.java
JDBC/jdk1.2.x/Applications/ScrollDemo:
DialogConnection.class ScrollDemo.class readme.txt
DialogConnection.java ScrollDemo.java
JDBC/jdk1.2.x/Applications/ScrollDemo2:
DialogConnection.class ScrollDemo2.class
DialogConnection.java ScrollDemo2.java
JDBC/jdk1.3.x/Applications:
JDBCDemo RowSetDemo ScrollDemo ScrollDemo2
JDBC/jdk1.3.x/Applications/JDBCDemo:
DialogConnection.class JDBCDemo.class
DialogConnection.java JDBCDemo.java
JDBC/jdk1.3.x/Applications/RowSetDemo:
DataTextField.class DialogConnection.java readme.txt
DataTextField.java RowSetDemo.class
DialogConnection.class RowSetDemo.java
JDBC/jdk1.3.x/Applications/ScrollDemo:
DialogConnection.class ScrollDemo.class readme.txt
DialogConnection.java ScrollDemo.java
JDBC/jdk1.3.x/Applications/ScrollDemo2:
DialogConnection.class ScrollDemo2.class
DialogConnection.java ScrollDemo2.java
```

3. Move into the appropriate directory and then execute the following command:

   ```
   java <classname>
   ```

   where "<classname>" represents the JDBC class file hosting your JDBC application demo. For instance if you wanted to run the "RowSetDemo" JDBC application you would type the following:

   ```
   java RowSetDemo
   ```

13.10 Important Multi-User JDBC Solution Development & Utilization Issues

13.10.1 Sensitivity To Changes In Underlying Database

It is extremely important to application developers and end-users alike to understand the degree to which the Resultsets presented to them by a JDBC solution are actually sensitive to underlying changes in the source database. JDBC 1.1 not only fails to provide you with Bi-directional Resultset Scrolling, it also presents what is basically a snapshot of the data in your database at the time a JDBC query is executed. This has the effect of increasing Multi-User JDBC solution development complexity or limiting the functionality and usability of JDBC by end-users.

Sensitive to changes in underlying database takes many forms, this includes: Static, KeySet, Dynamic, and Mixed modes of sensitivity.

Static. - same as basic JDBC, records scrolling occurs over a database snapshot and is insensitive to underlying change by other users

KeySet. - JDBC resultset records scroll over a set of record identifiers uniquely identifying records in the underlying database, this type of scrolling is sensitive to changes is those records with identifiers at the time of query execution. This form of scrolling is insensitive to record record additions or deletions.

Dynamic. - JDBC resultset records scroll over a set of record identifiers uniquely identifying records in the underlying database, these unique identifiers are recreated before each RowSet traversal (collection of resultset records used as scrolling marker or sliding window or Cursor), rather than once at query execution time. This type of scrolling is sensitive to all changes in the underlying database but may introduce a performance penalties depending on the size of RowSets and available network bandwidth.

Mixed. - JDBC resultset records scroll over a set of record identifiers uniquely identifying records in the underlying database, these unique identifiers are created to a limited size (known as the KeySet Size) at query execution time, only when RowSet traversal goes beyond the existing set of unique row identifiers is another collection of unique identifiers assembled. This type of scrolling is sensitive to all changes in the underlying database, but insensitive to Additions or Deletions affecting records in the current RowSet scrolling across a current KeySet, once KeySet boundaries are crossed Insertions or Deletions are recognized. This mode of sensitivity provides increased performance and the expense of reduced sensitivity.

13.10.2 Concurrency Control

In addition to being sensitive to changes in the underlying database, Multi-User applications need to be able to protect users and application processes from the effects of one another when the same record or collection of records are being manipulated at the same time. The process by which these issue are addressed is known as Concurrency Control.

Concurrency control occurs in one of two ways, Optimistic or Pessimistic control.

Optimistic Concurrency Control. - presumes that probability and frequency of multiple users and processes instigating changes to the same database records is low. As result when an end-user or process attempts to change records it first of all determines if the record values at the point of change are still the same as what they were at the time of retrieval. If they are unchanged at the point of change then the change occurs otherwise the change process is rejected and then re-attempted. Although this reduces concurrent user latency, it does have the knock on effect of reducing data integrity if changes rejections aren't managed carefully.

Pessimistic Concurrency Control. - presumes that the probability and frequency of multiple user processing and instigating changes to the same records is high. As a result an end-user or process attempts to changes records it first of all secures Exclusive Locks on the records in question, performs the changes, and then releases the locks. Although this increases and preserves data integrity it does introduce concurrent use latency , which is perceived as performance degradation by the end-user or application developer.

OpenLink's Scrollable ResultSet and RowSet extensions for JDBC all the Multi-User JDBC solution issues raised in this section, our bundled and

```
http://www.openlinksw.com/demo
```

live online demonstrations enable you to evaluate this for yourself and ultimately make a knowledgeable JDBC Driver product and vendor selection.

13.11 JDBC 3 Driver Classes

The OpenLink JDBC driver for JDBC 3.0 has the following classes:

13.11.1 Class BaseRowSet

13.11.1.1 Synopsis

```
abstract public class BaseRowSet implements RowSet,Serializable {
  // Public Constructors
  public BaseRowSet();
  // Public Methods
  public void close() throws java.sql.SQLException;
  public void addRowSetListener(javax.sql.RowSetListener rowsetlistener);
  public void removeRowSetListener(javax.sql.RowSetListener rowsetlistener);
  public void clearParameters() throws java.sql.SQLException;
  public String getCommand();
  public int getConcurrency() throws java.sql.SQLException;
  public String getDataSourceName();
  public boolean getEscapeProcessing() throws java.sql.SQLException;
  public int getFetchDirection() throws java.sql.SQLException;
  public int getFetchSize() throws java.sql.SQLException;
  public int getMaxFieldSize() throws java.sql.SQLException;
  public int getMaxRows() throws java.sql.SQLException;
  public Object[] getParams() throws java.sql.SQLException;
  public String getPassword();
  public int getQueryTimeout() throws java.sql.SQLException;
  public int getTransactionIsolation();
  public int getType() throws java.sql.SQLException;
  public Map getTypeMap() throws java.sql.SQLException;
  public String getUrl() throws java.sql.SQLException;
  public String getUsername();
  public boolean isReadOnly();
  public void setArray(int parameterIndex, java.sql.Array x)
      throws java.sql.SQLException;
  public void setAsciiStream(int parameterIndex, java.io.InputStream x,
                             int length) throws java.sql.SQLException;
  public void setBigDecimal(int parameterIndex, java.math.BigDecimal x)
      throws java.sql.SQLException;
  public void setBinaryStream(int parameterIndex, java.io.InputStream x,
                              int length) throws java.sql.SQLException;
  public void setBlob(int parameterIndex, java.sql.Blob x)
      throws java.sql.SQLException;
  public void setBoolean(int parameterIndex, boolean x)
      throws java.sql.SQLException;
  public void setByte(int parameterIndex, byte x) throws java.sql.SQLException;
  public void setBytes(int parameterIndex, byte[] x)
      throws java.sql.SQLException;
  public void setCharacterStream(int parameterIndex, java.io.Reader x,
                                 int length) throws java.sql.SQLException;
  public void setClob(int parameterIndex, java.sql.Clob x)
      throws java.sql.SQLException;
```

```
    public void setDate(int parameterIndex, java.sql.Date x)
        throws java.sql.SQLException;
    public void setDate(int parameterIndex, java.sql.Date x,
                        java.util.Calendar cal) throws java.sql.SQLException;
    public void setDouble(int parameterIndex, double x)
        throws java.sql.SQLException;
    public void setFloat(int parameterIndex, float x)
        throws java.sql.SQLException;
    public void setInt(int parameterIndex, int x) throws java.sql.SQLException;
    public void setLong(int parameterIndex, long x) throws java.sql.SQLException;
    public void setRef(int parameterIndex, java.sql.Ref x)
        throws java.sql.SQLException;
    public void setShort(int parameterIndex, short x)
        throws java.sql.SQLException;
    public void setString(int parameterIndex, java.lang.String x)
        throws java.sql.SQLException;
    public void setTime(int parameterIndex, java.sql.Time x)
        throws java.sql.SQLException;
    public void setTime(int parameterIndex, java.sql.Time x,
                        java.util.Calendar cal) throws java.sql.SQLException;
    public void setTimestamp(int parameterIndex, java.sql.Timestamp x)
        throws java.sql.SQLException;
    public void setTimestamp(int parameterIndex, java.sql.Timestamp x,
                             java.util.Calendar cal)
        throws java.sql.SQLException;
    public void setUnicodeStream(int parameterIndex, java.io.InputStream x,
                                 int length) throws java.sql.SQLException;
    public void setNull(int parameterIndex, int sqlType)
        throws java.sql.SQLException;
    public void setNull(int parameterIndex, int sqlType,
                        java.lang.String typeName) throws java.sql.SQLException;
    public void setObject(int parameterIndex, java.lang.Object x)
        throws java.sql.SQLException;
    public void setObject(int parameterIndex, java.lang.Object x,
                          int targetSqlType) throws java.sql.SQLException;
    public void setObject(int parameterIndex, java.lang.Object x,
                          int targetSqlType, int scale)
        throws java.sql.SQLException;
    public void setCommand(java.lang.String s) throws java.sql.SQLException;
    public void setConcurrency(int i) throws java.sql.SQLException;
    public void setDataSourceName(java.lang.String s)
        throws java.sql.SQLException;
    public void setEscapeProcessing(boolean flag) throws java.sql.SQLException;
    public void setFetchDirection(int direction) throws java.sql.SQLException;
    public void setFetchSize(int rows) throws java.sql.SQLException;
    public void setMaxFieldSize(int max) throws java.sql.SQLException;
    public void setMaxRows(int max) throws java.sql.SQLException;
    public void setQueryTimeout(int seconds) throws java.sql.SQLException;
    public void setReadOnly(boolean value) throws java.sql.SQLException;
    public void setPassword(java.lang.String s) throws java.sql.SQLException;
    public void setTransactionIsolation(int value) throws java.sql.SQLException;
    public void setType(int value) throws java.sql.SQLException;
    public void setTypeMap(java.util.Map value) throws java.sql.SQLException;
    public void setUrl(java.lang.String s) throws java.sql.SQLException;
    public void setUsername(java.lang.String s) throws java.sql.SQLException;
}
```

Inheritance Path

java.lang.Object

|

openlink.javax.BaseRowSet

13.11.1.2 Members

13.11.1.2.1 Method addRowSetListener(RowSetListener)

```
Synopsis: public void addRowSetListener(
                javax.sql.RowSetListener rowsetlistener);
```

Parameters

`listener` - an event listener

RowSet listener registration. Listeners are notified when an event occurs.

13.11.1.2.2 Method clearParameters()

```
Synopsis: public void clearParameters() throws java.sql.SQLException;
```

Exceptions

SQLException if a database-access error occurs.

In general, parameter values remain in force for repeated use of a RowSet. Setting a parameter value automatically clears its previous value. However, in some cases it is useful to immediately release the resources used by the current parameter values; this can be done by calling clearParameters.

13.11.1.2.3 Method getCommand()

```
Synopsis: public String getCommand();
```

Get the rowset's command property. The command property contains a command string that can be executed to fill the rowset with data. The default value is null.

13.11.1.2.4 Method getConcurrency()

```
Synopsis: public int getConcurrency() throws java.sql.SQLException;
```

Exceptions

SQLException if a database-access error occurs.

Get the rowset concurrency.

13.11.1.2.5 Method getDataSourceName()

```
Synopsis: public String getDataSourceName();
```

The JNDI name that identifies a JDBC data source. Users should set either the url or data source name properties. The most recent property set is used to get a connection.

13.11.1.2.6 Method getEscapeProcessing()

```
Synopsis: public boolean getEscapeProcessing() throws java.sql.SQLException;
```

Exceptions

SQLException if a database-access error occurs.

If escape scanning is on (the default), the driver will do escape substitution before sending the SQL to the database.

13.11.1.2.7 Method getFetchDirection()

```
Synopsis: public int getFetchDirection() throws java.sql.SQLException;
```

Exceptions

SQLException if a database-access error occurs

Determine the fetch direction.

13.11.1.2.8 Method getFetchSize()

```
Synopsis: public int getFetchSize() throws java.sql.SQLException;
```

Determine the default fetch size.

13.11.1.2.9 Method getMaxFieldSize()

```
Synopsis: public int getMaxFieldSize() throws java.sql.SQLException;
```

Exceptions

SQLException	if a database-access error occurs.

The maxFieldSize limit (in bytes) is the maximum amount of data returned for any column value; it only applies to BINARY, VARBINARY, LONGVARBINARY, CHAR, VARCHAR, and LONGVARCHAR columns. If the limit is exceeded, the excess data is silently discarded.

13.11.1.2.10 Method getMaxRows()

```
Synopsis: public int getMaxRows() throws java.sql.SQLException;
```

Exceptions

SQLException	if a database-access error occurs.

The maxRows limit is the maximum number of rows that a RowSet can contain. If the limit is exceeded, the excess rows are silently dropped.

13.11.1.2.11 Method getParams()

```
Synopsis: public Object[] getParams() throws java.sql.SQLException;
```

Exceptions

SQLException	if a database-access error occurs.

Get the parameters that were set on the rowset.

13.11.1.2.12 Method getPassword()

```
Synopsis: public String getPassword();
```

The password used to create a database connection. The password property is set at runtime before calling execute(). It is not usually part of the serialized state of a rowset object.

13.11.1.2.13 Method getQueryTimeout()

```
Synopsis: public int getQueryTimeout() throws java.sql.SQLException;
```

Exceptions

SQLException	if a database-access error occurs.

The queryTimeout limit is the number of seconds the driver will wait for a Statement to execute. If the limit is exceeded, a SQLException is thrown.

13.11.1.2.14 Method getTransactionIsolation()

```
Synopsis: public int getTransactionIsolation();
```

The transaction isolation property contains the JDBC transaction isolation level used.

13.11.1.2.15 Method getType()

```
Synopsis: public int getType() throws java.sql.SQLException;
```

Exceptions

SQLException if a database-access error occurs

Return the type of this result set.

13.11.1.2.16 Method getTypeMap()

```
Synopsis: public Map getTypeMap() throws java.sql.SQLException;
```

Exceptions

SQLException if a database-access error occurs.

Get the type-map object associated with this rowset. By default, the map returned is empty.

13.11.1.2.17 Method getUrl()

```
Synopsis: public String getUrl() throws java.sql.SQLException;
```

Exceptions

SQLException if a database-access error occurs.

Get the url used to create a JDBC connection. The default value is null.

13.11.1.2.18 Method getUsername()

```
Synopsis: public String getUsername();
```

The username used to create a database connection. The username property is set at runtime before calling execute(). It is not usually part of the serialized state of a rowset object.

13.11.1.2.19 Method isReadOnly()

```
Synopsis: public boolean isReadOnly();
```

A rowset may be read-only. Attempts to update a read-only rowset will result in an SQLException being thrown. Rowsets are updateable, by default, if updates are possible.

13.11.1.2.20 Method removeRowSetListener(RowSetListener)

```
Synopsis: public void removeRowSetListener(
                      javax.sql.RowSetListener rowsetlistener);
```

Parameters

listener - an event listener

RowSet listener deregistration.

13.11.1.2.21 Method setArray(int, Array)

```
Synopsis: public void setArray(int parameterIndex, java.sql.Array x)
              throws java.sql.SQLException;
```

Parameters

i - the first parameter is 1, the second is 2, ...

x - an object representing an SQL array

Set an Array parameter.

13.11.1.2.22 Method setAsciiStream(int, InputStream, int)

```
Synopsis: public void setAsciiStream(int parameterIndex, java.io.InputStream x,
                                     int length)
              throws java.sql.SQLException;
```

Parameters

`parameterIndex` - the first parameter is 1, the second is 2, ...

x - the java input stream which contains the ASCII parameter value

length - the number of bytes in the stream

Exceptions

SQLException if a database-access error occurs.

When a very large ASCII value is input to a LONGVARCHAR parameter, it may be more practical to send it via a java.io.InputStream. JDBC will read the data from the stream as needed, until it reaches end-of-file.

Note: This stream object can either be a standard Java stream object or your own subclass that implements the standard interface.

13.11.1.2.23 Method setBigDecimal(int, BigDecimal)

```
Synopsis: public void setBigDecimal(int parameterIndex, java.math.BigDecimal x)
              throws java.sql.SQLException;
```

Parameters

`parameterIndex` - the first parameter is 1, the second is 2, ...

x - the parameter value

Exceptions

SQLException if a database-access error occurs.

Set a parameter to a java.lang.BigDecimal value.

13.11.1.2.24 Method setBinaryStream(int, InputStream, int)

```
Synopsis: public void setBinaryStream(int parameterIndex,
                                      java.io.InputStream x, int length)
              throws java.sql.SQLException;
```

Parameters

`parameterIndex` - the first parameter is 1, the second is 2, ...

x - the java input stream which contains the binary parameter value

length - the number of bytes in the stream

Exceptions

SQLException if a database-access error occurs.

When a very large binary value is input to a LONGVARBINARY parameter, it may be more practical to send it via a java.io.InputStream. JDBC will read the data from the stream as needed, until it reaches end-of-file.

Note: This stream object can either be a standard Java stream object or your own subclass that implements the standard interface.

13.11.1.2.25 Method setBlob(int, Blob)

```
Synopsis: public void setBlob(int parameterIndex, java.sql.Blob x)
              throws java.sql.SQLException;
```

Parameters

i - the first parameter is 1, the second is 2, ...

Parameters

x - an object representing a BLOB

Set a BLOB parameter.

13.11.1.2.26 Method setBoolean(int, boolean)

```
Synopsis: public void setBoolean(int parameterIndex, boolean x)
                 throws java.sql.SQLException;
```

Parameters

parameterIndex - the first parameter is 1, the second is 2, ...

x - the parameter value

Exceptions

SQLException if a database-access error occurs.

Set a parameter to a Java boolean value.

13.11.1.2.27 Method setByte(int, byte)

```
Synopsis: public void setByte(int parameterIndex, byte x)
                 throws java.sql.SQLException;
```

Parameters

parameterIndex - the first parameter is 1, the second is 2, ...

x - the parameter value

Exceptions

SQLException if a database-access error occurs.

Set a parameter to a Java byte value.

13.11.1.2.28 Method setBytes(int, byte[])

```
Synopsis: public void setBytes(int parameterIndex, byte[] x)
                 throws java.sql.SQLException;
```

Parameters

parameterIndex - the first parameter is 1, the second is 2, ...

x - the parameter value

Exceptions

SQLException if a database-access error occurs.

Set a parameter to a Java array of bytes.

13.11.1.2.29 Method setCharacterStream(int, Reader, int)

```
Synopsis: public void setCharacterStream(int parameterIndex, java.io.Reader x,
                                         int length)
                 throws java.sql.SQLException;
```

Parameters

parameterIndex - the first parameter is 1, the second is 2, ...

x - the java reader which contains the UNICODE data

length - the number of characters in the stream

Exceptions

SQLException if a database-access error occurs.

When a very large UNICODE value is input to a LONGVARCHAR parameter, it may be more practical to send it via a java.io.Reader. JDBC will read the data from the stream as needed, until it reaches end-of-file.

Note: This stream object can either be a standard Java stream object or your own subclass that implements the standard interface.

13.11.1.2.30 Method setClob(int, Clob)

```
Synopsis: public void setClob(int parameterIndex, java.sql.Clob x)
                 throws java.sql.SQLException;
```

Parameters

`i` - the first parameter is 1, the second is 2, ...

`x` - an object representing a CLOB

Set a CLOB parameter.

13.11.1.2.31 Method setCommand(String)

```
Synopsis: public void setCommand(java.lang.String s)
                 throws java.sql.SQLException;
```

Parameters

`cmd` - a command string, may be null

Exceptions

SQLException if a database-access error occurs.

Set the rowset's command property. This property is optional. The command property may not be needed when a rowset is produced by a data source that doesn't support commands, such as a spreadsheet.

13.11.1.2.32 Method setConcurrency(int)

```
Synopsis: public void setConcurrency(int i) throws java.sql.SQLException;
```

Parameters

`concurrency` - a value from ResultSet.CONCUR_XXX

Exceptions

SQLException if a database-access error occurs.

Set the rowset concurrency.

13.11.1.2.33 Method setDataSourceName(String)

```
Synopsis: public void setDataSourceName(java.lang.String s)
                 throws java.sql.SQLException;
```

Parameters

`name` - a data source name

Exceptions

SQLException if a database-access error occurs.

Set the data source name.

13.11.1.2.34 Method setDate(int, Date)

```
Synopsis: public void setDate(int parameterIndex, java.sql.Date x)
                 throws java.sql.SQLException;
```

Parameters

`parameterIndex` - the first parameter is 1, the second is 2, ...

`x` - the parameter value

Exceptions

Exceptions

SQLException	if a database-access error occurs.

Set a parameter to a java.sql.Date value.

13.11.1.2.35 Method setDate(int, Date, Calendar)

```
Synopsis: public void setDate(int parameterIndex, java.sql.Date x,
                              java.util.Calendar cal)
            throws java.sql.SQLException;
```

Parameters

parameterIndex - the first parameter is 1, the second is 2, ...

x - the parameter value

Exceptions

SQLException	if a database-access error occurs.

Set a parameter to a java.sql.Date value. The driver converts this to a SQL DATE value when it sends it to the database.

13.11.1.2.36 Method setDouble(int, double)

```
Synopsis: public void setDouble(int parameterIndex, double x)
            throws java.sql.SQLException;
```

Parameters

parameterIndex - the first parameter is 1, the second is 2, ...

x - the parameter value

Exceptions

SQLException	if a database-access error occurs.

Set a parameter to a Java double value.

13.11.1.2.37 Method setEscapeProcessing(boolean)

```
Synopsis: public void setEscapeProcessing(boolean flag)
            throws java.sql.SQLException;
```

Parameters

enable - true to enable; false to disable

Exceptions

SQLException	if a database-access error occurs.

If escape scanning is on (the default), the driver will do escape substitution before sending the SQL to the database.

13.11.1.2.38 Method setFetchDirection(int)

```
Synopsis: public void setFetchDirection(int direction)
            throws java.sql.SQLException;
```

Exceptions

SQLException	if a database-access error occurs, or the result set type is TYPE_FORWARD_ONLY and direction is not FETCH_FORWARD.

Give a hint as to the direction in which the rows in this result set will be processed. The initial value is determined by the statement that produced the result set. The fetch direction may be changed at any time.

13.11.1.2.39 Method setFetchSize(int)

```
Synopsis: public void setFetchSize(int rows) throws java.sql.SQLException;
```

Parameters

rows - the number of rows to fetch

Exceptions

SQLException if a database-access error occurs, or the condition 0 <= rows <= this.getMaxRows() is not satisfied.

Give the JDBC driver a hint as to the number of rows that should be fetched from the database when more rows are needed for this result set. If the fetch size specified is zero, then the JDBC driver ignores the value, and is free to make its own best guess as to what the fetch size should be. The default value is set by the statement that creates the result set. The fetch size may be changed at any time.

13.11.1.2.40 Method setFloat(int, float)

```
Synopsis: public void setFloat(int parameterIndex, float x)
                throws java.sql.SQLException;
```

Parameters

parameterIndex - the first parameter is 1, the second is 2, ...

x - the parameter value

Exceptions

SQLException if a database-access error occurs.

Set a parameter to a Java float value. The driver converts this to a SQL FLOAT value when it sends it to the database.

13.11.1.2.41 Method setInt(int, int)

```
Synopsis: public void setInt(int parameterIndex, int x)
                throws java.sql.SQLException;
```

Parameters

parameterIndex - the first parameter is 1, the second is 2, ...

x - the parameter value

Exceptions

SQLException if a database-access error occurs.

Set a parameter to a Java int value.

13.11.1.2.42 Method setLong(int, long)

```
Synopsis: public void setLong(int parameterIndex, long x)
                throws java.sql.SQLException;
```

Parameters

parameterIndex - the first parameter is 1, the second is 2, ...

x - the parameter value

Exceptions

SQLException if a database-access error occurs.

Set a parameter to a Java long value.

13.11.1.2.43 Method setMaxFieldSize(int)

```
Synopsis: public void setMaxFieldSize(int max) throws java.sql.SQLException;
```

Parameters

max - the new max column size limit; zero means unlimited

Exceptions

SQLException if a database-access error occurs.

The maxFieldSize limit (in bytes) is set to limit the size of data that can be returned for any column value; it only applies to BINARY, VARBINARY, LONGVARBINARY, CHAR, VARCHAR, and LONGVARCHAR fields. If the limit is exceeded, the excess data is silently discarded. For maximum portability use values greater than 256.

13.11.1.2.44 Method setMaxRows(int)

```
Synopsis: public void setMaxRows(int max) throws java.sql.SQLException;
```

Parameters

`max`	- the new max rows limit; zero means unlimited

Exceptions

SQLException	if a database-access error occurs.

The maxRows limit is set to limit the number of rows that any RowSet can contain. If the limit is exceeded, the excess rows are silently dropped.

13.11.1.2.45 Method setNull(int, int)

```
Synopsis: public void setNull(int parameterIndex, int sqlType)
              throws java.sql.SQLException;
```

Parameters

`parameterIndex`	- the first parameter is 1, the second is 2, ...
`sqlType`	- SQL type code defined by java.sql.Types

Exceptions

SQLException	if a database-access error occurs.

Set a parameter to SQL NULL.

Note: You must specify the parameter's SQL type.

13.11.1.2.46 Method setNull(int, int, String)

```
Synopsis: public void setNull(int parameterIndex, int sqlType,
                              java.lang.String typeName)
              throws java.sql.SQLException;
```

Parameters

`parameterIndex`	- the first parameter is 1, the second is 2, ...
`sqlType`	- a value from java.sql.Types
`typeName`	- the fully-qualified name of an SQL user-named type, ignored if the parameter is not a user-named type or REF

Exceptions

SQLException	if a database access error occurs

Sets the designated parameter to SQL NULL. This version of setNull should be used for user-named types and REF type parameters. Examples of user-named types include: STRUCT, DISTINCT, JAVA_OBJECT, and named array types.

Note: To be portable, applications must give the SQL type code and the fully-qualified SQL type name when specifying a NULL user-defined or REF parameter. In the case of a user-named type the name is the type name of the parameter itself. For a REF parameter the name is the type name of the referenced type. If a JDBC driver does not need the type code or type name information, it may ignore it. Although it is intended for user-named and Ref parameters, this method may be used to set a null parameter of any JDBC type. If the parameter does not have a user-named or REF type, the given typeName is ignored.

13.11.1.2.47 Method setObject(int, Object)

```
Synopsis: public void setObject(int parameterIndex, java.lang.Object x)
              throws java.sql.SQLException;
```

Parameters

`parameterIndex` - The first parameter is 1, the second is 2, ...

`x` - The object containing the input parameter value

Exceptions

SQLException if a database-access error occurs.

Set the value of a parameter using an object; use the java.lang equivalent objects for integral values.

The JDBC specification specifies a standard mapping from Java Object types to SQL types. The given argument java object will be converted to the corresponding SQL type before being sent to the database.

Note that this method may be used to pass datatabase specific abstract data types, by using a Driver specific Java type. If the object is of a class implementing SQLData, the rowset should call its method writeSQL() to write it to the SQL data stream. else If the object is of a class implementing Ref, Blob, Clob, Struct, or Array then pass it to the database as a value of the corresponding SQL type. Raise an exception if there is an ambiguity, for example, if the object is of a class implementing more than one of those interfaces.

13.11.1.2.48 Method setObject(int, Object, int)

```
Synopsis: public void setObject(int parameterIndex, java.lang.Object x,
                                int targetSqlType)
              throws java.sql.SQLException;
```

Exceptions

SQLException if a database-access error occurs.

This method is like setObject above, but the scale used is the scale of the second parameter. Scalar values have a scale of zero. Literal values have the scale present in the literal. While it is supported, it is not recommended that this method not be called with floating point input values.

13.11.1.2.49 Method setObject(int, Object, int, int)

```
Synopsis: public void setObject(int parameterIndex, java.lang.Object x,
                                int targetSqlType, int scale)
              throws java.sql.SQLException;
```

Parameters

`parameterIndex` - The first parameter is 1, the second is 2, ...

`x` - The object containing the input parameter value

`targetSqlType` - The SQL type (as defined in java.sql.Types) to be sent to the database. The scale argument may further qualify this type.

`scale` - For java.sql.Types.DECIMAL or java.sql.Types.NUMERIC types this is the number of digits after the decimal. For all other types this value will be ignored,

Exceptions

SQLException if a database-access error occurs.

Additional Information

See Also java.sql.Types

Set the value of a parameter using an object; use the java.lang equivalent objects for integral values.

The given Java object will be converted to the targetSqlType before being sent to the database. If the object is of a class implementing SQLData, the rowset should call its method writeSQL() to write it to the SQL data stream. else If the object is of a class implementing Ref, Blob, Clob, Struct, or Array then pass it to the database as a value of the corresponding SQL type.

Note that this method may be used to pass datatabase- specific abstract data types.

13.11.1.2.50 Method setPassword(String)

```
Synopsis: public void setPassword(java.lang.String s)
                throws java.sql.SQLException;
```

Parameters

`password` - the password string

Exceptions

SQLException if a database-access error occurs.

Set the password.

13.11.1.2.51 Method setQueryTimeout(int)

```
Synopsis: public void setQueryTimeout(int seconds)
                throws java.sql.SQLException;
```

Parameters

`seconds` - the new query timeout limit in seconds; zero means unlimited

Exceptions

SQLException if a database-access error occurs.

The queryTimeout limit is the number of seconds the driver will wait for a Statement to execute. If the limit is exceeded, a SQLException is thrown.

13.11.1.2.52 Method setReadOnly(boolean)

```
Synopsis: public void setReadOnly(boolean value) throws java.sql.SQLException;
```

Parameters

`value` - true if read-only, false otherwise

Exceptions

SQLException if a database-access error occurs.

Set the read-onlyness of the rowset

13.11.1.2.53 Method setRef(int, Ref)

```
Synopsis: public void setRef(int parameterIndex, java.sql.Ref x)
                throws java.sql.SQLException;
```

Parameters

`i` - the first parameter is 1, the second is 2, ...

`x` - an object representing data of an SQL REF Type

Set a REF(<structured-type>) parameter.

13.11.1.2.54 Method setShort(int, short)

```
Synopsis: public void setShort(int parameterIndex, short x)
                throws java.sql.SQLException;
```

Parameters

`parameterIndex` - the first parameter is 1, the second is 2, ...

`x` - the parameter value

Exceptions

SQLException if a database-access error occurs.

Set a parameter to a Java short value.

13.11.1.2.55 Method setString(int, String)

```
Synopsis: public void setString(int parameterIndex, java.lang.String x)
              throws java.sql.SQLException;
```

Parameters

`parameterIndex` - the first parameter is 1, the second is 2, ...

`x` - the parameter value

Exceptions

SQLException if a database-access error occurs.

Set a parameter to a Java String value.

13.11.1.2.56 Method setTime(int, Time)

```
Synopsis: public void setTime(int parameterIndex, java.sql.Time x)
              throws java.sql.SQLException;
```

Parameters

`parameterIndex` - the first parameter is 1, the second is 2, ...

`x` - the parameter value

Exceptions

SQLException if a database-access error occurs.

Set a parameter to a java.sql.Time value.

13.11.1.2.57 Method setTime(int, Time, Calendar)

```
Synopsis: public void setTime(int parameterIndex, java.sql.Time x,
                              java.util.Calendar cal)
              throws java.sql.SQLException;
```

Parameters

`parameterIndex` - the first parameter is 1, the second is 2, ...

`x` - the parameter value

Exceptions

SQLException if a database-access error occurs.

Set a parameter to a java.sql.Time value. The driver converts this to a SQL TIME value when it sends it to the database.

13.11.1.2.58 Method setTimestamp(int, Timestamp)

```
Synopsis: public void setTimestamp(int parameterIndex, java.sql.Timestamp x)
              throws java.sql.SQLException;
```

Parameters

`parameterIndex` - the first parameter is 1, the second is 2, ...

`x` - the parameter value

Exceptions

SQLException if a database-access error occurs.

Set a parameter to a java.sql.Timestamp value.

13.11.1.2.59 Method setTimestamp(int, Timestamp, Calendar)

```
Synopsis: public void setTimestamp(int parameterIndex, java.sql.Timestamp x,
                                   java.util.Calendar cal)
              throws java.sql.SQLException;
```

Parameters

`parameterIndex` - the first parameter is 1, the second is 2, ...

`x` - the parameter value

Exceptions

SQLException if a database-access error occurs.

Set a parameter to a java.sql.Timestamp value. The driver converts this to a SQL TIMESTAMP value when it sends it to the database.

13.11.1.2.60 Method setTransactionIsolation(int)

```
Synopsis: public void setTransactionIsolation(int value)
              throws java.sql.SQLException;
```

Parameters

`level` - the transaction isolation level

Exceptions

SQLException if a database-access error occurs.

Set the transaction isolation.

13.11.1.2.61 Method setType(int)

```
Synopsis: public void setType(int value) throws java.sql.SQLException;
```

Parameters

`value` - may be TYPE_FORWARD_ONLY, TYPE_SCROLL_INSENSITIVE, or TYPE_SCROLL_SENSITIVE

Exceptions

SQLException if a database-access error occurs

Set the type of this result set.

13.11.1.2.62 Method setTypeMap(Map)

```
Synopsis: public void setTypeMap(java.util.Map value)
              throws java.sql.SQLException;
```

Parameters

`map` - a map object

Exceptions

SQLException if a database-access error occurs.

Install a type-map object as the default type-map for this rowset.

13.11.1.2.63 Method setUnicodeStream(int, InputStream, int)

```
Synopsis: public void setUnicodeStream(int parameterIndex,
                                       java.io.InputStream x, int length)
              throws java.sql.SQLException;
```

Parameters

`parameterIndex` - the first parameter is 1, the second is 2, ...

`x` - the java input stream which contains the UNICODE parameter value

`length` - the number of bytes in the stream

Exceptions

SQLException if a database-access error occurs.

13.11.2 Deprecated

When a very large UNICODE value is input to a LONGVARCHAR parameter, it may be more practical to send it via a java.io.InputStream. JDBC will read the data from the stream as needed, until it reaches end-of-file. The JDBC driver will do any necessary conversion from UNICODE to the database char format.

Note: This stream object can either be a standard Java stream object or your own subclass that implements the standard interface.

13.11.2.1 Method setUrl(String)

```
Synopsis: public void setUrl(java.lang.String s) throws java.sql.SQLException;
```

Parameters

`url` - a string value, may be null

Exceptions

SQLException if a database-access error occurs.

Set the url used to create a connection. Setting this property is optional. If a url is used, a JDBC driver that accepts the url must be loaded by the application before the rowset is used to connect to a database. The rowset will use the url internally to create a database connection when reading or writing data. Either a url or a data source name is used to create a connection, whichever was specified most recently.

13.11.2.1.1 Method setUsername(String)

```
Synopsis: public void setUsername(java.lang.String s)
                throws java.sql.SQLException;
```

Parameters

`name` - a user name

Exceptions

SQLException if a database-access error occurs.

Set the user name.

13.11.3 Class OPLCachedRowSet

A OPLCachedRowSet is a disconnected, serializable, scrollable container for tabular data. A primary purpose of the OPLCachedRowSet class is to provide a representation of a JDBC ResultSet that can be passed between different components of a remote application. For example, a OPLCachedRowSet can be used to send the result of a query executed by an Enterprise JavaBeans component running in a server environment over a network to a client running in a web browser. A second use for OPLCachedRowSets is to provide scrolling and updating for ResultSets that don't provide these capabilities themselves. A OPLCachedRowSet can be used to augment the capabilities of a JDBC driver that doesn't have full support for scrolling and updating. Finally, a OPLCachedRowSet can be used to provide Java applications with access to tabular data in an environment such as a thin client or PDA, where it would be inappropriate to use a JDBC driver due to resource limitations or security considerations. The OPLCachedRowSet class provides a means to "get rows in" and "get changed rows out" without the need to implement the full JDBC API.

A OPLCachedRowSet object can contain data retrieved via a JDBC driver or data from some other source, such as a spreadsheet. Both a OPLCachedRowSet object and its metadata can be created from scratch. A component that acts as a factory for rowsets can use this capability to create a rowset containing data from non-JDBC data sources.

The term 'disconnected' implies that a OPLCachedRowSet only makes use of a JDBC connection briefly while data is being read from the database and used to populate it with rows, and again while updated rows are being propagated back to the underlying database. During the remainder of its lifetime, a OPLCachedRowSet object isn't associated with an underlying database connection. A OPLCachedRowSet object can simply be thought of as a disconnected set of rows that are being cached outside of a data source. Since all data is cached in memory, OPLCachedRowSets are not appropriate for extremely large data sets.

The contents of a OPLCachedRowSet may be updated and the updates can be propagated to an underlying data source. OPLCachedRowSets support an optimistic concurrency control mechanism - no locks are maintained in the underlying database during disconnected use of the rowset. Both the original value and current value of the OPLCachedRowSet are maintained for use by the optimistic routines.

13.11.3.1 Synopsis

```
public class OPLCachedRowSet extends BaseRowSet implements RowSetInternal,Serializable,Cloneable {
  // Public Constructors
  public OPLCachedRowSet() throws java.sql.SQLException;
  // Public Methods
  public void finalize() throws java.lang.Throwable;
  public void setCommand(java.lang.String cmd) throws java.sql.SQLException;
  public void setConcurrency(int concurrency) throws java.sql.SQLException;
  public void acceptChanges() throws java.sql.SQLException;
  public void acceptChanges(java.sql.Connection _conn)
      throws java.sql.SQLException;
  public void execute() throws java.sql.SQLException;
  public void execute(java.sql.Connection _conn) throws java.sql.SQLException;
  public void populate(java.sql.ResultSet rs) throws java.sql.SQLException;
  public void setShowDeleted(boolean value) throws java.sql.SQLException;
  public boolean getShowDeleted() throws java.sql.SQLException;
  public String getTableName() throws java.sql.SQLException;
  public void setTableName(java.lang.String _tableName)
      throws java.sql.SQLException;
  public int[] getKeyCols() throws java.sql.SQLException;
  public void setKeyColumns(int[] keys) throws java.sql.SQLException;
  public void cancelRowDelete() throws java.sql.SQLException;
  public void cancelRowInsert() throws java.sql.SQLException;
  public void cancelRowUpdates() throws java.sql.SQLException;
  public boolean columnUpdated(int columnIndex) throws java.sql.SQLException;
  public void setOriginal() throws java.sql.SQLException;
  public void setOriginalRow() throws java.sql.SQLException;
  public void restoreOriginal() throws java.sql.SQLException;
  public int size();
  public Collection toCollection() throws java.sql.SQLException;
  public Collection toCollection(int col) throws java.sql.SQLException;
  public void release() throws java.sql.SQLException;
  public RowSet createCopy() throws java.sql.SQLException;
  public RowSet createShared() throws java.sql.SQLException;
  public void setMetaData(javax.sql.RowSetMetaData md)
      throws java.sql.SQLException;
  public Connection getConnection() throws java.sql.SQLException;
  public ResultSet getOriginal() throws java.sql.SQLException;
  public ResultSet getOriginalRow() throws java.sql.SQLException;
  public void close() throws java.sql.SQLException;
  public boolean next() throws java.sql.SQLException;
  public boolean previous() throws java.sql.SQLException;
  public boolean first() throws java.sql.SQLException;
  public boolean last() throws java.sql.SQLException;
  public boolean absolute(int row) throws java.sql.SQLException;
  public boolean relative(int rows) throws java.sql.SQLException;
  public void beforeFirst() throws java.sql.SQLException;
  public void afterLast() throws java.sql.SQLException;
  public boolean isBeforeFirst() throws java.sql.SQLException;
  public boolean isAfterLast() throws java.sql.SQLException;
  public boolean isFirst() throws java.sql.SQLException;
  public boolean isLast() throws java.sql.SQLException;
  public int getRow() throws java.sql.SQLException;
  public boolean rowUpdated() throws java.sql.SQLException;
  public boolean rowInserted() throws java.sql.SQLException;
  public boolean rowDeleted() throws java.sql.SQLException;
  public void refreshRow() throws java.sql.SQLException;
  public void insertRow() throws java.sql.SQLException;
  public void updateRow() throws java.sql.SQLException;
  public void deleteRow() throws java.sql.SQLException;
  public void moveToInsertRow() throws java.sql.SQLException;
  public void moveToCurrentRow() throws java.sql.SQLException;
  public boolean wasNull() throws java.sql.SQLException;
  public SQLWarning getWarnings() throws java.sql.SQLException;
  public void clearWarnings() throws java.sql.SQLException;
  public String getCursorName() throws java.sql.SQLException;
public ResultSetMetaData getMetaData() throws java.sql.SQLException;
public int findColumn(java.lang.String columnName)
    throws java.sql.SQLException;
public String getString(int columnIndex) throws java.sql.SQLException;
public boolean getBoolean(int columnIndex) throws java.sql.SQLException;
public byte getByte(int columnIndex) throws java.sql.SQLException;
public short getShort(int columnIndex) throws java.sql.SQLException;
public int getInt(int columnIndex) throws java.sql.SQLException;
public long getLong(int columnIndex) throws java.sql.SQLException;
public float getFloat(int columnIndex) throws java.sql.SQLException;
public double getDouble(int columnIndex) throws java.sql.SQLException;
public BigDecimal getBigDecimal(int columnIndex)
    throws java.sql.SQLException;
public BigDecimal getBigDecimal(int columnIndex, int scale)
    throws java.sql.SQLException;
public byte[] getBytes(int columnIndex) throws java.sql.SQLException;
public Date getDate(int columnIndex) throws java.sql.SQLException;
public Time getTime(int columnIndex) throws java.sql.SQLException;
public Timestamp getTimestamp(int columnIndex) throws java.sql.SQLException;
public InputStream getAsciiStream(int columnIndex)
    throws java.sql.SQLException;
public InputStream getUnicodeStream(int columnIndex)
    throws java.sql.SQLException;
public InputStream getBinaryStream(int columnIndex)
    throws java.sql.SQLException;
public Object getObject(int columnIndex) throws java.sql.SQLException;
public String getString(java.lang.String columnName)
    throws java.sql.SQLException;
public boolean getBoolean(java.lang.String columnName)
    throws java.sql.SQLException;
public byte getByte(java.lang.String columnName)
    throws java.sql.SQLException;
public short getShort(java.lang.String columnName)
    throws java.sql.SQLException;
public int getInt(java.lang.String columnName) throws java.sql.SQLException;
public long getLong(java.lang.String columnName)
    throws java.sql.SQLException;
public float getFloat(java.lang.String columnName)
    throws java.sql.SQLException;
public double getDouble(java.lang.String columnName)
    throws java.sql.SQLException;
public BigDecimal getBigDecimal(java.lang.String columnName, int scale)
    throws java.sql.SQLException;
public byte[] getBytes(java.lang.String columnName)
    throws java.sql.SQLException;
public Date getDate(java.lang.String columnName)
    throws java.sql.SQLException;
public Time getTime(java.lang.String columnName)
    throws java.sql.SQLException;
public Timestamp getTimestamp(java.lang.String columnName)
    throws java.sql.SQLException;
public InputStream getAsciiStream(java.lang.String columnName)
    throws java.sql.SQLException;
public InputStream getUnicodeStream(java.lang.String columnName)
    throws java.sql.SQLException;
public InputStream getBinaryStream(java.lang.String columnName)
    throws java.sql.SQLException;
public Object getObject(java.lang.String columnName)
    throws java.sql.SQLException;
public Reader getCharacterStream(int columnIndex)
    throws java.sql.SQLException;
public Reader getCharacterStream(java.lang.String columnName)
    throws java.sql.SQLException;
public BigDecimal getBigDecimal(java.lang.String columnName)
    throws java.sql.SQLException;
public void updateNull(int columnIndex) throws java.sql.SQLException;
public void updateBoolean(int columnIndex, boolean x)
    throws java.sql.SQLException;
public void updateByte(int columnIndex, byte x) throws java.sql.SQLException;
public void updateShort(int columnIndex, short x)
    throws java.sql.SQLException;
public void updateInt(int columnIndex, int x) throws java.sql.SQLException;
public void updateLong(int columnIndex, long x) throws java.sql.SQLException;
public void updateFloat(int columnIndex, float x)
    throws java.sql.SQLException;
public void updateDouble(int columnIndex, double x)
    throws java.sql.SQLException;
public void updateBigDecimal(int columnIndex, java.math.BigDecimal x)
    throws java.sql.SQLException;
public void updateString(int columnIndex, java.lang.String x)
    throws java.sql.SQLException;
public void updateBytes(int columnIndex, byte[] x)
    throws java.sql.SQLException;
public void updateDate(int columnIndex, java.sql.Date x)
    throws java.sql.SQLException;
public void updateTime(int columnIndex, java.sql.Time x)
    throws java.sql.SQLException;
public void updateTimestamp(int columnIndex, java.sql.Timestamp x)
    throws java.sql.SQLException;
public void updateAsciiStream(int columnIndex, java.io.InputStream x,
                              int length) throws java.sql.SQLException;
public void updateBinaryStream(int columnIndex, java.io.InputStream x,
                               int length) throws java.sql.SQLException;
public void updateCharacterStream(int columnIndex, java.io.Reader x,
                                  int length) throws java.sql.SQLException;
public void updateObject(int columnIndex, java.lang.Object x, int scale)
    throws java.sql.SQLException;
public void updateObject(int columnIndex, java.lang.Object x)
    throws java.sql.SQLException;
public void updateNull(java.lang.String columnName)
    throws java.sql.SQLException;
public void updateBoolean(java.lang.String columnName, boolean x)
    throws java.sql.SQLException;
public void updateByte(java.lang.String columnName, byte x)
    throws java.sql.SQLException;
public void updateShort(java.lang.String columnName, short x)
    throws java.sql.SQLException;
public void updateInt(java.lang.String columnName, int x)
    throws java.sql.SQLException;
public void updateLong(java.lang.String columnName, long x)
    throws java.sql.SQLException;
public void updateFloat(java.lang.String columnName, float x)
    throws java.sql.SQLException;
public void updateDouble(java.lang.String columnName, double x)
    throws java.sql.SQLException;
public void updateBigDecimal(java.lang.String columnName,
                             java.math.BigDecimal x)
    throws java.sql.SQLException;
public void updateString(java.lang.String columnName, java.lang.String x)
    throws java.sql.SQLException;
public void updateBytes(java.lang.String columnName, byte[] x)
    throws java.sql.SQLException;
public void updateDate(java.lang.String columnName, java.sql.Date x)
    throws java.sql.SQLException;
public void updateTime(java.lang.String columnName, java.sql.Time x)
    throws java.sql.SQLException;
public void updateTimestamp(java.lang.String columnName,
                            java.sql.Timestamp x)
    throws java.sql.SQLException;
public void updateAsciiStream(java.lang.String columnName,
                              java.io.InputStream x, int length)
    throws java.sql.SQLException;
public void updateBinaryStream(java.lang.String columnName,
                               java.io.InputStream x, int length)
    throws java.sql.SQLException;
public void updateCharacterStream(java.lang.String columnName,
                                  java.io.Reader reader, int length)
    throws java.sql.SQLException;
public void updateObject(java.lang.String columnName, java.lang.Object x,
                         int scale) throws java.sql.SQLException;
public void updateObject(java.lang.String columnName, java.lang.Object x)
    throws java.sql.SQLException;
public Statement getStatement() throws java.sql.SQLException;
public Object getObject(int colIndex, java.util.Map map)
    throws java.sql.SQLException;
public Ref getRef(int colIndex) throws java.sql.SQLException;
```

```
        public Blob getBlob(int colIndex) throws java.sql.SQLException;
        public Clob getClob(int colIndex) throws java.sql.SQLException;
        public Array getArray(int colIndex) throws java.sql.SQLException;
        public Object getObject(java.lang.String colName, java.util.Map map)
            throws java.sql.SQLException;
        public Ref getRef(java.lang.String colName) throws java.sql.SQLException;
        public Blob getBlob(java.lang.String colName) throws java.sql.SQLException;
        public Clob getClob(java.lang.String colName) throws java.sql.SQLException;
        public Array getArray(java.lang.String colName) throws java.sql.SQLException;
        public Date getDate(int columnIndex, java.util.Calendar cal)
            throws java.sql.SQLException;
        public Date getDate(java.lang.String columnName, java.util.Calendar cal)
            throws java.sql.SQLException;
        public Time getTime(int columnIndex, java.util.Calendar cal)
            throws java.sql.SQLException;
        public Time getTime(java.lang.String columnName, java.util.Calendar cal)
            throws java.sql.SQLException;
        public Timestamp getTimestamp(int columnIndex, java.util.Calendar cal)
            throws java.sql.SQLException;
        public Timestamp getTimestamp(java.lang.String columnName,
                                      java.util.Calendar cal)
            throws java.sql.SQLException;
        public URL getURL(int columnIndex) throws java.sql.SQLException;
        public URL getURL(java.lang.String columnName) throws java.sql.SQLException;
        public void updateRef(int columnIndex, java.sql.Ref x)
            throws java.sql.SQLException;
        public void updateRef(java.lang.String columnName, java.sql.Ref x)
            throws java.sql.SQLException;
        public void updateBlob(int columnIndex, java.sql.Blob x)
            throws java.sql.SQLException;
        public void updateBlob(java.lang.String columnName, java.sql.Blob x)
            throws java.sql.SQLException;
        public void updateClob(int columnIndex, java.sql.Clob x)
            throws java.sql.SQLException;
        public void updateClob(java.lang.String columnName, java.sql.Clob x)
            throws java.sql.SQLException;
        public void updateArray(int columnIndex, java.sql.Array x)
            throws java.sql.SQLException;
        public void updateArray(java.lang.String columnName, java.sql.Array x)
            throws java.sql.SQLException;
      }
```

Inheritance Path

java.lang.Object

|

openlink.javax.BaseRowSet

|

openlink.javax.OPLCachedRowSet

13.11.3.2 Members

13.11.3.2.1 Constructor OPLCachedRowSet()

Synopsis: `public OPLCachedRowSet() throws java.sql.SQLException;`

Exceptions

SQLException	if an error occurs.

Create a OPLCachedRowSet object. The object has no metadata.

13.11.3.2.2 Method absolute(int)

Synopsis: `public boolean absolute(int row) throws java.sql.SQLException;`

Exceptions

SQLException	if a database-access error occurs, or row is 0, or rowset type is TYPE_FORWARD_ONLY.

Move to an absolute row number in the rowset. It notifies listeners that the cursor has moved.

If row is positive, moves to an absolute row with respect to the beginning of the rowset. The first row is row 1, the second is row 2, etc.

If row is negative, moves to an absolute row position with respect to the end of rowset. For example, calling absolute(-1) positions the cursor on the last row, absolute(-2) indicates the next-to-last row, etc.

An attempt to position the cursor beyond the first/last row in the rowset, leaves the cursor before/after the first/last row, respectively.

Note: Calling absolute(1) is the same as calling first(). Calling absolute(-1) is the same as calling last().

13.11.3.2.3 Method acceptChanges()

```
Synopsis: public void acceptChanges() throws java.sql.SQLException;
```

Exceptions

SQLException if an error occurs.

Propagate all row update, insert, and delete changes to a data source. An SQLException is thrown if any of the updates cannot be propagated. If acceptChanges() fails then the caller can assume that none of the updates are reflected in the data source. The current row is set to the first "updated" row that resulted in an exception, in the case that an exception is thrown. With one exception, if the row that caused the exception is a "deleted" row, then in the usual case, when deleted rows are not shown, the current row isn't affected. When successful, calling acceptChanges() replaces the original value of the rowset with the current value. Note: Both the original and current value of the rowset are maintained. The original state is the value of the rowset following its creation (i.e. empty), or following the last call to acceptChanges(), execute(), populate(), release(), or restoreOriginal(). Internally, a RowSetWriter component is envoked to write the data for each row.

13.11.3.2.4 Method acceptChanges(Connection)

```
Synopsis: public void acceptChanges(java.sql.Connection _conn)
                 throws java.sql.SQLException;
```

Parameters

_conn - a database connection

Exceptions

SQLException if an error occurs.

Like acceptChanges() above, but takes a Connection argument. The Connection is used internally when doing the updates. This form isn't used unless the underlying data source is a JDBC data source.

13.11.3.2.5 Method afterLast()

```
Synopsis: public void afterLast() throws java.sql.SQLException;
```

Exceptions

SQLException if a database-access error occurs, or rowset type is TYPE_FORWARD_ONLY.

Moves to the end of the rowset, just after the last row. Has no effect if the rowset contains no rows. It notifies listeners that the cursor has moved.

13.11.3.2.6 Method beforeFirst()

```
Synopsis: public void beforeFirst() throws java.sql.SQLException;
```

Exceptions

SQLException if a database-access error occurs, or rowset type is TYPE_FORWARD_ONLY

Moves to the front of the rowset, just before the first row. Has no effect if the rowset contains no rows. It notifies listeners that the cursor has moved.

13.11.3.2.7 Method cancelRowDelete()

```
Synopsis: public void cancelRowDelete() throws java.sql.SQLException;
```

Exceptions

SQLException	if an error occurs.

Cancels deletion of the current row and notifies listeners that a row has changed. After calling cancelRowDelete() the current row is no longer marked for deletion. An exception is thrown if there is no current row. Note: This method can be ignored if deleted rows aren't being shown (the normal case).

13.11.3.2.8 Method cancelRowInsert()

```
Synopsis: public void cancelRowInsert() throws java.sql.SQLException;
```

Exceptions

SQLException	if an error occurs.

Cancels insertion of the current row and notifies listeners that a row has changed. An exception is thrown if the row isn't an inserted row. The current row is immediately removed from the rowset. This operation cannot be undone.

13.11.3.2.9 Method cancelRowUpdates()

```
Synopsis: public void cancelRowUpdates() throws java.sql.SQLException;
```

Exceptions

SQLException	if an error occurs.

The cancelRowUpdates() method may be called after calling an updateXXX() method(s) and before calling updateRow() to rollback the updates made to a row. If no updates have been made or updateRow() has already been called, then this method has no effect. It notifies listeners that a row has changed, if it has effect.

13.11.3.2.10 Method clearWarnings()

```
Synopsis: public void clearWarnings() throws java.sql.SQLException;
```

Exceptions

SQLException	if a database-access error occurs.

After this call getWarnings returns null until a new warning is reported for this ResultSet.

13.11.3.2.11 Method close()

```
Synopsis: public void close() throws java.sql.SQLException;
```

Exceptions

SQLException	if an error occurs.

Releases the current contents of this rowset, discarding outstanding updates. The rowset contains no rows after the method release is called. This method sends a RowSetChangedEvent object to all registered listeners prior to returning.

13.11.3.2.12 Method columnUpdated(int)

```
Synopsis: public boolean columnUpdated(int columnIndex)
              throws java.sql.SQLException;
```

Parameters

Parameters

`columnIndex` - the first column is 1, the second is 2, ...

`return` - true if the column has been updated

Exceptions

SQLException if a database-access error occurs

Determine if the column from the current row has been updated.

13.11.3.2.13 Method createCopy()

```
Synopsis: public RowSet createCopy() throws java.sql.SQLException;
```

Exceptions

SQLException if an error occurs.

Creates a RowSet object that is a deep copy of this OPLCachedRowSet object's data. Updates made on a copy are not visible to the original rowset; a copy of a rowset is completely independent from the original. Making a copy saves the cost of creating an identical rowset from first principles, which can be quite expensive. For example, it doesn't do the query to a remote database server.

13.11.3.2.14 Method createShared()

```
Synopsis: public RowSet createShared() throws java.sql.SQLException;
```

Exceptions

SQLException if an error occurs.

Returns a new rowset object backed by the same data. Updates made by a shared duplicate are visible to the original rowset and other duplicates. A rowset and its duplicates form a set of cursors that iterate over a shared set of rows, providing different views of the underlying data. Duplicates also share property values. So, for example, if a rowset is read-only then all of its duplicates will be read-only.

13.11.3.2.15 Method deleteRow()

```
Synopsis: public void deleteRow() throws java.sql.SQLException;
```

Exceptions

SQLException if a database-access error occurs, or if called when on the insert row.

Delete the current row from this OPLCachedRowSet object and it notifies listeners that a row has changed. Cannot be called when the cursor is on the insert row. The method marks the current row as deleted, but it does not delete the row from the underlying data source. The method acceptChanges must be called to delete the row in the data source.

13.11.3.2.16 Method execute()

```
Synopsis: public void execute() throws java.sql.SQLException;
```

Exceptions

SQLException if an error occurs.

Populates this OPLCachedRowSet object with data. This form of the method uses the rowset's user, password, and url or data source name properties to create a database connection. If properties that are needed have not been set, this method will throw an exception. Another form of this method uses an existing JDBC Connection object instead of creating a new one; therefore, it ignores the properties used for establishing a new connection. The query specified by the command property is executed to create a ResultSet object from which to retrieve data. The current contents of the rowset are discarded, and the rowset's metadata is also (re)set. If there are outstanding updates, they are also ignored. The method execute closes any database connections that it creates.

13.11.3.2.17 Method execute(Connection)

```
Synopsis: public void execute(java.sql.Connection _conn)
              throws java.sql.SQLException;
```

Parameters

`_conn` - a database connection

Exceptions

SQLException if an error occurs.

Populates the rowset with data. The first form uses the properties: url/data source name, user, and password to create a database connection. If these properties haven't been set, an exception is thrown. The second form uses an existing JDBC connection object instead. The values of the url/data source name, user, and password properties are ignored when the second form is used. Execute() closes any database connections that it creates. The command specified by the command property is executed to retrieve the data. The current contents of the rowset are discarded and the rowset's metadata is also (re)set. If there are outstanding updates, they are also ignored.

13.11.3.2.18 Method findColumn(String)

```
Synopsis: public int findColumn(java.lang.String columnName)
              throws java.sql.SQLException;
```

Parameters

`columnName` - the name of the column

`return` - the column index

Exceptions

SQLException if a database-access error occurs.

Map a Resultset column name to a ResultSet column index.

13.11.3.2.19 Method first()

```
Synopsis: public boolean first() throws java.sql.SQLException;
```

Exceptions

SQLException if a database-access error occurs, or rowset type is TYPE_FORWARD_ONLY.

Moves to the first row in the rowset. It notifies listeners that the cursor has moved.

13.11.3.2.20 Method getArray(int)

```
Synopsis: public Array getArray(int colIndex) throws java.sql.SQLException;
```

Parameters

`colIndex` - the first column is 1, the second is 2, ...

`return` - an object representing an SQL array

Get an array column.

13.11.3.2.21 Method getArray(String)

```
Synopsis: public Array getArray(java.lang.String colName)
              throws java.sql.SQLException;
```

Parameters

`colName` - the column name

`return` - an object representing an SQL array

Get an array column.

13.11.3.2.22 Method getAsciiStream(int)

```
Synopsis: public InputStream getAsciiStream(int columnIndex)
              throws java.sql.SQLException;
```

Parameters

`columnIndex` - the first column is 1, the second is 2, ...

`return` - a Java input stream that delivers the database column value as a stream of one byte ASCII characters. If the value is SQL NULL then the result is null.

Exceptions

SQLException if a database-access error occurs.

A column value can be retrieved as a stream of ASCII characters and then read in chunks from the stream. This method is particularly suitable for retrieving large LONGVARCHAR values. The JDBC driver will do any necessary conversion from the database format into ASCII.

Note: All the data in the returned stream must be read prior to getting the value of any other column. The next call to a get method implicitly closes the stream. . Also, a stream may return 0 for available() whether there is data available or not.

13.11.3.2.23 Method getAsciiStream(String)

```
Synopsis: public InputStream getAsciiStream(java.lang.String columnName)
              throws java.sql.SQLException;
```

Parameters

`columnName` - is the SQL name of the column

`return` - a Java input stream that delivers the database column value as a stream of one byte ASCII characters. If the value is SQL NULL then the result is null.

Exceptions

SQLException if a database-access error occurs.

A column value can be retrieved as a stream of ASCII characters and then read in chunks from the stream. This method is particularly suitable for retrieving large LONGVARCHAR values. The JDBC driver will do any necessary conversion from the database format into ASCII.

Note: All the data in the returned stream must be read prior to getting the value of any other column. The next call to a get method implicitly closes the stream.

13.11.3.2.24 Method getBigDecimal(int)

```
Synopsis: public BigDecimal getBigDecimal(int columnIndex)
              throws java.sql.SQLException;
```

Parameters

`columnIndex` - the first column is 1, the second is 2, ...

`return` - the column value (full precision); if the value is SQL NULL, the result is null

Exceptions

SQLException if a database-access error occurs.

Get the value of a column in the current row as a java.math.BigDecimal object.

13.11.3.2.25 Method getBigDecimal(int, int)

```
Synopsis: public BigDecimal getBigDecimal(int columnIndex, int scale)
              throws java.sql.SQLException;
```

Parameters

`columnIndex` - the first column is 1, the second is 2, ...

Parameters

`scale` - the number of digits to the right of the decimal

`return` - the column value; if the value is SQL NULL, the result is null

Exceptions

SQLException if a database-access error occurs.

13.11.4 Deprecated

Get the value of a column in the current row as a java.math.BigDecimal object.

13.11.4.1 Method getBigDecimal(String)

```
Synopsis: public BigDecimal getBigDecimal(java.lang.String columnName)
                throws java.sql.SQLException;
```

Get the value of a column in the current row as a java.math.BigDecimal object.

13.11.4.1.1 Method getBigDecimal(String, int)

```
Synopsis: public BigDecimal getBigDecimal(java.lang.String columnName,
                                          int scale)
                throws java.sql.SQLException;
```

Parameters

`columnName` - is the SQL name of the column

`scale` - the number of digits to the right of the decimal

`return` - the column value; if the value is SQL NULL, the result is null

Exceptions

SQLException if a database-access error occurs.

13.11.5 Deprecated

Get the value of a column in the current row as a java.math.BigDecimal object.

13.11.5.1 Method getBinaryStream(int)

```
Synopsis: public InputStream getBinaryStream(int columnIndex)
                throws java.sql.SQLException;
```

Parameters

`columnIndex` - the first column is 1, the second is 2, ...

`return` - a Java input stream that delivers the database column value as a stream of uninterpreted bytes. If the value is SQL NULL then the result is null.

Exceptions

SQLException if a database-access error occurs.

A column value can be retrieved as a stream of uninterpreted bytes and then read in chunks from the stream. This method is particularly suitable for retrieving large LONGVARBINARY values.

Note: All the data in the returned stream must be read prior to getting the value of any other column. The next call to a get method implicitly closes the stream. Also, a stream may return 0 for available() whether there is data available or not.

13.11.5.1.1 Method getBinaryStream(String)

```
Synopsis: public InputStream getBinaryStream(java.lang.String columnName)
                throws java.sql.SQLException;
```

Parameters

Parameters

`columnName` - is the SQL name of the column

`return` - a Java input stream that delivers the database column value as a stream of uninterpreted bytes. If the value is SQL NULL then the result is null.

Exceptions

SQLException if a database-access error occurs.

A column value can be retrieved as a stream of uninterpreted bytes and then read in chunks from the stream. This method is particularly suitable for retrieving large LONGVARBINARY values.

Note: All the data in the returned stream must be read prior to getting the value of any other column. The next call to a get method implicitly closes the stream.

13.11.5.1.2 Method getBlob(int)

```
Synopsis: public Blob getBlob(int colIndex) throws java.sql.SQLException;
```

Parameters

`colIndex` - the first column is 1, the second is 2, ...

`return` - an object representing a BLOB

Get a BLOB column.

13.11.5.1.3 Method getBlob(String)

```
Synopsis: public Blob getBlob(java.lang.String colName)
                throws java.sql.SQLException;
```

Parameters

`colName` - the column name

`return` - an object representing a BLOB

Get a BLOB column.

13.11.5.1.4 Method getBoolean(int)

```
Synopsis: public boolean getBoolean(int columnIndex)
                throws java.sql.SQLException;
```

Parameters

`columnIndex` - the first column is 1, the second is 2, ...

`return` - the column value; if the value is SQL NULL, the result is false

Exceptions

SQLException if a database-access error occurs.

Get the value of a column in the current row as a Java boolean.

13.11.5.1.5 Method getBoolean(String)

```
Synopsis: public boolean getBoolean(java.lang.String columnName)
                throws java.sql.SQLException;
```

Parameters

`columnName` - is the SQL name of the column

`return` - the column value; if the value is SQL NULL, the result is false

Exceptions

SQLException if a database-access error occurs.

Get the value of a column in the current row as a Java boolean.

13.11.5.1.6 Method getByte(int)

```
Synopsis: public byte getByte(int columnIndex) throws java.sql.SQLException;
```

Parameters

`columnIndex` - the first column is 1, the second is 2, ...

`return` - the column value; if the value is SQL NULL, the result is 0

Exceptions

SQLException if a database-access error occurs.

Get the value of a column in the current row as a Java byte.

13.11.5.1.7 Method getByte(String)

```
Synopsis: public byte getByte(java.lang.String columnName)
              throws java.sql.SQLException;
```

Parameters

`columnName` - is the SQL name of the column

`return` - the column value; if the value is SQL NULL, the result is 0

Exceptions

SQLException if a database-access error occurs.

Get the value of a column in the current row as a Java byte.

13.11.5.1.8 Method getBytes(int)

```
Synopsis: public byte[] getBytes(int columnIndex) throws java.sql.SQLException;
```

Parameters

`columnIndex` - the first column is 1, the second is 2, ...

`return` - the column value; if the value is SQL NULL, the result is null

Exceptions

SQLException if a database-access error occurs.

Get the value of a column in the current row as a Java byte array. The bytes represent the raw values returned by the driver.

13.11.5.1.9 Method getBytes(String)

```
Synopsis: public byte[] getBytes(java.lang.String columnName)
              throws java.sql.SQLException;
```

Parameters

`columnName` - is the SQL name of the column

`return` - the column value; if the value is SQL NULL, the result is null

Exceptions

SQLException if a database-access error occurs.

Get the value of a column in the current row as a Java byte array. The bytes represent the raw values returned by the driver.

13.11.5.1.10 Method getCharacterStream(int)

```
Synopsis: public Reader getCharacterStream(int columnIndex)
              throws java.sql.SQLException;
```

Get the value of a column in the current row as a java.io.Reader.

13.11.5.1.11 Method getCharacterStream(String)

```
Synopsis: public Reader getCharacterStream(java.lang.String columnName)
                throws java.sql.SQLException;
```

Get the value of a column in the current row as a java.io.Reader.

13.11.5.1.12 Method getClob(int)

```
Synopsis: public Clob getClob(int colIndex) throws java.sql.SQLException;
```

Parameters

`colIndex` - the first column is 1, the second is 2, ...

`return` - an object representing a CLOB

Get a CLOB column.

13.11.5.1.13 Method getClob(String)

```
Synopsis: public Clob getClob(java.lang.String colName)
                throws java.sql.SQLException;
```

Parameters

`colName` - the column name

`return` - an object representing a CLOB

Get a CLOB column.

13.11.5.1.14 Method getConnection()

```
Synopsis: public Connection getConnection() throws java.sql.SQLException;
```

Exceptions

SQLException if a database-access error occurs.

Get the Connection passed to the rowset.

13.11.5.1.15 Method getCursorName()

```
Synopsis: public String getCursorName() throws java.sql.SQLException;
```

Exceptions

SQLException if an error occurs.

Get the name of the SQL cursor used by this ResultSet.

13.11.5.1.16 Method getDate(int)

```
Synopsis: public Date getDate(int columnIndex) throws java.sql.SQLException;
```

Parameters

`columnIndex` - the first column is 1, the second is 2, ...

`return` - the column value; if the value is SQL NULL, the result is null

Exceptions

SQLException if a database-access error occurs.

Get the value of a column in the current row as a java.sql.Date object.

13.11.5.1.17 Method getDate(int, Calendar)

```
Synopsis: public Date getDate(int columnIndex, java.util.Calendar cal)
              throws java.sql.SQLException;
```

Parameters

`columnIndex` - the first column is 1, the second is 2, ...

`cal` - the calendar to use in constructing the date

`return` - the column value; if the value is SQL NULL, the result is null

Exceptions

SQLException if a database-access error occurs.

Get the value of a column in the current row as a java.sql.Date object. Use the calendar to construct an appropriate millisecond value for the Date, if the underlying database doesn't store timezone information.

13.11.5.1.18 Method getDate(String)

```
Synopsis: public Date getDate(java.lang.String columnName)
              throws java.sql.SQLException;
```

Parameters

`columnName` - is the SQL name of the column

`return` - the column value; if the value is SQL NULL, the result is null

Exceptions

SQLException if a database-access error occurs.

Get the value of a column in the current row as a java.sql.Date object.

13.11.5.1.19 Method getDate(String, Calendar)

```
Synopsis: public Date getDate(java.lang.String columnName,
                              java.util.Calendar cal)
              throws java.sql.SQLException;
```

Parameters

`columnName` - is the SQL name of the column

`cal` - the calendar to use in constructing the date

`return` - the column value; if the value is SQL NULL, the result is null

Exceptions

SQLException if a database-access error occurs.

Get the value of a column in the current row as a java.sql.Date object. Use the calendar to construct an appropriate millisecond value for the Date, if the underlying database doesn't store timezone information.

13.11.5.1.20 Method getDouble(int)

```
Synopsis: public double getDouble(int columnIndex)
              throws java.sql.SQLException;
```

Parameters

`columnIndex` - the first column is 1, the second is 2, ...

`return` - the column value; if the value is SQL NULL, the result is 0

Exceptions

SQLException if a database-access error occurs.

Get the value of a column in the current row as a Java double.

13.11.5.1.21 Method getDouble(String)

```
Synopsis: public double getDouble(java.lang.String columnName)
              throws java.sql.SQLException;
```

Parameters

`columnName` - is the SQL name of the column

`return` - the column value; if the value is SQL NULL, the result is 0

Exceptions

SQLException if a database-access error occurs.

Get the value of a column in the current row as a Java double.

13.11.5.1.22 Method getFloat(int)

```
Synopsis: public float getFloat(int columnIndex) throws java.sql.SQLException;
```

Parameters

`columnIndex` - the first column is 1, the second is 2, ...

`return` - the column value; if the value is SQL NULL, the result is 0

Exceptions

SQLException if a database-access error occurs.

Get the value of a column in the current row as a Java float.

13.11.5.1.23 Method getFloat(String)

```
Synopsis: public float getFloat(java.lang.String columnName)
              throws java.sql.SQLException;
```

Parameters

`columnName` - is the SQL name of the column

`return` - the column value; if the value is SQL NULL, the result is 0

Exceptions

SQLException if a database-access error occurs.

Get the value of a column in the current row as a Java float.

13.11.5.1.24 Method getInt(int)

```
Synopsis: public int getInt(int columnIndex) throws java.sql.SQLException;
```

Parameters

`columnIndex` - the first column is 1, the second is 2, ...

`return` - the column value; if the value is SQL NULL, the result is 0

Exceptions

SQLException if a database-access error occurs.

Get the value of a column in the current row as a Java int.

13.11.5.1.25 Method getInt(String)

```
Synopsis: public int getInt(java.lang.String columnName)
              throws java.sql.SQLException;
```

Parameters

`columnName` - is the SQL name of the column

`return` - the column value; if the value is SQL NULL, the result is 0

Exceptions

SQLException if a database-access error occurs.

Get the value of a column in the current row as a Java int.

13.11.5.1.26 Method getKeyCols()

```
Synopsis: public int[] getKeyCols() throws java.sql.SQLException;
```

Exceptions

SQLException if an error occurs.

Returns the columns that make a key to uniquely identify a row in this OPLCachedRowSet object.

13.11.5.1.27 Method getLong(int)

```
Synopsis: public long getLong(int columnIndex) throws java.sql.SQLException;
```

Parameters

`columnIndex` - the first column is 1, the second is 2, ...

`return` - the column value; if the value is SQL NULL, the result is 0

Exceptions

SQLException if a database-access error occurs.

Get the value of a column in the current row as a Java long.

13.11.5.1.28 Method getLong(String)

```
Synopsis: public long getLong(java.lang.String columnName)
                  throws java.sql.SQLException;
```

Parameters

`columnName` - is the SQL name of the column

`return` - the column value; if the value is SQL NULL, the result is 0

Exceptions

SQLException if a database-access error occurs.

Get the value of a column in the current row as a Java long.

13.11.5.1.29 Method getMetaData()

```
Synopsis: public ResultSetMetaData getMetaData() throws java.sql.SQLException;
```

Exceptions

SQLException if a database-access error occurs.

The number, types and properties of a ResultSet's columns are provided by the getMetaData method.

13.11.5.1.30 Method getObject(int)

```
Synopsis: public Object getObject(int columnIndex)
                  throws java.sql.SQLException;
```

Parameters

`columnIndex` - the first column is 1, the second is 2, ...

`return` - a java.lang.Object holding the column value.

Exceptions

SQLException if a database-access error occurs.

Get the value of a column in the current row as a Java object.

This method will return the value of the given column as a Java object. The type of the Java object will be the default Java object type corresponding to the column's SQL type, following the mapping for built-in types specified in the JDBC spec.

This method may also be used to read database specific abstract data types. JDBC 2.0 New behavior for getObject(). The behavior of method getObject() is extended to materialize data of SQL user-defined types. When the column @column is a structured or distinct value, the behavior of this method is as if it were a call to: getObject(column, this.getStatement().getConnection().getTypeMap()).

13.11.5.1.31 Method getObject(int, Map)

```
Synopsis: public Object getObject(int colIndex, java.util.Map map)
                throws java.sql.SQLException;
```

Parameters

`colIndex` - the first column is 1, the second is 2, ...

`map` - the mapping from SQL type names to Java classes

`return` - an object representing the SQL value

Returns the value of column @i as a Java object. Use the map to determine the class from which to construct data of SQL structured and distinct types.

13.11.5.1.32 Method getObject(String)

```
Synopsis: public Object getObject(java.lang.String columnName)
                throws java.sql.SQLException;
```

Parameters

`columnName` - is the SQL name of the column

`return` - a java.lang.Object holding the column value.

Exceptions

SQLException if a database-access error occurs.

Get the value of a column in the current row as a Java object.

This method will return the value of the given column as a Java object. The type of the Java object will be the default Java object type corresponding to the column's SQL type, following the mapping for built-in types specified in the JDBC spec.

This method may also be used to read database specific abstract data types. JDBC 2.0 New behavior for getObject(). The behavior of method getObject() is extended to materialize data of SQL user-defined types. When the column @columnName is a structured or distinct value, the behavior of this method is as if it were a call to: getObject(columnName, this.getStatement().getConnection().getTypeMap()).

13.11.5.1.33 Method getObject(String, Map)

```
Synopsis: public Object getObject(java.lang.String colName, java.util.Map map)
                throws java.sql.SQLException;
```

Parameters

`colName` - the column name

`map` - the mapping from SQL type names to Java classes

`return` - an object representing the SQL value

Returns the value of column @i as a Java object. Use the map to determine the class from which to construct data of SQL structured and distinct types.

13.11.5.1.34 Method getOriginal()

```
Synopsis: public ResultSet getOriginal() throws java.sql.SQLException;
```

Exceptions

SQLException if a database-access error occurs.

Returns a result set containing the original value of the rowset. The cursor is positioned before the first row in the result set. Only rows contained in the result set returned by getOriginal() are said to have an original value.

13.11.5.1.35 Method getOriginalRow()

```
Synopsis: public ResultSet getOriginalRow() throws java.sql.SQLException;
```

Exceptions

SQLException if a database-access error occurs.

Returns a result set containing the original value of the current row only. If the current row has no original value an empty result set is returned. If there is no current row an exception is thrown.

13.11.5.1.36 Method getRef(int)

```
Synopsis: public Ref getRef(int colIndex) throws java.sql.SQLException;
```

Parameters

`colIndex` - the first column is 1, the second is 2, ...

`return` - an object representing data of an SQL REF type

Get a REF(<structured-type>) column.

13.11.5.1.37 Method getRef(String)

```
Synopsis: public Ref getRef(java.lang.String colName)
                throws java.sql.SQLException;
```

Parameters

`colName` - the column name

`return` - an object representing data of an SQL REF type

Get a REF(<structured-type>) column.

13.11.5.1.38 Method getRow()

```
Synopsis: public int getRow() throws java.sql.SQLException;
```

Exceptions

SQLException if a database-access error occurs.

Determine the current row number. The first row is number 1, the second number 2, etc.

13.11.5.1.39 Method getShort(int)

```
Synopsis: public short getShort(int columnIndex) throws java.sql.SQLException;
```

Parameters

`columnIndex` - the first column is 1, the second is 2, ...

`return` - the column value; if the value is SQL NULL, the result is 0

Exceptions

SQLException if a database-access error occurs.

Get the value of a column in the current row as a Java short.

13.11.5.1.40 Method getShort(String)

```
Synopsis: public short getShort(java.lang.String columnName)
              throws java.sql.SQLException;
```

Parameters

`columnName` - is the SQL name of the column

`return` - the column value; if the value is SQL NULL, the result is 0

Exceptions

SQLException if a database-access error occurs.

Get the value of a column in the current row as a Java short.

13.11.5.1.41 Method getShowDeleted()

```
Synopsis: public boolean getShowDeleted() throws java.sql.SQLException;
```

Exceptions

SQLException if an error occurs.

This property determines whether or not rows marked for deletion appear in the set of current rows. The default value is false.

13.11.5.1.42 Method getStatement()

```
Synopsis: public Statement getStatement() throws java.sql.SQLException;
```

Exceptions

SQLException if a database-access error occurs

Return the Statement that produced the ResultSet.

13.11.5.1.43 Method getString(int)

```
Synopsis: public String getString(int columnIndex)
              throws java.sql.SQLException;
```

Parameters

`columnIndex` - the first column is 1, the second is 2, ...

`return` - the column value; if the value is SQL NULL, the result is null

Exceptions

SQLException if a database-access error occurs.

Get the value of a column in the current row as a Java String.

13.11.5.1.44 Method getString(String)

```
Synopsis: public String getString(java.lang.String columnName)
              throws java.sql.SQLException;
```

Parameters

`columnName` - is the SQL name of the column

`return` - the column value; if the value is SQL NULL, the result is null

Exceptions

SQLException if a database-access error occurs.

Get the value of a column in the current row as a Java String.

13.11.5.1.45 Method getTableName()

```
Synopsis: public String getTableName() throws java.sql.SQLException;
```

Exceptions

SQLException if an error occurs.

Returns an identifier for the object (table) that was used to create this rowset.

13.11.5.1.46 Method getTime(int)

```
Synopsis: public Time getTime(int columnIndex) throws java.sql.SQLException;
```

Parameters

columnIndex - the first column is 1, the second is 2, ...

`return` - the column value; if the value is SQL NULL, the result is null

Exceptions

SQLException if a database-access error occurs.

Get the value of a column in the current row as a java.sql.Time object.

13.11.5.1.47 Method getTime(int, Calendar)

```
Synopsis: public Time getTime(int columnIndex, java.util.Calendar cal)
              throws java.sql.SQLException;
```

Parameters

columnIndex - the first column is 1, the second is 2, ...

cal - the calendar to use in constructing the time

`return` - the column value; if the value is SQL NULL, the result is null

Exceptions

SQLException if a database-access error occurs.

Get the value of a column in the current row as a java.sql.Time object. Use the calendar to construct an appropriate millisecond value for the Time, if the underlying database doesn't store timezone information.

13.11.5.1.48 Method getTime(String)

```
Synopsis: public Time getTime(java.lang.String columnName)
              throws java.sql.SQLException;
```

Parameters

columnName - is the SQL name of the column

`return` - the column value; if the value is SQL NULL, the result is null

Exceptions

SQLException if a database-access error occurs.

Get the value of a column in the current row as a java.sql.Time object.

13.11.5.1.49 Method getTime(String, Calendar)

```
Synopsis: public Time getTime(java.lang.String columnName,
                              java.util.Calendar cal)
              throws java.sql.SQLException;
```

Parameters

columnName - is the SQL name of the column

cal - the calendar to use in constructing the time

Parameters

`return` - the column value; if the value is SQL NULL, the result is null

Exceptions

SQLException if a database-access error occurs.

Get the value of a column in the current row as a java.sql.Time object. Use the calendar to construct an appropriate millisecond value for the Time, if the underlying database doesn't store timezone information.

13.11.5.1.50 Method getTimestamp(int)

```
Synopsis: public Timestamp getTimestamp(int columnIndex)
                throws java.sql.SQLException;
```

Parameters

`columnIndex` - the first column is 1, the second is 2, ...

`return` - the column value; if the value is SQL NULL, the result is null

Exceptions

SQLException if a database-access error occurs.

Get the value of a column in the current row as a java.sql.Timestamp object.

13.11.5.1.51 Method getTimestamp(int, Calendar)

```
Synopsis: public Timestamp getTimestamp(int columnIndex,
                                        java.util.Calendar cal)
                throws java.sql.SQLException;
```

Parameters

`columnIndex` - the first column is 1, the second is 2, ...

`cal` - the calendar to use in constructing the timestamp

`return` - the column value; if the value is SQL NULL, the result is null

Exceptions

SQLException if a database-access error occurs.

Get the value of a column in the current row as a java.sql.Timestamp object. Use the calendar to construct an appropriate millisecond value for the Timestamp, if the underlying database doesn't store timezone information.

13.11.5.1.52 Method getTimestamp(String)

```
Synopsis: public Timestamp getTimestamp(java.lang.String columnName)
                throws java.sql.SQLException;
```

Parameters

`columnName` - is the SQL name of the column

`return` - the column value; if the value is SQL NULL, the result is null

Exceptions

SQLException if a database-access error occurs.

Get the value of a column in the current row as a java.sql.Timestamp object.

13.11.5.1.53 Method getTimestamp(String, Calendar)

```
Synopsis: public Timestamp getTimestamp(java.lang.String columnName,
                                        java.util.Calendar cal)
                throws java.sql.SQLException;
```

Parameters

`columnName` - is the SQL name of the column

`cal` - the calendar to use in constructing the timestamp

Parameters

`return` - the column value; if the value is SQL NULL, the result is null

Exceptions

SQLException if a database-access error occurs.

Get the value of a column in the current row as a java.sql.Timestamp object. Use the calendar to construct an appropriate millisecond value for the Timestamp, if the underlying database doesn't store timezone information.

13.11.5.1.54 Method getUnicodeStream(int)

```
Synopsis: public InputStream getUnicodeStream(int columnIndex)
              throws java.sql.SQLException;
```

Parameters

`columnIndex` - the first column is 1, the second is 2, ...

`return` - a Java input stream that delivers the database column value as a stream of two byte Unicode characters. If the value is SQL NULL then the result is null.

Exceptions

SQLException if a database-access error occurs.

13.11.6 Deprecated

A column value can be retrieved as a stream of Unicode characters and then read in chunks from the stream. This method is particularly suitable for retrieving large LONGVARCHAR values. The JDBC driver will do any necessary conversion from the database format into Unicode.

Note: All the data in the returned stream must be read prior to getting the value of any other column. The next call to a get method implicitly closes the stream. . Also, a stream may return 0 for available() whether there is data available or not.

13.11.6.1 Method getUnicodeStream(String)

```
Synopsis: public InputStream getUnicodeStream(java.lang.String columnName)
              throws java.sql.SQLException;
```

Parameters

`columnName` - is the SQL name of the column

`return` - a Java input stream that delivers the database column value as a stream of two byte Unicode characters. If the value is SQL NULL then the result is null.

Exceptions

SQLException if a database-access error occurs.

13.11.7 Deprecated

A column value can be retrieved as a stream of Unicode characters and then read in chunks from the stream. This method is particularly suitable for retrieving large LONGVARCHAR values. The JDBC driver will do any necessary conversion from the database format into Unicode.

Note: All the data in the returned stream must be read prior to getting the value of any other column. The next call to a get method implicitly closes the stream.

13.11.7.1 Method getURL(int)

```
Synopsis: public URL getURL(int columnIndex) throws java.sql.SQLException;
```

Parameters

`columnIndex` - the index of the column 1 is the first, 2 is the second,...

`return` - the column value as a `java.net.URL` object; if the value is SQL `NULL`, the value returned is `null` in the

Parameters

Java programming language

Exceptions

SQLException if a database access error occurs, or if a URL is malformed

Additional Information

Since 1.4

Retrieves the value of the designated column in the current row of this `ResultSet` object as a `java.net.URL` object in the Java programming language.

13.11.7.1.1 Method getURL(String)

```
Synopsis: public URL getURL(java.lang.String columnName)
                 throws java.sql.SQLException;
```

Parameters

`columnName` - the SQL name of the column

`return` - the column value as a `java.net.URL` object; if the value is SQL `NULL`, the value returned is `null` in the Java programming language

Exceptions

SQLException if a database access error occurs or if a URL is malformed

Additional Information

Since 1.4

Retrieves the value of the designated column in the current row of this `ResultSet` object as a `java.net.URL` object in the Java programming language.

13.11.7.1.2 Method getWarnings()

```
Synopsis: public SQLWarning getWarnings() throws java.sql.SQLException;
```

Exceptions

SQLException if a database-access error occurs.

The first warning reported by calls on this ResultSet is returned. Subsequent ResultSet warnings will be chained to this SQLWarning.

The warning chain is automatically cleared each time a new row is read.

Note: This warning chain only covers warnings caused by ResultSet methods. Any warning caused by statement methods (such as reading OUT parameters) will be chained on the Statement object.

13.11.7.1.3 Method insertRow()

```
Synopsis: public void insertRow() throws java.sql.SQLException;
```

Exceptions

SQLException if a database access error occurs, if this method is called when the cursor is not on the insert row

Inserts the contents of the insert row into this rowset following the current row and it notifies listeners that the row has changed. The cursor must be on the insert row when this method is called. The method marks the current row as inserted, but it does not insert the row to the underlying data source. The method acceptChanges must be called to insert the row to the data source.

13.11.7.1.4 Method isAfterLast()

```
Synopsis: public boolean isAfterLast() throws java.sql.SQLException;
```

Exceptions

SQLException if a database-access error occurs.

Determine if the cursor is after the last row in the rowset.

13.11.7.1.5 Method isBeforeFirst()

```
Synopsis: public boolean isBeforeFirst() throws java.sql.SQLException;
```

Exceptions

SQLException if a database-access error occurs.

Determine if the cursor is before the first row in the rowset.

13.11.7.1.6 Method isFirst()

```
Synopsis: public boolean isFirst() throws java.sql.SQLException;
```

Exceptions

SQLException if a database-access error occurs.

Determine if the cursor is on the first row of the rowset.

13.11.7.1.7 Method isLast()

```
Synopsis: public boolean isLast() throws java.sql.SQLException;
```

Exceptions

SQLException if a database-access error occurs.

Determine if the cursor is on the last row of the rowset. Note: Calling isLast() may be expensive since the rowset might need to check ahead one row in order to determine whether the current row is the last row in the rowset.

13.11.7.1.8 Method last()

```
Synopsis: public boolean last() throws java.sql.SQLException;
```

Exceptions

SQLException if a database-access error occurs, or rowset type is TYPE_FORWARD_ONLY.

Moves to the last row in the rowset. It notifies listeners that the cursor has moved.

13.11.7.1.9 Method moveToCurrentRow()

```
Synopsis: public void moveToCurrentRow() throws java.sql.SQLException;
```

Exceptions

SQLException if a database-access error occurs, or the rowset is not updatable

Move the cursor to the remembered cursor position, usually the current row. Has no effect unless the cursor is on the insert row.

13.11.7.1.10 Method moveToInsertRow()

```
Synopsis: public void moveToInsertRow() throws java.sql.SQLException;
```

Exceptions

SQLException if a database-access error occurs, or the rowset is not updatable

Move to the insert row. The current cursor position is remembered while the cursor is positioned on the insert row. The insert row is a special row associated with an updatable rowset. It is essentially a buffer where a new row may be constructed by calling the updateXXX() methods prior to inserting the row into the rowset. Only the updateXXX(), getXXX(), and insertRow() methods may be called when the cursor is on the insert row. All of the columns in a rowset must be given a value each time this method is called before calling insertRow(). UpdateXXX()must be called before getXXX() on a column.

13.11.7.1.11 Method next()

```
Synopsis: public boolean next() throws java.sql.SQLException;
```

Exceptions

SQLException	if a database-access error occurs.

A ResultSet is initially positioned before its first row; the first call to next makes the first row the current row; the second call makes the second row the current row, etc.

If an input stream from the previous row is open, it is implicitly closed. The ResultSet's warning chain is cleared when a new row is read.

13.11.7.1.12 Method populate(ResultSet)

```
Synopsis: public void populate(java.sql.ResultSet rs)
              throws java.sql.SQLException;
```

Parameters

rs - the data to be read

Exceptions

SQLException	if an error occurs.

Populate the OPLCachedRowSet object with data from a ResultSet. This method is an alternative to execute() for filling the rowset with data. Populate() doesn't require that the properties needed by execute(), such as the command property, be set. A RowSetChangedEvent is sent to all registered listeners prior to returning.

13.11.7.1.13 Method previous()

```
Synopsis: public boolean previous() throws java.sql.SQLException;
```

Exceptions

SQLException	if a database-access error occurs, or rowset type is TYPE_FORWAR_DONLY.

Moves to the previous row in the rowset.

Note: previous() is not the same as relative(-1) since it makes sense to call previous() when there is no current row.

13.11.7.1.14 Method refreshRow()

```
Synopsis: public void refreshRow() throws java.sql.SQLException;
```

Exceptions

SQLException	if a database-access error occurs, or if called when on the insert row.

Sets the current row with its original value and marks the row as not updated, thus undoing any changes made to the row since the last call to the methods updateRow or deleteRow. This method should be called only when the cursor is on a row in this rowset. Cannot be called when on the insert row.

13.11.7.1.15 Method relative(int)

```
Synopsis: public boolean relative(int rows) throws java.sql.SQLException;
```

Exceptions

SQLException	if a database-access error occurs, or there is no current row, or rowset type is TYPE_FORWARD_ONLY.

Moves a relative number of rows, either positive or negative. Attempting to move beyond the first/last row in the rowset positions the cursor before/after the the first/last row. Calling relative(0) is valid, but does not change the cursor position. It notifies listeners that the cursor has moved.

Note: Calling relative(1) is different than calling next() since is makes sense to call next() when there is no current row, for example, when the cursor is positioned before the first row or after the last row of the rowset.

13.11.7.1.16 Method release()

Synopsis: `public void` **`release`**`() throws java.sql.SQLException;`

Exceptions

SQLException	if an error occurs.

Releases the current contents of the rowset. Outstanding updates are discarded. The rowset contains no rows after release is called. A RowSetChangedEvent is sent to all registered listeners prior to returning.

13.11.7.1.17 Method restoreOriginal()

Synopsis: `public void` **`restoreOriginal`**`() throws java.sql.SQLException;`

Exceptions

SQLException	if an error occurs.

Restores the rowset to its original state (the original value of the rowset becomes the current value). All updates, inserts, and deletes made to the original state are lost. The cursor is positioned before the first row. A RowSetChangedEvent is sent to all registered listeners prior to returning.

13.11.7.1.18 Method rowDeleted()

Synopsis: `public boolean` **`rowDeleted`**`() throws java.sql.SQLException;`

Exceptions

SQLException	if a database-access error occurs

Additional Information

See Also	java.sql.DatabaseMetaData.deletesAreDetected

Determine if this row has been deleted. A deleted row may leave a visible "hole" in a rowset. This method can be used to detect holes in a rowset. The value returned depends on whether or not the rowset can detect deletions.

13.11.7.1.19 Method rowInserted()

Synopsis: `public boolean` **`rowInserted`**`() throws java.sql.SQLException;`

Exceptions

SQLException	if a database-access error occurs

Additional Information

See Also	java.sql.DatabaseMetaData.insertsAreDetected

Determine if the current row has been inserted. The value returned depends on whether or not the rowset can detect visible inserts.

13.11.7.1.20 Method rowUpdated()

Synopsis: `public boolean` **`rowUpdated`**`() throws java.sql.SQLException;`

Exceptions

SQLException	if a database-access error occurs

Additional Information

Additional Information

See Also java.sql.DatabaseMetaData.updatesAreDetected

Determine if the current row has been updated. The value returned depends on whether or not the rowset can detect updates.

13.11.7.1.21 Method setCommand(String)

```
Synopsis: public void setCommand(java.lang.String cmd)
                 throws java.sql.SQLException;
```

Parameters

`cmd` - - a String object containing an SQL query that will be set as the command

Exceptions

SQLException - if an error occurs

Sets this OPLCachedRowSet object's command property to the given String object and clears the parameters, if any, that were set for the previous command.

13.11.7.1.22 Method setConcurrency(int)

```
Synopsis: public void setConcurrency(int concurrency)
                 throws java.sql.SQLException;
```

Parameters

`concurrency` - - one of the following constants: ResultSet.CONCUR_READ_ONLY or ResultSet.CONCUR_UPDATABLE

Exceptions

SQLException - if an error occurs

Sets the concurrency for this rowset to the specified concurrency. The default concurrency is ResultSet.CONCUR_UPDATABLE.

13.11.7.1.23 Method setKeyColumns(int[])

```
Synopsis: public void setKeyColumns(int[] keys) throws java.sql.SQLException;
```

Parameters

`keys` - - an array of int indicating the columns that form a key for this OPLCachedRowSet object; every element in the array must be greater than 0 and less than or equal to the number of columns in this rowset

Exceptions

SQLException if an error occurs.

Sets this OPLCachedRowSet object's keyCols field with the given array of column numbers, which forms a key for uniquely identifying a row in this rowset. Note: If you don't set the keyCols, the OPLCachedRowSet will set automatically based on RowSetMetaData

13.11.7.1.24 Method setMetaData(RowSetMetaData)

```
Synopsis: public void setMetaData(javax.sql.RowSetMetaData md)
                 throws java.sql.SQLException;
```

Parameters

`md` - a metadata object

Exceptions

SQLException if a database-access error occurs.

Set the rowset's metadata.

13.11.7.1.25 Method setOriginal()

```
Synopsis: public void setOriginal() throws java.sql.SQLException;
```

Exceptions

SQLException if an error occurs.

Marks all rows in this rowset as being original rows. Any updates made to the rows become the original values for the rowset. Calls to the method setOriginal cannot be reversed.

13.11.7.1.26 Method setOriginalRow()

```
Synopsis: public void setOriginalRow() throws java.sql.SQLException;
```

Exceptions

SQLException if an error occurs.

Marks the current row in this rowset as being an original row. The row is no longer marked as inserted, deleted, or updated, and its values become the original values. A call to setOriginalRow cannot be reversed.

13.11.7.1.27 Method setShowDeleted(boolean)

```
Synopsis: public void setShowDeleted(boolean value)
              throws java.sql.SQLException;
```

Parameters

`value` - true if deleted rows should be shown, false otherwise

Exceptions

SQLException if an error occurs.

Set the show deleted property.

13.11.7.1.28 Method setTableName(String)

```
Synopsis: public void setTableName(java.lang.String _tableName)
              throws java.sql.SQLException;
```

Parameters

`_tabName` - - a String object that identifies the table from which this OPLCachedRowSet object was derived

Exceptions

SQLException if an error occurs.

Sets the identifier for the table from which this rowset was derived to the given table name. Note: You don't usually need to set a table name, because the OPLCachedRowSet tries to determine the table name from your SQL query command.

13.11.7.1.29 Method size()

```
Synopsis: public int size();
```

Returns the number of rows in this OPLCachedRowSet object.

13.11.7.1.30 Method toCollection()

```
Synopsis: public Collection toCollection() throws java.sql.SQLException;
```

Exceptions

SQLException if an error occurs.

Convert the rowset to a collection of tables. Each tables represents a row of the original rowset. The tables are keyed by column number. A copy of the rowset's contents is made.

13.11.7.1.31 Method toCollection(int)

```
Synopsis: public Collection toCollection(int col) throws java.sql.SQLException;
```

Exceptions

SQLException if an error occurs.

Return a column of the rowset as a collection. Makes a copy of the column's data.

13.11.7.1.32 Method updateArray(int, Array)

```
Synopsis: public void updateArray(int columnIndex, java.sql.Array x)
                 throws java.sql.SQLException;
```

Parameters

`columnIndex` - the first column is 1, the second is 2, ...

`x` - the new column value

Exceptions

SQLException if a database access error occurs

Additional Information

Since 1.4

Updates the designated column with a `java.sql.Array` value. The updater methods are used to update column values in the current row or the insert row. The updater methods do not update the underlying database; instead the `updateRow` or `insertRow` methods are called to update the database.

13.11.7.1.33 Method updateArray(String, Array)

```
Synopsis: public void updateArray(java.lang.String columnName,
                                  java.sql.Array x)
                 throws java.sql.SQLException;
```

Parameters

`columnName` - the name of the column

`x` - the new column value

Exceptions

SQLException if a database access error occurs

Additional Information

Since 1.4

Updates the designated column with a `java.sql.Array` value. The updater methods are used to update column values in the current row or the insert row. The updater methods do not update the underlying database; instead the `updateRow` or `insertRow` methods are called to update the database.

13.11.7.1.34 Method updateAsciiStream(int, InputStream, int)

```
Synopsis: public void updateAsciiStream(int columnIndex, java.io.InputStream x,
                                        int length)
                 throws java.sql.SQLException;
```

Parameters

`columnIndex` - the first column is 1, the second is 2, ...

`x` - the new column value

`length` - the length of the stream

Exceptions

SQLException if a database-access error occurs

Update a column with an ascii stream value. The updateXXX() methods are used to update column values in the current row, or the insert row. The updateXXX() methods do not update the underlying database, instead the updateRow() or insertRow() methods are called to update the database.

13.11.7.1.35 Method updateAsciiStream(String, InputStream, int)

```
Synopsis: public void updateAsciiStream(java.lang.String columnName,
                                        java.io.InputStream x,
                                        int length)
                 throws java.sql.SQLException;
```

Parameters

columnName - the name of the column

x - the new column value

length - of the stream

Exceptions

SQLException if a database-access error occurs

Update a column with an ascii stream value. The updateXXX() methods are used to update column values in the current row, or the insert row. The updateXXX() methods do not update the underlying database, instead the updateRow() or insertRow() methods are called to update the database.

13.11.7.1.36 Method updateBigDecimal(int, BigDecimal)

```
Synopsis: public void updateBigDecimal(int columnIndex, java.math.BigDecimal x)
                 throws java.sql.SQLException;
```

Parameters

columnIndex - the first column is 1, the second is 2, ...

x - the new column value

Exceptions

SQLException if a database-access error occurs

Update a column with a BigDecimal value. The updateXXX() methods are used to update column values in the current row, or the insert row. The updateXXX() methods do not update the underlying database, instead the updateRow() or insertRow() methods are called to update the database.

13.11.7.1.37 Method updateBigDecimal(String, BigDecimal)

```
Synopsis: public void updateBigDecimal(java.lang.String columnName,
                                       java.math.BigDecimal x)
                 throws java.sql.SQLException;
```

Parameters

columnName - the name of the column

x - the new column value

Exceptions

SQLException if a database-access error occurs

Update a column with a BigDecimal value. The updateXXX() methods are used to update column values in the current row, or the insert row. The updateXXX() methods do not update the underlying database, instead the updateRow() or insertRow() methods are called to update the database.

13.11.7.1.38 Method updateBinaryStream(int, InputStream, int)

```
Synopsis: public void updateBinaryStream(int columnIndex,
                                         java.io.InputStream x, int length)
                 throws java.sql.SQLException;
```

Parameters

columnIndex - the first column is 1, the second is 2, ...

x - the new column value

length - the length of the stream

Exceptions

SQLException if a database-access error occurs

Update a column with a binary stream value. The updateXXX() methods are used to update column values in the current row, or the insert row. The updateXXX() methods do not update the underlying database, instead the updateRow() or insertRow() methods are called to update the database.

13.11.7.1.39 Method updateBinaryStream(String, InputStream, int)

```
Synopsis: public void updateBinaryStream(java.lang.String columnName,
                                         java.io.InputStream x,
                                         int length)
                throws java.sql.SQLException;
```

Parameters

columnName - the name of the column

x - the new column value

length - of the stream

Exceptions

SQLException if a database-access error occurs

Update a column with a binary stream value. The updateXXX() methods are used to update column values in the current row, or the insert row. The updateXXX() methods do not update the underlying database, instead the updateRow() or insertRow() methods are called to update the database.

13.11.7.1.40 Method updateBlob(int, Blob)

```
Synopsis: public void updateBlob(int columnIndex, java.sql.Blob x)
                throws java.sql.SQLException;
```

Parameters

columnIndex - the first column is 1, the second is 2, ...

x - the new column value

Exceptions

SQLException if a database access error occurs

Additional Information

Since 1.4

Updates the designated column with a `java.sql.Blob` value. The updater methods are used to update column values in the current row or the insert row. The updater methods do not update the underlying database; instead the `updateRow` or `insertRow` methods are called to update the database.

13.11.7.1.41 Method updateBlob(String, Blob)

```
Synopsis: public void updateBlob(java.lang.String columnName, java.sql.Blob x)
                throws java.sql.SQLException;
```

Parameters

columnName - the name of the column

x - the new column value

Exceptions

SQLException if a database access error occurs

Additional Information

Since 1.4

Updates the designated column with a `java.sql.Blob` value. The updater methods are used to update column values in the current row or the insert row. The updater methods do not update the underlying database; instead the `updateRow` or `insertRow` methods are called to update the database.

13.11.7.1.42 Method updateBoolean(int, boolean)

```
Synopsis: public void updateBoolean(int columnIndex, boolean x)
            throws java.sql.SQLException;
```

Parameters

`columnIndex` - the first column is 1, the second is 2, ...

`x` - the new column value

Exceptions

SQLException if a database-access error occurs

Update a column with a boolean value. The updateXXX() methods are used to update column values in the current row, or the insert row. The updateXXX() methods do not update the underlying database, instead the updateRow() or insertRow() methods are called to update the database.

13.11.7.1.43 Method updateBoolean(String, boolean)

```
Synopsis: public void updateBoolean(java.lang.String columnName, boolean x)
            throws java.sql.SQLException;
```

Parameters

`columnName` - the name of the column

`x` - the new column value

Exceptions

SQLException if a database-access error occurs

Update a column with a boolean value. The updateXXX() methods are used to update column values in the current row, or the insert row. The updateXXX() methods do not update the underlying database, instead the updateRow() or insertRow() methods are called to update the database.

13.11.7.1.44 Method updateByte(int, byte)

```
Synopsis: public void updateByte(int columnIndex, byte x)
            throws java.sql.SQLException;
```

Parameters

`columnIndex` - the first column is 1, the second is 2, ...

`x` - the new column value

Exceptions

SQLException if a database-access error occurs

Update a column with a byte value. The updateXXX() methods are used to update column values in the current row, or the insert row. The updateXXX() methods do not update the underlying database, instead the updateRow() or insertRow() methods are called to update the database.

13.11.7.1.45 Method updateByte(String, byte)

```
Synopsis: public void updateByte(java.lang.String columnName, byte x)
            throws java.sql.SQLException;
```

Parameters

Parameters

`columnName` - the name of the column

`x` - the new column value

Exceptions

SQLException if a database-access error occurs

Update a column with a byte value. The updateXXX() methods are used to update column values in the current row, or the insert row. The updateXXX() methods do not update the underlying database, instead the updateRow() or insertRow() methods are called to update the database.

13.11.7.1.46 Method updateBytes(int, byte[])

```
Synopsis: public void updateBytes(int columnIndex, byte[] x)
              throws java.sql.SQLException;
```

Parameters

`columnIndex` - the first column is 1, the second is 2, ...

`x` - the new column value

Exceptions

SQLException if a database-access error occurs

Update a column with a byte array value. The updateXXX() methods are used to update column values in the current row, or the insert row. The updateXXX() methods do not update the underlying database, instead the updateRow() or insertRow() methods are called to update the database.

13.11.7.1.47 Method updateBytes(String, byte[])

```
Synopsis: public void updateBytes(java.lang.String columnName, byte[] x)
              throws java.sql.SQLException;
```

Parameters

`columnName` - the name of the column

`x` - the new column value

Exceptions

SQLException if a database-access error occurs

Update a column with a byte array value. The updateXXX() methods are used to update column values in the current row, or the insert row. The updateXXX() methods do not update the underlying database, instead the updateRow() or insertRow() methods are called to update the database.

13.11.7.1.48 Method updateCharacterStream(int, Reader, int)

```
Synopsis: public void updateCharacterStream(int columnIndex, java.io.Reader x,
                                           int length)
              throws java.sql.SQLException;
```

Parameters

`columnIndex` - the first column is 1, the second is 2, ...

`x` - the new column value

`length` - the length of the stream

Exceptions

SQLException if a database-access error occurs

Update a column with a character stream value. The updateXXX() methods are used to update column values in the current row, or the insert row. The updateXXX() methods do not update the underlying database, instead the updateRow() or insertRow() methods are called to update the database.

13.11.7.1.49 Method updateCharacterStream(String, Reader, int)

```
Synopsis: public void updateCharacterStream(java.lang.String columnName,
                                            java.io.Reader reader,
                                            int length)
              throws java.sql.SQLException;
```

Parameters

`columnName` - the name of the column	
`x` - the new column value	
`length` - of the stream	

Exceptions

SQLException	if a database-access error occurs

Update a column with a character stream value. The updateXXX() methods are used to update column values in the current row, or the insert row. The updateXXX() methods do not update the underlying database, instead the updateRow() or insertRow() methods are called to update the database.

13.11.7.1.50 Method updateClob(int, Clob)

```
Synopsis: public void updateClob(int columnIndex, java.sql.Clob x)
              throws java.sql.SQLException;
```

Parameters

`columnIndex` - the first column is 1, the second is 2, ...

`x` - the new column value

Exceptions

SQLException	if a database access error occurs

Additional Information

Since	1.4

Updates the designated column with a `java.sql.Clob` value. The updater methods are used to update column values in the current row or the insert row. The updater methods do not update the underlying database; instead the `updateRow` or `insertRow` methods are called to update the database.

13.11.7.1.51 Method updateClob(String, Clob)

```
Synopsis: public void updateClob(java.lang.String columnName, java.sql.Clob x)
              throws java.sql.SQLException;
```

Parameters

`columnName` - the name of the column

`x` - the new column value

Exceptions

SQLException	if a database access error occurs

Additional Information

Since	1.4

Updates the designated column with a `java.sql.Clob` value. The updater methods are used to update column values in the current row or the insert row. The updater methods do not update the underlying database; instead the `updateRow` or `insertRow` methods are called to update the database.

13.11.7.1.52 Method updateDate(int, Date)

```
Synopsis: public void updateDate(int columnIndex, java.sql.Date x)
              throws java.sql.SQLException;
```

Parameters

`columnIndex` - the first column is 1, the second is 2, ...

`x` - the new column value

Exceptions

SQLException if a database-access error occurs

Update a column with a Date value. The updateXXX() methods are used to update column values in the current row, or the insert row. The updateXXX() methods do not update the underlying database, instead the updateRow() or insertRow() methods are called to update the database.

13.11.7.1.53 Method updateDate(String, Date)

```
Synopsis: public void updateDate(java.lang.String columnName, java.sql.Date x)
              throws java.sql.SQLException;
```

Parameters

`columnName` - the name of the column

`x` - the new column value

Exceptions

SQLException if a database-access error occurs

Update a column with a Date value. The updateXXX() methods are used to update column values in the current row, or the insert row. The updateXXX() methods do not update the underlying database, instead the updateRow() or insertRow() methods are called to update the database.

13.11.7.1.54 Method updateDouble(int, double)

```
Synopsis: public void updateDouble(int columnIndex, double x)
              throws java.sql.SQLException;
```

Parameters

`columnIndex` - the first column is 1, the second is 2, ...

`x` - the new column value

Exceptions

SQLException if a database-access error occurs

Update a column with a Double value. The updateXXX() methods are used to update column values in the current row, or the insert row. The updateXXX() methods do not update the underlying database, instead the updateRow() or insertRow() methods are called to update the database.

13.11.7.1.55 Method updateDouble(String, double)

```
Synopsis: public void updateDouble(java.lang.String columnName, double x)
              throws java.sql.SQLException;
```

Parameters

`columnName` - the name of the column

`x` - the new column value

Exceptions

SQLException if a database-access error occurs

Update a column with a double value. The updateXXX() methods are used to update column values in the current row, or the insert row. The updateXXX() methods do not update the underlying database, instead the updateRow() or insertRow() methods are called to update the database.

13.11.7.1.56 Method updateFloat(int, float)

```
Synopsis: public void updateFloat(int columnIndex, float x)
              throws java.sql.SQLException;
```

Parameters

`columnIndex` - the first column is 1, the second is 2, ...

`x` - the new column value

Exceptions

SQLException if a database-access error occurs

Update a column with a float value. The updateXXX() methods are used to update column values in the current row, or the insert row. The updateXXX() methods do not update the underlying database, instead the updateRow() or insertRow() methods are called to update the database.

13.11.7.1.57 Method updateFloat(String, float)

```
Synopsis: public void updateFloat(java.lang.String columnName, float x)
              throws java.sql.SQLException;
```

Parameters

`columnName` - the name of the column

`x` - the new column value

Exceptions

SQLException if a database-access error occurs

Update a column with a float value. The updateXXX() methods are used to update column values in the current row, or the insert row. The updateXXX() methods do not update the underlying database, instead the updateRow() or insertRow() methods are called to update the database.

13.11.7.1.58 Method updateInt(int, int)

```
Synopsis: public void updateInt(int columnIndex, int x)
              throws java.sql.SQLException;
```

Parameters

`columnIndex` - the first column is 1, the second is 2, ...

`x` - the new column value

Exceptions

SQLException if a database-access error occurs

Update a column with an integer value. The updateXXX() methods are used to update column values in the current row, or the insert row. The updateXXX() methods do not update the underlying database, instead the updateRow() or insertRow() methods are called to update the database.

13.11.7.1.59 Method updateInt(String, int)

```
Synopsis: public void updateInt(java.lang.String columnName, int x)
              throws java.sql.SQLException;
```

Parameters

`columnName` - the name of the column

`x` - the new column value

Exceptions

SQLException if a database-access error occurs

Update a column with an integer value. The updateXXX() methods are used to update column values in the current row, or the insert row. The updateXXX() methods do not update the underlying database, instead the updateRow() or insertRow() methods are called to update the database.

13.11.7.1.60 Method updateLong(int, long)

```
Synopsis: public void updateLong(int columnIndex, long x)
              throws java.sql.SQLException;
```

Parameters

columnIndex - the first column is 1, the second is 2, ...

x - the new column value

Exceptions

SQLException if a database-access error occurs

Update a column with a long value. The updateXXX() methods are used to update column values in the current row, or the insert row. The updateXXX() methods do not update the underlying database, instead the updateRow() or insertRow() methods are called to update the database.

13.11.7.1.61 Method updateLong(String, long)

```
Synopsis: public void updateLong(java.lang.String columnName, long x)
              throws java.sql.SQLException;
```

Parameters

columnName - the name of the column

x - the new column value

Exceptions

SQLException if a database-access error occurs

Update a column with a long value. The updateXXX() methods are used to update column values in the current row, or the insert row. The updateXXX() methods do not update the underlying database, instead the updateRow() or insertRow() methods are called to update the database.

13.11.7.1.62 Method updateNull(int)

```
Synopsis: public void updateNull(int columnIndex) throws java.sql.SQLException;
```

Parameters

columnIndex - the first column is 1, the second is 2, ...

Exceptions

SQLException if a database-access error occurs

Give a nullable column a null value. The updateXXX() methods are used to update column values in the current row, or the insert row. The updateXXX() methods do not update the underlying database, instead the updateRow() or insertRow() methods are called to update the database.

13.11.7.1.63 Method updateNull(String)

```
Synopsis: public void updateNull(java.lang.String columnName)
              throws java.sql.SQLException;
```

Parameters

columnName - the name of the column

Exceptions

SQLException if a database-access error occurs

Update a column with a null value. The updateXXX() methods are used to update column values in the current row, or the insert row. The updateXXX() methods do not update the underlying database, instead the updateRow() or insertRow() methods are called to update the database.

13.11.7.1.64 Method updateObject(int, Object)

```
Synopsis: public void updateObject(int columnIndex, java.lang.Object x)
              throws java.sql.SQLException;
```

Parameters

columnIndex - the first column is 1, the second is 2, ...

Parameters

`x` - the new column value

Exceptions

SQLException if a database-access error occurs

Update a column with an Object value. The updateXXX() methods are used to update column values in the current row, or the insert row. The updateXXX() methods do not update the underlying database, instead the updateRow() or insertRow() methods are called to update the database.

13.11.7.1.65 Method updateObject(int, Object, int)

```
Synopsis: public void updateObject(int columnIndex, java.lang.Object x,
                                   int scale) throws java.sql.SQLException;
```

Parameters

`columnIndex` - the first column is 1, the second is 2, ...

`x` - the new column value

`scale` - For java.sql.Types.DECIMAL or java.sql.Types.NUMERIC types this is the number of digits after the decimal. For all other types this value will be ignored.

Exceptions

SQLException if a database-access error occurs

Update a column with an Object value. The updateXXX() methods are used to update column values in the current row, or the insert row. The updateXXX() methods do not update the underlying database, instead the updateRow() or insertRow() methods are called to update the database.

13.11.7.1.66 Method updateObject(String, Object)

```
Synopsis: public void updateObject(java.lang.String columnName,
                                   java.lang.Object x)
              throws java.sql.SQLException;
```

Parameters

`columnName` - the name of the column

`x` - the new column value

Exceptions

SQLException if a database-access error occurs

Update a column with an Object value. The updateXXX() methods are used to update column values in the current row, or the insert row. The updateXXX() methods do not update the underlying database, instead the updateRow() or insertRow() methods are called to update the database.

13.11.7.1.67 Method updateObject(String, Object, int)

```
Synopsis: public void updateObject(java.lang.String columnName,
                                   java.lang.Object x, int scale)
              throws java.sql.SQLException;
```

Parameters

`columnName` - the name of the column

`x` - the new column value

`scale` - For java.sql.Types.DECIMAL or java.sql.Types.NUMERIC types this is the number of digits after the decimal. For all other types this value will be ignored.

Exceptions

SQLException if a database-access error occurs

Update a column with an Object value. The updateXXX() methods are used to update column values in the current row, or the insert row. The updateXXX() methods do not update the underlying database, instead the updateRow() or insertRow() methods are called to update the database.

13.11.7.1.68 Method updateRef(int, Ref)

```
Synopsis: public void updateRef(int columnIndex, java.sql.Ref x)
              throws java.sql.SQLException;
```

Parameters

`columnIndex` - the first column is 1, the second is 2, ...

`x` - the new column value

Exceptions

SQLException if a database access error occurs

Additional Information

Since 1.4

Updates the designated column with a `java.sql.Ref` value. The updater methods are used to update column values in the current row or the insert row. The updater methods do not update the underlying database; instead the `updateRow` or `insertRow` methods are called to update the database.

13.11.7.1.69 Method updateRef(String, Ref)

```
Synopsis: public void updateRef(java.lang.String columnName, java.sql.Ref x)
              throws java.sql.SQLException;
```

Parameters

`columnName` - the name of the column

`x` - the new column value

Exceptions

SQLException if a database access error occurs

Additional Information

Since 1.4

Updates the designated column with a `java.sql.Ref` value. The updater methods are used to update column values in the current row or the insert row. The updater methods do not update the underlying database; instead the `updateRow` or `insertRow` methods are called to update the database.

13.11.7.1.70 Method updateRow()

```
Synopsis: public void updateRow() throws java.sql.SQLException;
```

Exceptions

SQLException if a database-access error occurs, or if called when on the insert row

Marks the current row of this rowset as updated but it does not update the row to the underlying data source. The method acceptChanges must be called to update the row to the data source. It notifies listeners that the row has changed also. Cannot be called when on the insert row.

13.11.7.1.71 Method updateShort(int, short)

```
Synopsis: public void updateShort(int columnIndex, short x)
              throws java.sql.SQLException;
```

Parameters

`columnIndex` - the first column is 1, the second is 2, ...

`x` - the new column value

Exceptions

SQLException if a database-access error occurs

Update a column with a short value. The updateXXX() methods are used to update column values in the current row, or the insert row. The updateXXX() methods do not update the underlying database, instead the updateRow() or insertRow() methods are called to update the database.

13.11.7.1.72 Method updateShort(String, short)

```
Synopsis: public void updateShort(java.lang.String columnName, short x)
              throws java.sql.SQLException;
```

Parameters

columnName - the name of the column

x - the new column value

Exceptions

SQLException if a database-access error occurs

Update a column with a short value. The updateXXX() methods are used to update column values in the current row, or the insert row. The updateXXX() methods do not update the underlying database, instead the updateRow() or insertRow() methods are called to update the database.

13.11.7.1.73 Method updateString(int, String)

```
Synopsis: public void updateString(int columnIndex, java.lang.String x)
              throws java.sql.SQLException;
```

Parameters

columnIndex - the first column is 1, the second is 2, ...

x - the new column value

Exceptions

SQLException if a database-access error occurs

Update a column with a String value. The updateXXX() methods are used to update column values in the current row, or the insert row. The updateXXX() methods do not update the underlying database, instead the updateRow() or insertRow() methods are called to update the database.

13.11.7.1.74 Method updateString(String, String)

```
Synopsis: public void updateString(java.lang.String columnName,
                                   java.lang.String x)
              throws java.sql.SQLException;
```

Parameters

columnName - the name of the column

x - the new column value

Exceptions

SQLException if a database-access error occurs

Update a column with a String value. The updateXXX() methods are used to update column values in the current row, or the insert row. The updateXXX() methods do not update the underlying database, instead the updateRow() or insertRow() methods are called to update the database.

13.11.7.1.75 Method updateTime(int, Time)

```
Synopsis: public void updateTime(int columnIndex, java.sql.Time x)
              throws java.sql.SQLException;
```

Parameters

columnIndex - the first column is 1, the second is 2, ...

x - the new column value

Exceptions

SQLException if a database-access error occurs

Update a column with a Time value. The updateXXX() methods are used to update column values in the current row, or the insert row. The updateXXX() methods do not update the underlying database, instead the updateRow() or insertRow() methods are called to update the database.

13.11.7.1.76 Method updateTime(String, Time)

```
Synopsis: public void updateTime(java.lang.String columnName, java.sql.Time x)
              throws java.sql.SQLException;
```

Parameters

`columnName` - the name of the column

`x` - the new column value

Exceptions

SQLException if a database-access error occurs

Update a column with a Time value. The updateXXX() methods are used to update column values in the current row, or the insert row. The updateXXX() methods do not update the underlying database, instead the updateRow() or insertRow() methods are called to update the database.

13.11.7.1.77 Method updateTimestamp(int, Timestamp)

```
Synopsis: public void updateTimestamp(int columnIndex, java.sql.Timestamp x)
              throws java.sql.SQLException;
```

Parameters

`columnIndex` - the first column is 1, the second is 2, ...

`x` - the new column value

Exceptions

SQLException if a database-access error occurs

Update a column with a Timestamp value. The updateXXX() methods are used to update column values in the current row, or the insert row. The updateXXX() methods do not update the underlying database, instead the updateRow() or insertRow() methods are called to update the database.

13.11.7.1.78 Method updateTimestamp(String, Timestamp)

```
Synopsis: public void updateTimestamp(java.lang.String columnName,
                                      java.sql.Timestamp x)
              throws java.sql.SQLException;
```

Parameters

`columnName` - the name of the column

`x` - the new column value

Exceptions

SQLException if a database-access error occurs

Update a column with a Timestamp value. The updateXXX() methods are used to update column values in the current row, or the insert row. The updateXXX() methods do not update the underlying database, instead the updateRow() or insertRow() methods are called to update the database.

13.11.7.1.79 Method wasNull()

```
Synopsis: public boolean wasNull() throws java.sql.SQLException;
```

Exceptions

SQLException if a database-access error occurs.

A column may have the value of SQL NULL; wasNull reports whether the last column read had this special value. Note that you must first call getXXX on a column to try to read its value and then call wasNull() to find if the value was the SQL NULL.

13.11.8 Class OPLConnectionPoolDataSource

A ConnectionPoolDataSource object is a factory for PooledConnection objects. An object that implements this interface will typically be registered with a JNDI service.

13.11.8.1 Synopsis

```
public class OPLConnectionPoolDataSource extends OPLDataSource implements ConnectionPoolDataSource,Serial
  // Public Constructors
  public OPLConnectionPoolDataSource();
  // Public Methods
  public void finalize() throws java.lang.Throwable;
  public OPLPoolStatistic get_statistics();
  public OPLPoolStatistic[] getAll_statistics();
  public void close() throws java.sql.SQLException;
  public void connectionClosed(javax.sql.ConnectionEvent event);
  public void connectionErrorOccurred(javax.sql.ConnectionEvent event);
  public Reference getReference() throws javax.naming.NamingException;
  public void fill() throws java.sql.SQLException;
  public PooledConnection getPooledConnection() throws java.sql.SQLException;
  public PooledConnection getPooledConnection(java.lang.String _user,
                                             java.lang.String _password)
      throws java.sql.SQLException;
  public int getMinPoolSize();
  public void setMinPoolSize(int parm) throws java.sql.SQLException;
  public int getMaxPoolSize();
  public void setMaxPoolSize(int parm) throws java.sql.SQLException;
  public int getInitialPoolSize();
  public void setInitialPoolSize(int parm) throws java.sql.SQLException;
  public int getMaxIdleTime();
  public void setMaxIdleTime(int parm) throws java.sql.SQLException;
  public int getPropertyCycle();
  public void setPropertyCycle(int parm);
  public int getMaxStatements();
  public void setMaxStatements(int parm) throws java.sql.SQLException;
}
```

Inheritance Path

java.lang.Object

|

openlink.javax.OPLDataSource

|

openlink.javax.OPLConnectionPoolDataSource

13.11.8.2 Members

13.11.8.2.1 Method close()

Synopsis: `public void close() throws java.sql.SQLException;`

Exceptions

java.sql.SQLException	if a database-access error occurs

Physically close all the pooled connections in the cache and free all the resources

13.11.8.2.2 Method connectionClosed(ConnectionEvent)

Synopsis: `public void connectionClosed(javax.sql.ConnectionEvent event);`

Parameters

Parameters

`event` - an event object describing the source of the event

Invoked when the application calls close() on its representation of the connection.

13.11.8.2.3 Method connectionErrorOccurred(ConnectionEvent)

Synopsis: `public void` **`connectionErrorOccurred`**`(javax.sql.ConnectionEvent` *`event`*`);`

Parameters

`event` - an event object describing the source of the event

Invoked when a fatal connection error occurs, just before an SQLException is thrown to the application.

13.11.8.2.4 Method fill()

Synopsis: `public void` **`fill`**`() throws java.sql.SQLException;`

Exceptions

java.sql.SQLException if a error occurs

Fills the cache with PooledConnections for later use. Ignored if the MinPoolSize is 0. It is usually called when the OPLConnectionPoolDataSource is created via JNDI calls.

13.11.8.2.5 Method get_statistics()

Synopsis: `public OPLPoolStatistic` **`get_statistics`**`();`

Return the cache statistics for the OPLConnectionPoolDataSource

13.11.8.2.6 Method getAll_statistics()

Synopsis: `public OPLPoolStatistic[]` **`getAll_statistics`**`();`

Return an array of the cache statistics for the all created OPLConnectionPoolDataSources

13.11.8.2.7 Method getInitialPoolSize()

Synopsis: `public int` **`getInitialPoolSize`**`();`

Get the number of physical connections the pool will contain when it is created

13.11.8.2.8 Method getMaxIdleTime()

Synopsis: `public int` **`getMaxIdleTime`**`();`

Get the number of seconds that a physical connection will remain unused in the pool before the connection is closed. Zero (0) indicates no limit.

13.11.8.2.9 Method getMaxPoolSize()

Synopsis: `public int` **`getMaxPoolSize`**`();`

Get the maximum number of physical connections the pool will be able contain. Zero (0) indicates no maximum size.

13.11.8.2.10 Method getMaxStatements()

Synopsis: `public int` **`getMaxStatements`**`();`

Get the total number of statements that the pool will keep open. Zero (0) indicates that caching of statements is disabled.

13.11.8.2.11 Method getMinPoolSize()

```
Synopsis: public int getMinPoolSize();
```

Get the minimum number of physical connections the pool will keep available at all times. Zero (0) indicates that connections will be created as needed.

13.11.8.2.12 Method getPooledConnection()

```
Synopsis: public PooledConnection getPooledConnection() throws
            java.sql.SQLException;
```

Exceptions

java.sql.SQLException if a database-access error occurs

Attempt to establish a database connection.

13.11.8.2.13 Method getPooledConnection(String, String)

```
Synopsis: public PooledConnection getPooledConnection(java.lang.String _user,
                                                      java.lang.String _password)
              throws java.sql.SQLException;
```

Parameters

`user` - the database user on whose behalf the Connection is being made

`password` - the user's password

`return` - a PooledConnection to the database

Exceptions

java.sql.SQLException if a database-access error occurs

Attempt to establish a database connection.

13.11.8.2.14 Method getPropertyCycle()

```
Synopsis: public int getPropertyCycle();
```

Get the interval, in seconds, that the pool will wait before enforcing the current policy defined by the values of the above connection pool properties

13.11.8.2.15 Method setInitialPoolSize(int)

```
Synopsis: public void setInitialPoolSize(int parm)
              throws java.sql.SQLException;
```

Parameters

`parm` - a number of physical connections

Exceptions

java.sql.SQLException if an error occurs

Set the number of physical connections the pool should contain when it is created

13.11.8.2.16 Method setMaxIdleTime(int)

```
Synopsis: public void setMaxIdleTime(int parm) throws java.sql.SQLException;
```

Parameters

Parameters

`parm` - a number of seconds

Exceptions

java.sql.SQLException if an error occurs

Set the number of seconds that a physical connection should remain unused in the pool before the connection is closed. Zero (0) indicates no limit.

13.11.8.2.17 Method setMaxPoolSize(int)

```
Synopsis: public void setMaxPoolSize(int parm) throws java.sql.SQLException;
```

Parameters

`parm` - a maximum number of physical connections

Exceptions

java.sql.SQLException if an error occurs

Set the maximum number of physical conections that the pool should contain. Zero (0) indicates no maximum size. The default value is 0 .

13.11.8.2.18 Method setMaxStatements(int)

```
Synopsis: public void setMaxStatements(int parm) throws java.sql.SQLException;
```

Parameters

`parm` - a total number of statements

Exceptions

java.sql.SQLException if an error occurs

Set the total number of statements that the pool should keep open. Zero (0) indicates that caching of statements is disabled.

13.11.8.2.19 Method setMinPoolSize(int)

```
Synopsis: public void setMinPoolSize(int parm) throws java.sql.SQLException;
```

Parameters

`parm` - a minimum number of physical connections

Exceptions

java.sql.SQLException if an error occurs

Set the number of physical connections the pool should keep available at all times. Zero (0) indicates that connections should be created as needed The default value is 0 .

13.11.8.2.20 Method setPropertyCycle(int)

```
Synopsis: public void setPropertyCycle(int parm);
```

Parameters

`parm` - an interval (in seconds)

Set the interval, in seconds, that the pool should wait before enforcing the current policy defined by the values of the above connection pool properties

13.11.9 Class OPLDataSource

A DataSource object is a factory for Connection objects. An object that implements the DataSource interface will typically be registered with a JNDI service provider. A JDBC driver that is accessed via the DataSource API does not automatically register itself with the DriverManager.

13.11.9.1 Synopsis

```
public class OPLDataSource implements DataSource,Serializable,Referenceable {
  // Public Constructors
  public OPLDataSource();
  // Public Methods
  public Reference getReference() throws javax.naming.NamingException;
  public Connection getConnection() throws java.sql.SQLException;
  public Connection getConnection(java.lang.String user,
                                  java.lang.String password)
      throws java.sql.SQLException;
  public PrintWriter getLogWriter() throws java.sql.SQLException;
  public void setLogWriter(java.io.PrintWriter parm)
      throws java.sql.SQLException;
  public int getLoginTimeout() throws java.sql.SQLException;
  public void setLoginTimeout(int parm) throws java.sql.SQLException;
  public String getDataSourceName();
  public void setDataSourceName(java.lang.String parm);
  public String getDescription();
  public void setDescription(java.lang.String parm);
  public int getPortNumber();
  public void setPortNumber(int parm);
  public String getServerName();
  public void setServerName(java.lang.String parm);
  public String getDatabaseName();
  public void setDatabaseName(java.lang.String parm);
  public String getUser();
  public void setUser(java.lang.String parm);
  public String getPassword();
  public void setPassword(java.lang.String parm);
  public String getSVT();
  public void setSVT(java.lang.String parm);
  public boolean getReadOnly();
  public void setReadOnly(boolean parm);
  public String getDbOptions();
  public void setDbOptions(java.lang.String parm);
  public String getFBS();
  public void setFBS(java.lang.String parm);
  public String getCharSet();
  public void setCharSet(java.lang.String parm);
  public String getCursorModel();
  public void setCursorModel(java.lang.String parm);
  public String getConcurrencyType();
  public void setConcurrencyType(java.lang.String parm);
  public boolean getEncrypted();
  public void setEncrypted(boolean parm);
  public boolean getDeferLongFetch();
  public void setDeferLongFetch(boolean parm);
  public String getApplication();
  public void setApplication(java.lang.String parm);
  public boolean getUnicode();
  public void setUnicode(boolean parm);
  public String getURL();
  public void setURL(java.lang.String parm);
}
```

Inheritance Path

java.lang.Object

|

openlink.javax.OPLDataSource

13.11.9.2 Members

13.11.9.2.1 Method getApplication()

```
Synopsis: public String getApplication();
```

Get the Application name set on this DataSource instance.

13.11.9.2.2 Method getCharSet()

```
Synopsis: public String getCharSet();
```

Get the charset of remote database set on this DataSource instance.

13.11.9.2.3 Method getConcurrencyType()

```
Synopsis: public String getConcurrencyType();
```

Get the default Concurrency Type for scrollable ResultSets set on this DataSource instance.

13.11.9.2.4 Method getConnection()

```
Synopsis: public Connection getConnection() throws java.sql.SQLException;
```

Exceptions

java.sql.SQLException if a database-access error occurs

Attempt to establish a database connection.

13.11.9.2.5 Method getConnection(String, String)

```
Synopsis: public Connection getConnection(java.lang.String user,
                                          java.lang.String password)
                   throws java.sql.SQLException;
```

Parameters

`user` - the database user on whose behalf the Connection is being made

`password` - the user's password

`return` - a Connection to the database

Exceptions

java.sql.SQLException if a database-access error occurs

Attempt to establish a database connection.

13.11.9.2.6 Method getCursorModel()

```
Synopsis: public String getCursorModel();
```

Get the default Cursor Model for scrollable ResultSets set on this DataSource instance.

13.11.9.2.7 Method getDatabaseName()

```
Synopsis: public String getDatabaseName();
```

Get the name of the database set on this DataSource instance.

13.11.9.2.8 Method getDataSourceName()

```
Synopsis: public String getDataSourceName();
```

Get the datasource name for this instance if set. The default value is "OPLDataSourceName"

13.11.9.2.9 Method getDbOptions()

```
Synopsis: public String getDbOptions();
```

Get the Database Options set on this DataSource instance.

13.11.9.2.10 Method getDeferLongFetch()

```
Synopsis: public boolean getDeferLongFetch();
```

Get if the DataSource instance uses the deferLongFetch for long data.

13.11.9.2.11 Method getDescription()

```
Synopsis: public String getDescription();
```

Get the description of this data source.

13.11.9.2.12 Method getEncrypted()

```
Synopsis: public boolean getEncrypted();
```

Get if the outgoing packets are encrypted for this DataSource instance.

13.11.9.2.13 Method getFBS()

```
Synopsis: public String getFBS();
```

Get the Fetch Buffer Size set on this DataSource instance.

13.11.9.2.14 Method getLoginTimeout()

```
Synopsis: public int getLoginTimeout() throws java.sql.SQLException;
```

Exceptions

java.sql.SQLException	if a database-access error occurs

Gets the maximum time in seconds that this data source can wait while attempting to connect to a database. A value of zero means that the timeout is the default system timeout if there is one; otherwise it means that there is no timeout. When a DataSource object is created the login timeout is initially zero.

13.11.9.2.15 Method getLogWriter()

```
Synopsis: public PrintWriter getLogWriter() throws java.sql.SQLException;
```

Exceptions

java.sql.SQLException	if a database-access error occurs

The log writer is a character output stream to which all logging and tracing messages for this data source object instance will be printed. This includes messages printed by the methods of this object, messages printed by methods of other objects manufactured by this object, and so on. Messages printed to a data source specific log writer are not printed to the log writer associated with the java.sql.Drivermanager class. When a DataSource object is created the log writer is initially null, in other words, logging is disabled.

13.11.9.2.16 Method getPassword()

```
Synopsis: public String getPassword();
```

Get the password set on this DataSource instance.

13.11.9.2.17 Method getPortNumber()

Synopsis: `public int` **`getPortNumber`**`();`

Get the port number on which oplrqb is listening for requests. The default value is 5000

13.11.9.2.18 Method getReadOnly()

Synopsis: `public boolean` **`getReadOnly`**`();`

Get the ReadOnly attribute set on this DataSource instance.

13.11.9.2.19 Method getServerName()

Synopsis: `public String` **`getServerName`**`();`

Get the name of the host on which oplrqb is running. The default value is "localhost"

13.11.9.2.20 Method getSVT()

Synopsis: `public String` **`getSVT`**`();`

Get the ServerType set on this DataSource instance.

13.11.9.2.21 Method getUnicode()

Synopsis: `public boolean` **`getUnicode`**`();`

Get if the DataSource instance uses the Unicode connection with a DBMS agent.

13.11.9.2.22 Method getURL()

Synopsis: `public String` **`getURL`**`();`

Get the URL for this Datasource instance.

13.11.9.2.23 Method getUser()

Synopsis: `public String` **`getUser`**`();`

Get the user name set on this DataSource instance.

13.11.9.2.24 Method setApplication(String)

Synopsis: `public void` **`setApplication`**`(java.lang.String` *`parm`*`);`

Parameters

`parm` - Application name to be set

Set the Application name with which connections have to be obtained. Will be overwritten with value from URL, if URL is set.

13.11.9.2.25 Method setCharSet(String)

Synopsis: `public void` **`setCharSet`**`(java.lang.String` *`parm`*`);`

Parameters

Parameters

`parm` - CharSet to be set

Set the charset of remote database with which connections have to be obtained. The default value is get from `System.getProperty("fil...` Will be overwritten with value from URL, if URL is set.

13.11.9.2.26 Method setConcurrencyType(String)

```
Synopsis: public void setConcurrencyType(java.lang.String parm);
```

Parameters

`parm` - Concurrency Type to be set

Set the default Concurrency Type for scrollable ResultSets with which connections have to be obtained. Will be overwritten with value from URL, if URL is set.

13.11.9.2.27 Method setCursorModel(String)

```
Synopsis: public void setCursorModel(java.lang.String parm);
```

Parameters

`parm` - CursorModel to be set

Set the default Cursor Model for scrollable ResultSets with which connections have to be obtained. Will be overwritten with value from URL, if URL is set.

13.11.9.2.28 Method setDatabaseName(String)

```
Synopsis: public void setDatabaseName(java.lang.String parm);
```

Parameters

`parm` - database name to be set

Set the name of a particular database on a server. Will be overwritten with value from URL, if URL is set.

13.11.9.2.29 Method setDataSourceName(String)

```
Synopsis: public void setDataSourceName(java.lang.String parm);
```

Parameters

`parm` - DataSource name to be set

Set the DataSource name. The default value is "OPLDataSourceName"

13.11.9.2.30 Method setDbOptions(String)

```
Synopsis: public void setDbOptions(java.lang.String parm);
```

Parameters

`parm` - Database Options to be set

Set the Database Options with which connections have to be obtained. Will be overwritten with value from URL, if URL is set.

13.11.9.2.31 Method setDeferLongFetch(boolean)

```
Synopsis: public void setDeferLongFetch(boolean parm);
```

Parameters

`parm` - true for a DeferLongFetch connection

Set the DeferLongFetch attribute with which connections have to be obtained. The default value is false . Will be overwritten with value from URL, if URL is set.

13.11.9.2.32 Method setDescription(String)

```
Synopsis: public void setDescription(java.lang.String parm);
```

Parameters

parm - Description to be set.

Set the description for this data source instance.

13.11.9.2.33 Method setEncrypted(boolean)

```
Synopsis: public void setEncrypted(boolean parm);
```

Parameters

parm - true if outgoing packets must be encrypted

Set the Encrypted flag for outgoing packets with which connections have to be obtained. Will be overwritten with value from URL, if URL is set.

13.11.9.2.34 Method setFBS(String)

```
Synopsis: public void setFBS(java.lang.String parm);
```

Parameters

parm - FetchBufferSize to be set

Sets number of JDBC resultset rows that get packed into a single network packet Will be overwritten with value from URL, if URL is set.

13.11.9.2.35 Method setLoginTimeout(int)

```
Synopsis: public void setLoginTimeout(int parm) throws java.sql.SQLException;
```

Parameters

parm - the data source login time limit (in seconds)

Exceptions

java.sql.SQLException if a database-access error occurs

Sets the maximum time in seconds that this data source will wait while attempting to connect to a database. A value of zero specifies that the timeout is the default system timeout if there is one; otherwise it specifies that there is no timeout. When a DataSource object is created the login timeout is initially zero.

13.11.9.2.36 Method setLogWriter(PrintWriter)

```
Synopsis: public void setLogWriter(java.io.PrintWriter parm)
              throws java.sql.SQLException;
```

Parameters

parm - the new log writer; to disable, set to null

Exceptions

java.sql.SQLException if a database-access error occurs

The log writer is a character output stream to which all logging and tracing messages for this data source object instance will be printed. This includes messages printed by the methods of this object, messages printed by methods of other objects manufactured by this object, and so on. Messages printed to a data source specific log writer are not printed to the log writer associated with the java.sql.Drivermanager class. When a DataSource object is created the log writer is initially null, in other words, logging is disabled.

13.11.9.2.37 Method setPassword(String)

```
Synopsis: public void setPassword(java.lang.String parm);
```

Parameters

parm - password to be set

Set the password with which connections have to be obtained. Will be overwritten with value from URL, if URL is set.

13.11.9.2.38 Method setPortNumber(int)

```
Synopsis: public void setPortNumber(int parm);
```

Parameters

parm - port number on which oplrqb is listening

Set the port number where the oplrqb is listening for requests. The default value is 5000 . Will be overwritten with value from URL, if URL is set.

13.11.9.2.39 Method setReadOnly(boolean)

```
Synopsis: public void setReadOnly(boolean parm);
```

Parameters

parm - true for a readOnly connection

Set the ReadOnly attribute with which connections have to be obtained. The default value is false . Will be overwritten with value from URL, if URL is set.

13.11.9.2.40 Method setServerName(String)

```
Synopsis: public void setServerName(java.lang.String parm);
```

Parameters

parm - name of the host on which oplrqb is running

Set the name of the host where the oplrqb is running. The default value is "localhost" . Will be overwritten with value from URL, if URL is set.

13.11.9.2.41 Method setSVT(String)

```
Synopsis: public void setSVT(java.lang.String parm);
```

Parameters

parm - ServerType to be set

Set the ServerType with which connections have to be obtained. Will be overwritten with value from URL, if URL is set.

13.11.9.2.42 Method setUnicode(boolean)

```
Synopsis: public void setUnicode(boolean parm);
```

Parameters

parm - true for a Unicode connection

Set the Unicode attribute with which connections have to be obtained. The default value is false . Will be overwritten with value from URL, if URL is set.

13.11.9.2.43 Method setURL(String)

```
Synopsis: public void setURL(java.lang.String parm);
```

Parameters

parm - URL to be set

Set the URL with which connections have to be obtained.

13.11.9.2.44 Method setUser(String)

```
Synopsis: public void setUser(java.lang.String parm);
```

Parameters

parm - username to be set

Set the user name with which connections have to be obtained. Will be overwritten with value from URL, if URL is set.

13.11.10 Class OPLDataSourceFactory

13.11.10.1 Synopsis

```
public class OPLDataSourceFactory implements ObjectFactory {
  // Public Constructors
  public OPLDataSourceFactory();
  // Public Methods
  public Object getObjectInstance(java.lang.Object obj, javax.naming.Name name,
                                  javax.naming.Context nameCtx,
                                  java.util.Hashtable environment)
      throws java.lang.Exception;
}
```

Inheritance Path

java.lang.Object

|

openlink.javax.OPLDataSourceFactory

13.11.11 Class OPLJdbcRowSet

A OPLJdbcRowSet is a connected rowset. The purpose of the OPLJdbcRowSet class is to act like a JavaBean at design time and provide a thin layer that wraps around a JDBC ResultSet at runtime.

When execute() is called a OPLJdbcRowSet object opens a JDBC connection internally which remains open until close() is called, unless a Connection is passed to execute() explicitly, in which case that Connection is used instead. ResultSet operations such as cursor movement, updating, etc. are simply delegated to an underlying JDBC ResultSet object which is maintained internally.

13.11.11.1 Synopsis

```
public class OPLJdbcRowSet extends BaseRowSet  {
  // Public Constructors
  public OPLJdbcRowSet();
  // Public Methods
  public void finalize() throws java.lang.Throwable;
  public void execute() throws java.sql.SQLException;
  public void execute(java.sql.Connection _conn) throws java.sql.SQLException;
  public void close() throws java.sql.SQLException;
  public void cancelRowUpdates() throws java.sql.SQLException;
  public boolean next() throws java.sql.SQLException;
  public boolean previous() throws java.sql.SQLException;
  public boolean first() throws java.sql.SQLException;
  public boolean last() throws java.sql.SQLException;
  public boolean absolute(int row) throws java.sql.SQLException;
  public boolean relative(int rows) throws java.sql.SQLException;
public void beforeFirst() throws java.sql.SQLException;
public void afterLast() throws java.sql.SQLException;
public boolean isBeforeFirst() throws java.sql.SQLException;
public boolean isAfterLast() throws java.sql.SQLException;
public boolean isFirst() throws java.sql.SQLException;
public boolean isLast() throws java.sql.SQLException;
public int getRow() throws java.sql.SQLException;
public boolean rowUpdated() throws java.sql.SQLException;
public boolean rowInserted() throws java.sql.SQLException;
public boolean rowDeleted() throws java.sql.SQLException;
public void refreshRow() throws java.sql.SQLException;
public void insertRow() throws java.sql.SQLException;
public void updateRow() throws java.sql.SQLException;
public void deleteRow() throws java.sql.SQLException;
public void moveToInsertRow() throws java.sql.SQLException;
public void moveToCurrentRow() throws java.sql.SQLException;
public boolean wasNull() throws java.sql.SQLException;
public SQLWarning getWarnings() throws java.sql.SQLException;
public void clearWarnings() throws java.sql.SQLException;
public String getCursorName() throws java.sql.SQLException;
public ResultSetMetaData getMetaData() throws java.sql.SQLException;
public int findColumn(java.lang.String columnName)
    throws java.sql.SQLException;
public String getString(int columnIndex) throws java.sql.SQLException;
public boolean getBoolean(int columnIndex) throws java.sql.SQLException;
public byte getByte(int columnIndex) throws java.sql.SQLException;
public short getShort(int columnIndex) throws java.sql.SQLException;
public int getInt(int columnIndex) throws java.sql.SQLException;
public long getLong(int columnIndex) throws java.sql.SQLException;
public float getFloat(int columnIndex) throws java.sql.SQLException;
public double getDouble(int columnIndex) throws java.sql.SQLException;
public BigDecimal getBigDecimal(int columnIndex)
    throws java.sql.SQLException;
public BigDecimal getBigDecimal(int columnIndex, int scale)
    throws java.sql.SQLException;
public byte[] getBytes(int columnIndex) throws java.sql.SQLException;
public Date getDate(int columnIndex) throws java.sql.SQLException;
public Time getTime(int columnIndex) throws java.sql.SQLException;
public Timestamp getTimestamp(int columnIndex) throws java.sql.SQLException;
public InputStream getAsciiStream(int columnIndex)
    throws java.sql.SQLException;
public InputStream getUnicodeStream(int columnIndex)
    throws java.sql.SQLException;
public InputStream getBinaryStream(int columnIndex)
    throws java.sql.SQLException;
public Object getObject(int columnIndex) throws java.sql.SQLException;
public String getString(java.lang.String columnName)
    throws java.sql.SQLException;
public boolean getBoolean(java.lang.String columnName)
    throws java.sql.SQLException;
public byte getByte(java.lang.String columnName)
    throws java.sql.SQLException;
public short getShort(java.lang.String columnName)
    throws java.sql.SQLException;
public int getInt(java.lang.String columnName) throws java.sql.SQLException;
public long getLong(java.lang.String columnName)
    throws java.sql.SQLException;
public float getFloat(java.lang.String columnName)
    throws java.sql.SQLException;
public double getDouble(java.lang.String columnName)
    throws java.sql.SQLException;
public BigDecimal getBigDecimal(java.lang.String columnName, int scale)
    throws java.sql.SQLException;
public byte[] getBytes(java.lang.String columnName)
    throws java.sql.SQLException;
public Date getDate(java.lang.String columnName)
    throws java.sql.SQLException;
public Time getTime(java.lang.String columnName)
    throws java.sql.SQLException;
public Timestamp getTimestamp(java.lang.String columnName)
    throws java.sql.SQLException;
public InputStream getAsciiStream(java.lang.String columnName)
    throws java.sql.SQLException;
```

```
public InputStream getUnicodeStream(java.lang.String columnName)
    throws java.sql.SQLException;
public InputStream getBinaryStream(java.lang.String columnName)
    throws java.sql.SQLException;
public Object getObject(java.lang.String columnName)
    throws java.sql.SQLException;
public Reader getCharacterStream(int columnIndex)
    throws java.sql.SQLException;
public Reader getCharacterStream(java.lang.String columnName)
    throws java.sql.SQLException;
public BigDecimal getBigDecimal(java.lang.String columnName)
    throws java.sql.SQLException;
public void updateNull(int columnIndex) throws java.sql.SQLException;
public void updateBoolean(int columnIndex, boolean x)
    throws java.sql.SQLException;
public void updateByte(int columnIndex, byte x) throws java.sql.SQLException;
public void updateShort(int columnIndex, short x)
    throws java.sql.SQLException;
public void updateInt(int columnIndex, int x) throws java.sql.SQLException;
public void updateLong(int columnIndex, long x) throws java.sql.SQLException;
public void updateFloat(int columnIndex, float x)
    throws java.sql.SQLException;
public void updateDouble(int columnIndex, double x)
    throws java.sql.SQLException;
public void updateBigDecimal(int columnIndex, java.math.BigDecimal x)
    throws java.sql.SQLException;
public void updateString(int columnIndex, java.lang.String x)
    throws java.sql.SQLException;
public void updateBytes(int columnIndex, byte[] x)
    throws java.sql.SQLException;
public void updateDate(int columnIndex, java.sql.Date x)
    throws java.sql.SQLException;
public void updateTime(int columnIndex, java.sql.Time x)
    throws java.sql.SQLException;
public void updateTimestamp(int columnIndex, java.sql.Timestamp x)
    throws java.sql.SQLException;
public void updateAsciiStream(int columnIndex, java.io.InputStream x,
                              int length) throws java.sql.SQLException;
public void updateBinaryStream(int columnIndex, java.io.InputStream x,
                               int length) throws java.sql.SQLException;
public void updateCharacterStream(int columnIndex, java.io.Reader x,
                                  int length) throws java.sql.SQLException;
public void updateObject(int columnIndex, java.lang.Object x, int scale)
    throws java.sql.SQLException;
public void updateObject(int columnIndex, java.lang.Object x)
    throws java.sql.SQLException;
public void updateNull(java.lang.String columnName)
    throws java.sql.SQLException;
public void updateBoolean(java.lang.String columnName, boolean x)
    throws java.sql.SQLException;
public void updateByte(java.lang.String columnName, byte x)
    throws java.sql.SQLException;
public void updateShort(java.lang.String columnName, short x)
    throws java.sql.SQLException;
public void updateInt(java.lang.String columnName, int x)
    throws java.sql.SQLException;
public void updateLong(java.lang.String columnName, long x)
    throws java.sql.SQLException;
public void updateFloat(java.lang.String columnName, float x)
    throws java.sql.SQLException;
public void updateDouble(java.lang.String columnName, double x)
    throws java.sql.SQLException;
public void updateBigDecimal(java.lang.String columnName,
                             java.math.BigDecimal x)
    throws java.sql.SQLException;
public void updateString(java.lang.String columnName, java.lang.String x)
    throws java.sql.SQLException;
public void updateBytes(java.lang.String columnName, byte[] x)
    throws java.sql.SQLException;
public void updateDate(java.lang.String columnName, java.sql.Date x)
    throws java.sql.SQLException;
public void updateTime(java.lang.String columnName, java.sql.Time x)
    throws java.sql.SQLException;
    public void updateTimestamp(java.lang.String columnName,
                                java.sql.Timestamp x)
        throws java.sql.SQLException;
    public void updateAsciiStream(java.lang.String columnName,
                                  java.io.InputStream x, int length)
        throws java.sql.SQLException;
    public void updateBinaryStream(java.lang.String columnName,
                                   java.io.InputStream x, int length)
        throws java.sql.SQLException;
    public void updateCharacterStream(java.lang.String columnName,
                                      java.io.Reader x, int length)
        throws java.sql.SQLException;
    public void updateObject(java.lang.String columnName, java.lang.Object x,
                             int scale) throws java.sql.SQLException;
    public void updateObject(java.lang.String columnName, java.lang.Object x)
        throws java.sql.SQLException;
    public Statement getStatement() throws java.sql.SQLException;
    public Object getObject(int colIndex, java.util.Map map)
        throws java.sql.SQLException;
    public Ref getRef(int colIndex) throws java.sql.SQLException;
    public Blob getBlob(int colIndex) throws java.sql.SQLException;
    public Clob getClob(int colIndex) throws java.sql.SQLException;
    public Array getArray(int colIndex) throws java.sql.SQLException;
    public Object getObject(java.lang.String colName, java.util.Map map)
        throws java.sql.SQLException;
    public Ref getRef(java.lang.String colName) throws java.sql.SQLException;
    public Blob getBlob(java.lang.String colName) throws java.sql.SQLException;
    public Clob getClob(java.lang.String colName) throws java.sql.SQLException;
    public Array getArray(java.lang.String colName) throws java.sql.SQLException;
    public Date getDate(int columnIndex, java.util.Calendar cal)
        throws java.sql.SQLException;
    public Date getDate(java.lang.String columnName, java.util.Calendar cal)
        throws java.sql.SQLException;
    public Time getTime(int columnIndex, java.util.Calendar cal)
        throws java.sql.SQLException;
    public Time getTime(java.lang.String columnName, java.util.Calendar cal)
        throws java.sql.SQLException;
    public Timestamp getTimestamp(int columnIndex, java.util.Calendar cal)
        throws java.sql.SQLException;
    public Timestamp getTimestamp(java.lang.String columnName,
                                  java.util.Calendar cal)
        throws java.sql.SQLException;
    public URL getURL(int columnIndex) throws java.sql.SQLException;
    public URL getURL(java.lang.String columnName) throws java.sql.SQLException;
    public void updateRef(int columnIndex, java.sql.Ref x)
        throws java.sql.SQLException;
    public void updateRef(java.lang.String columnName, java.sql.Ref x)
        throws java.sql.SQLException;
    public void updateBlob(int columnIndex, java.sql.Blob x)
        throws java.sql.SQLException;
    public void updateBlob(java.lang.String columnName, java.sql.Blob x)
        throws java.sql.SQLException;
    public void updateClob(int columnIndex, java.sql.Clob x)
        throws java.sql.SQLException;
    public void updateClob(java.lang.String columnName, java.sql.Clob x)
        throws java.sql.SQLException;
    public void updateArray(int columnIndex, java.sql.Array x)
        throws java.sql.SQLException;
    public void updateArray(java.lang.String columnName, java.sql.Array x)
        throws java.sql.SQLException;
  }
```

Inheritance Path

java.lang.Object

|

openlink.javax.BaseRowSet

|

openlink.javax.OPLJdbcRowSet

13.11.11.2 Members

13.11.11.2.1 Method absolute(int)

```
Synopsis: public boolean absolute(int row) throws java.sql.SQLException;
```

Exceptions

SQLException	if a database-access error occurs, or row is 0, or result set type is TYPE_FORWARD_ONLY.

Move to an absolute row number in the result set.

If row is positive, moves to an absolute row with respect to the beginning of the result set. The first row is row 1, the second is row 2, etc.

If row is negative, moves to an absolute row position with respect to the end of result set. For example, calling absolute(-1) positions the cursor on the last row, absolute(-2) indicates the next-to-last row, etc.

An attempt to position the cursor beyond the first/last row in the result set, leaves the cursor before/after the first/last row, respectively.

Note: Calling absolute(1) is the same as calling first(). Calling absolute(-1) is the same as calling last().

13.11.11.2.2 Method afterLast()

```
Synopsis: public void afterLast() throws java.sql.SQLException;
```

Exceptions

SQLException	if a database-access error occurs, or result set type is TYPE_FORWARD_ONLY.

Moves to the end of the result set, just after the last row. Has no effect if the result set contains no rows.

13.11.11.2.3 Method beforeFirst()

```
Synopsis: public void beforeFirst() throws java.sql.SQLException;
```

Exceptions

SQLException	if a database-access error occurs, or result set type is TYPE_FORWARD_ONLY

Moves to the front of the result set, just before the first row. Has no effect if the result set contains no rows.

13.11.11.2.4 Method cancelRowUpdates()

```
Synopsis: public void cancelRowUpdates() throws java.sql.SQLException;
```

Exceptions

SQLException	if an error occurs.

The cancelRowUpdates() method may be called after calling an updateXXX() method(s) and before calling updateRow() to rollback the updates made to a row. If no updates have been made or updateRow() has already been called, then this method has no effect.

13.11.11.2.5 Method clearWarnings()

```
Synopsis: public void clearWarnings() throws java.sql.SQLException;
```

Exceptions

SQLException	if a database-access error occurs.

After this call getWarnings returns null until a new warning is reported for this ResultSet.

13.11.11.2.6 Method close()

```
Synopsis: public void close() throws java.sql.SQLException;
```

Exceptions

SQLException if a database-access error occurs.

In some cases, it is desirable to immediately release a ResultSet's database and JDBC resources instead of waiting for this to happen when it is automatically closed; the close method provides this immediate release.

Note: A ResultSet is automatically closed by the Statement that generated it when that Statement is closed, re-executed, or is used to retrieve the next result from a sequence of multiple results. A ResultSet is also automatically closed when it is garbage collected.

13.11.11.2.7 Method deleteRow()

```
Synopsis: public void deleteRow() throws java.sql.SQLException;
```

Exceptions

SQLException if a database-access error occurs, or if called when on the insert row.

Delete the current row from the result set and the underlying database. Cannot be called when on the insert row.

13.11.11.2.8 Method execute()

```
Synopsis: public void execute() throws java.sql.SQLException;
```

Exceptions

SQLException if a database-access error occurs.

Populates the rowset with data. Execute() may use the following properties: url, data source name, user name, password, transaction isolation, and type map to create a connection for reading data. Execute may use the following properties to create a statement to execute a command: command, read only, maximum field size, maximum rows, escape processing, and query timeout. If the required properties have not been set, an exception is thrown. If successful, the current contents of the rowset are discarded and the rowset's metadata is also (re)set. If there are outstanding updates, they are ignored.

13.11.11.2.9 Method execute(Connection)

```
Synopsis: public void execute(java.sql.Connection _conn)
                 throws java.sql.SQLException;
```

Parameters

`_conn` - a Connection to use

Exceptions

SQLException if a database-access error occurs.

Populates the rowset with data. Uses an existing JDBC connection object. The values of the url/data source name, user, and password properties are ignored. The command specified by the command property is executed to retrieve the data. The current contents of the rowset are discarded and the rowset's metadata is also (re)set. If there are outstanding updates, they are also ignored.

13.11.11.2.10 Method findColumn(String)

```
Synopsis: public int findColumn(java.lang.String columnName)
                 throws java.sql.SQLException;
```

Parameters

`columnName` - the name of the column

`return` - the column index

Exceptions

SQLException if a database-access error occurs.

Map a Resultset column name to a ResultSet column index.

13.11.11.2.11 Method first()

```
Synopsis: public boolean first() throws java.sql.SQLException;
```

Exceptions

SQLException if a database-access error occurs, or result set type is TYPE_FORWARD_ONLY.

Moves to the first row in the result set.

13.11.11.2.12 Method getArray(int)

```
Synopsis: public Array getArray(int colIndex) throws java.sql.SQLException;
```

Parameters

`colIndex` - the first column is 1, the second is 2, ...

`return` - an object representing an SQL array

Get an array column.

13.11.11.2.13 Method getArray(String)

```
Synopsis: public Array getArray(java.lang.String colName)
                 throws java.sql.SQLException;
```

Parameters

`colName` - the column name

`return` - an object representing an SQL array

Get an array column.

13.11.11.2.14 Method getAsciiStream(int)

```
Synopsis: public InputStream getAsciiStream(int columnIndex)
                 throws java.sql.SQLException;
```

Parameters

`columnIndex` - the first column is 1, the second is 2, ...

`return` - a Java input stream that delivers the database column value as a stream of one byte ASCII characters. If the value is SQL NULL then the result is null.

Exceptions

SQLException if a database-access error occurs.

A column value can be retrieved as a stream of ASCII characters and then read in chunks from the stream. This method is particularly suitable for retrieving large LONGVARCHAR values. The JDBC driver will do any necessary conversion from the database format into ASCII.

Note: All the data in the returned stream must be read prior to getting the value of any other column. The next call to a get method implicitly closes the stream. . Also, a stream may return 0 for available() whether there is data available or not.

13.11.11.2.15 Method getAsciiStream(String)

```
Synopsis: public InputStream getAsciiStream(java.lang.String columnName)
                 throws java.sql.SQLException;
```

Parameters

`columnName` - is the SQL name of the column

`return` - a Java input stream that delivers the database column value as a stream of one byte ASCII characters. If the value is SQL NULL then the result is null.

Exceptions

SQLException if a database-access error occurs.

A column value can be retrieved as a stream of ASCII characters and then read in chunks from the stream. This method is particularly suitable for retrieving large LONGVARCHAR values. The JDBC driver will do any necessary conversion from the database format into ASCII.

Note: All the data in the returned stream must be read prior to getting the value of any other column. The next call to a get method implicitly closes the stream.

13.11.11.2.16 Method getBigDecimal(int)

```
Synopsis: public BigDecimal getBigDecimal(int columnIndex)
                throws java.sql.SQLException;
```

Parameters

`columnIndex` - the first column is 1, the second is 2, ...

`return` - the column value (full precision); if the value is SQL NULL, the result is null

Exceptions

SQLException if a database-access error occurs.

Get the value of a column in the current row as a java.math.BigDecimal object.

13.11.11.2.17 Method getBigDecimal(int, int)

```
Synopsis: public BigDecimal getBigDecimal(int columnIndex, int scale)
                throws java.sql.SQLException;
```

Parameters

`columnIndex` - the first column is 1, the second is 2, ...

`scale` - the number of digits to the right of the decimal

`return` - the column value; if the value is SQL NULL, the result is null

Exceptions

SQLException if a database-access error occurs.

13.11.12 Deprecated

Get the value of a column in the current row as a java.math.BigDecimal object.

13.11.12.1 Method getBigDecimal(String)

```
Synopsis: public BigDecimal getBigDecimal(java.lang.String columnName)
                throws java.sql.SQLException;
```

Get the value of a column in the current row as a java.math.BigDecimal object.

13.11.12.1.1 Method getBigDecimal(String, int)

```
Synopsis: public BigDecimal getBigDecimal(java.lang.String columnName,
                                          int scale)
                throws java.sql.SQLException;
```

Parameters

`columnName` - is the SQL name of the column

Parameters

`scale` - the number of digits to the right of the decimal

`return` - the column value; if the value is SQL NULL, the result is null

Exceptions

SQLException if a database-access error occurs.

13.11.13 Deprecated

Get the value of a column in the current row as a java.math.BigDecimal object.

13.11.13.1 Method getBinaryStream(int)

```
Synopsis: public InputStream getBinaryStream(int columnIndex)
                throws java.sql.SQLException;
```

Parameters

`columnIndex` - the first column is 1, the second is 2, ...

`return` - a Java input stream that delivers the database column value as a stream of uninterpreted bytes. If the value is SQL NULL then the result is null.

Exceptions

SQLException if a database-access error occurs.

A column value can be retrieved as a stream of uninterpreted bytes and then read in chunks from the stream. This method is particularly suitable for retrieving large LONGVARBINARY values.

Note: All the data in the returned stream must be read prior to getting the value of any other column. The next call to a get method implicitly closes the stream. Also, a stream may return 0 for available() whether there is data available or not.

13.11.13.1.1 Method getBinaryStream(String)

```
Synopsis: public InputStream getBinaryStream(java.lang.String columnName)
                throws java.sql.SQLException;
```

Parameters

`columnName` - is the SQL name of the column

`return` - a Java input stream that delivers the database column value as a stream of uninterpreted bytes. If the value is SQL NULL then the result is null.

Exceptions

SQLException if a database-access error occurs.

A column value can be retrieved as a stream of uninterpreted bytes and then read in chunks from the stream. This method is particularly suitable for retrieving large LONGVARBINARY values.

Note: All the data in the returned stream must be read prior to getting the value of any other column. The next call to a get method implicitly closes the stream.

13.11.13.1.2 Method getBlob(int)

```
Synopsis: public Blob getBlob(int colIndex) throws java.sql.SQLException;
```

Parameters

`colIndex` - the first column is 1, the second is 2, ...

`return` - an object representing a BLOB

Get a BLOB column.

13.11.13.1.3 Method getBlob(String)

```
Synopsis: public Blob getBlob(java.lang.String colName)
              throws java.sql.SQLException;
```

Parameters

`colName` - the column name

`return` - an object representing a BLOB

Get a BLOB column.

13.11.13.1.4 Method getBoolean(int)

```
Synopsis: public boolean getBoolean(int columnIndex)
              throws java.sql.SQLException;
```

Parameters

`columnIndex` - the first column is 1, the second is 2, ...

`return` - the column value; if the value is SQL NULL, the result is false

Exceptions

SQLException if a database-access error occurs.

Get the value of a column in the current row as a Java boolean.

13.11.13.1.5 Method getBoolean(String)

```
Synopsis: public boolean getBoolean(java.lang.String columnName)
              throws java.sql.SQLException;
```

Parameters

`columnName` - is the SQL name of the column

`return` - the column value; if the value is SQL NULL, the result is false

Exceptions

SQLException if a database-access error occurs.

Get the value of a column in the current row as a Java boolean.

13.11.13.1.6 Method getByte(int)

```
Synopsis: public byte getByte(int columnIndex) throws java.sql.SQLException;
```

Parameters

`columnIndex` - the first column is 1, the second is 2, ...

`return` - the column value; if the value is SQL NULL, the result is 0

Exceptions

SQLException if a database-access error occurs.

Get the value of a column in the current row as a Java byte.

13.11.13.1.7 Method getByte(String)

```
Synopsis: public byte getByte(java.lang.String columnName)
              throws java.sql.SQLException;
```

Parameters

`columnName` - is the SQL name of the column

`return` - the column value; if the value is SQL NULL, the result is 0

Exceptions

Exceptions

SQLException if a database-access error occurs.

Get the value of a column in the current row as a Java byte.

13.11.13.1.8 Method getBytes(int)

```
Synopsis: public byte[] getBytes(int columnIndex) throws java.sql.SQLException;
```

Parameters

`columnIndex` - the first column is 1, the second is 2, ...

`return` - the column value; if the value is SQL NULL, the result is null

Exceptions

SQLException if a database-access error occurs.

Get the value of a column in the current row as a Java byte array. The bytes represent the raw values returned by the driver.

13.11.13.1.9 Method getBytes(String)

```
Synopsis: public byte[] getBytes(java.lang.String columnName)
              throws java.sql.SQLException;
```

Parameters

`columnName` - is the SQL name of the column

`return` - the column value; if the value is SQL NULL, the result is null

Exceptions

SQLException if a database-access error occurs.

Get the value of a column in the current row as a Java byte array. The bytes represent the raw values returned by the driver.

13.11.13.1.10 Method getCharacterStream(int)

```
Synopsis: public Reader getCharacterStream(int columnIndex)
              throws java.sql.SQLException;
```

Get the value of a column in the current row as a java.io.Reader.

13.11.13.1.11 Method getCharacterStream(String)

```
Synopsis: public Reader getCharacterStream(java.lang.String columnName)
              throws java.sql.SQLException;
```

Get the value of a column in the current row as a java.io.Reader.

13.11.13.1.12 Method getClob(int)

```
Synopsis: public Clob getClob(int colIndex) throws java.sql.SQLException;
```

Parameters

`colIndex` - the first column is 1, the second is 2, ...

`return` - an object representing a CLOB

Get a CLOB column.

13.11.13.1.13 Method getClob(String)

```
Synopsis: public Clob getClob(java.lang.String colName)
                throws java.sql.SQLException;
```

Parameters

`colName` - the column name

`return` - an object representing a CLOB

Get a CLOB column.

13.11.13.1.14 Method getCursorName()

```
Synopsis: public String getCursorName() throws java.sql.SQLException;
```

Exceptions

SQLException if an error occurs.

Get the name of the SQL cursor used by this ResultSet.

13.11.13.1.15 Method getDate(int)

```
Synopsis: public Date getDate(int columnIndex) throws java.sql.SQLException;
```

Parameters

`columnIndex` - the first column is 1, the second is 2, ...

`return` - the column value; if the value is SQL NULL, the result is null

Exceptions

SQLException if a database-access error occurs.

Get the value of a column in the current row as a java.sql.Date object.

13.11.13.1.16 Method getDate(int, Calendar)

```
Synopsis: public Date getDate(int columnIndex, java.util.Calendar cal)
                throws java.sql.SQLException;
```

Parameters

`columnIndex` - the first column is 1, the second is 2, ...

`cal` - the calendar to use in constructing the date

`return` - the column value; if the value is SQL NULL, the result is null

Exceptions

SQLException if a database-access error occurs.

Get the value of a column in the current row as a java.sql.Date object. Use the calendar to construct an appropriate millisecond value for the Date, if the underlying database doesn't store timezone information.

13.11.13.1.17 Method getDate(String)

```
Synopsis: public Date getDate(java.lang.String columnName)
                throws java.sql.SQLException;
```

Parameters

`columnName` - is the SQL name of the column

`return` - the column value; if the value is SQL NULL, the result is null

Exceptions

SQLException if a database-access error occurs.

Get the value of a column in the current row as a java.sql.Date object.

13.11.13.1.18 Method getDate(String, Calendar)

```
Synopsis: public Date getDate(java.lang.String columnName,
                              java.util.Calendar cal)
              throws java.sql.SQLException;
```

Parameters

columnName - is the SQL name of the column

cal - the calendar to use in constructing the date

return - the column value; if the value is SQL NULL, the result is null

Exceptions

SQLException if a database-access error occurs.

Get the value of a column in the current row as a java.sql.Date object. Use the calendar to construct an appropriate millisecond value for the Date, if the underlying database doesn't store timezone information.

13.11.13.1.19 Method getDouble(int)

```
Synopsis: public double getDouble(int columnIndex)
              throws java.sql.SQLException;
```

Parameters

columnIndex - the first column is 1, the second is 2, ...

return - the column value; if the value is SQL NULL, the result is 0

Exceptions

SQLException if a database-access error occurs.

Get the value of a column in the current row as a Java double.

13.11.13.1.20 Method getDouble(String)

```
Synopsis: public double getDouble(java.lang.String columnName)
              throws java.sql.SQLException;
```

Parameters

columnName - is the SQL name of the column

return - the column value; if the value is SQL NULL, the result is 0

Exceptions

SQLException if a database-access error occurs.

Get the value of a column in the current row as a Java double.

13.11.13.1.21 Method getFloat(int)

```
Synopsis: public float getFloat(int columnIndex) throws java.sql.SQLException;
```

Parameters

columnIndex - the first column is 1, the second is 2, ...

return - the column value; if the value is SQL NULL, the result is 0

Exceptions

SQLException if a database-access error occurs.

Get the value of a column in the current row as a Java float.

13.11.13.1.22 Method getFloat(String)

```
Synopsis: public float getFloat(java.lang.String columnName)
              throws java.sql.SQLException;
```

Parameters

`columnName` - is the SQL name of the column

`return` - the column value; if the value is SQL NULL, the result is 0

Exceptions

SQLException if a database-access error occurs.

Get the value of a column in the current row as a Java float.

13.11.13.1.23 Method getInt(int)

```
Synopsis: public int getInt(int columnIndex) throws java.sql.SQLException;
```

Parameters

`columnIndex` - the first column is 1, the second is 2, ...

`return` - the column value; if the value is SQL NULL, the result is 0

Exceptions

SQLException if a database-access error occurs.

Get the value of a column in the current row as a Java int.

13.11.13.1.24 Method getInt(String)

```
Synopsis: public int getInt(java.lang.String columnName)
              throws java.sql.SQLException;
```

Parameters

`columnName` - is the SQL name of the column

`return` - the column value; if the value is SQL NULL, the result is 0

Exceptions

SQLException if a database-access error occurs.

Get the value of a column in the current row as a Java int.

13.11.13.1.25 Method getLong(int)

```
Synopsis: public long getLong(int columnIndex) throws java.sql.SQLException;
```

Parameters

`columnIndex` - the first column is 1, the second is 2, ...

`return` - the column value; if the value is SQL NULL, the result is 0

Exceptions

SQLException if a database-access error occurs.

Get the value of a column in the current row as a Java long.

13.11.13.1.26 Method getLong(String)

```
Synopsis: public long getLong(java.lang.String columnName)
              throws java.sql.SQLException;
```

Parameters

`columnName` - is the SQL name of the column

`return` - the column value; if the value is SQL NULL, the result is 0

Exceptions

SQLException if a database-access error occurs.

Get the value of a column in the current row as a Java long.

13.11.13.1.27 Method getMetaData()

```
Synopsis: public ResultSetMetaData getMetaData() throws java.sql.SQLException;
```

Exceptions

SQLException if a database-access error occurs.

The number, types and properties of a ResultSet's columns are provided by the getMetaData method.

13.11.13.1.28 Method getObject(int)

```
Synopsis: public Object getObject(int columnIndex)
                 throws java.sql.SQLException;
```

Parameters

`columnIndex` - the first column is 1, the second is 2, ...

`return` - a java.lang.Object holding the column value.

Exceptions

SQLException if a database-access error occurs.

Get the value of a column in the current row as a Java object.

This method will return the value of the given column as a Java object. The type of the Java object will be the default Java object type corresponding to the column's SQL type, following the mapping for built-in types specified in the JDBC spec.

This method may also be used to read database specific abstract data types. JDBC 2.0 New behavior for getObject(). The behavior of method getObject() is extended to materialize data of SQL user-defined types. When the column @column is a structured or distinct value, the behavior of this method is as if it were a call to: getObject(column, this.getStatement().getConnection().getTypeMap()).

13.11.13.1.29 Method getObject(int, Map)

```
Synopsis: public Object getObject(int colIndex, java.util.Map map)
                 throws java.sql.SQLException;
```

Parameters

`colIndex` - the first column is 1, the second is 2, ...

`map` - the mapping from SQL type names to Java classes

`return` - an object representing the SQL value

Returns the value of column @i as a Java object. Use the map to determine the class from which to construct data of SQL structured and distinct types.

13.11.13.1.30 Method getObject(String)

```
Synopsis: public Object getObject(java.lang.String columnName)
                 throws java.sql.SQLException;
```

Parameters

`columnName` - is the SQL name of the column

`return` - a java.lang.Object holding the column value.

Exceptions

SQLException if a database-access error occurs.

Get the value of a column in the current row as a Java object.

This method will return the value of the given column as a Java object. The type of the Java object will be the default Java object type corresponding to the column's SQL type, following the mapping for built-in types specified in the JDBC spec.

This method may also be used to read database specific abstract data types. JDBC 2.0 New behavior for getObject(). The behavior of method getObject() is extended to materialize data of SQL user-defined types. When the column @columnName is a structured or distinct value, the behavior of this method is as if it were a call to: getObject(columnName, this.getStatement().getConnection().getTypeMap()).

13.11.13.1.31 Method getObject(String, Map)

```
Synopsis: public Object getObject(java.lang.String colName, java.util.Map map)
              throws java.sql.SQLException;
```

Parameters

`colName` - the column name

`map` - the mapping from SQL type names to Java classes

`return` - an object representing the SQL value

Returns the value of column @i as a Java object. Use the map to determine the class from which to construct data of SQL structured and distinct types.

13.11.13.1.32 Method getRef(int)

```
Synopsis: public Ref getRef(int colIndex) throws java.sql.SQLException;
```

Parameters

`colIndex` - the first column is 1, the second is 2, ...

`return` - an object representing data of an SQL REF type

Get a REF(<structured-type>) column.

13.11.13.1.33 Method getRef(String)

```
Synopsis: public Ref getRef(java.lang.String colName)
              throws java.sql.SQLException;
```

Parameters

`colName` - the column name

`return` - an object representing data of an SQL REF type

Get a REF(<structured-type>) column.

13.11.13.1.34 Method getRow()

```
Synopsis: public int getRow() throws java.sql.SQLException;
```

Exceptions

SQLException if a database-access error occurs.

Determine the current row number. The first row is number 1, the second number 2, etc.

13.11.13.1.35 Method getShort(int)

```
Synopsis: public short getShort(int columnIndex) throws java.sql.SQLException;
```

Parameters

`columnIndex` - the first column is 1, the second is 2, ...

`return` - the column value; if the value is SQL NULL, the result is 0

Exceptions

SQLException if a database-access error occurs.

Get the value of a column in the current row as a Java short.

13.11.13.1.36 Method getShort(String)

```
Synopsis: public short getShort(java.lang.String columnName)
              throws java.sql.SQLException;
```

Parameters

`columnName` - is the SQL name of the column

`return` - the column value; if the value is SQL NULL, the result is 0

Exceptions

SQLException if a database-access error occurs.

Get the value of a column in the current row as a Java short.

13.11.13.1.37 Method getStatement()

```
Synopsis: public Statement getStatement() throws java.sql.SQLException;
```

Exceptions

SQLException if a database-access error occurs

Return the Statement that produced the ResultSet.

13.11.13.1.38 Method getString(int)

```
Synopsis: public String getString(int columnIndex)
              throws java.sql.SQLException;
```

Parameters

`columnIndex` - the first column is 1, the second is 2, ...

`return` - the column value; if the value is SQL NULL, the result is null

Exceptions

SQLException if a database-access error occurs.

Get the value of a column in the current row as a Java String.

13.11.13.1.39 Method getString(String)

```
Synopsis: public String getString(java.lang.String columnName)
              throws java.sql.SQLException;
```

Parameters

`columnName` - is the SQL name of the column

`return` - the column value; if the value is SQL NULL, the result is null

Exceptions

SQLException if a database-access error occurs.

Get the value of a column in the current row as a Java String.

13.11.13.1.40 Method getTime(int)

```
Synopsis: public Time getTime(int columnIndex) throws java.sql.SQLException;
```

Parameters

`columnIndex` - the first column is 1, the second is 2, ...

`return` - the column value; if the value is SQL NULL, the result is null

Exceptions

SQLException if a database-access error occurs.

Get the value of a column in the current row as a java.sql.Time object.

13.11.13.1.41 Method getTime(int, Calendar)

```
Synopsis: public Time getTime(int columnIndex, java.util.Calendar cal)
              throws java.sql.SQLException;
```

Parameters

`columnIndex` - the first column is 1, the second is 2, ...

`cal` - the calendar to use in constructing the time

`return` - the column value; if the value is SQL NULL, the result is null

Exceptions

SQLException if a database-access error occurs.

Get the value of a column in the current row as a java.sql.Time object. Use the calendar to construct an appropriate millisecond value for the Time, if the underlying database doesn't store timezone information.

13.11.13.1.42 Method getTime(String)

```
Synopsis: public Time getTime(java.lang.String columnName)
              throws java.sql.SQLException;
```

Parameters

`columnName` - is the SQL name of the column

`return` - the column value; if the value is SQL NULL, the result is null

Exceptions

SQLException if a database-access error occurs.

Get the value of a column in the current row as a java.sql.Time object.

13.11.13.1.43 Method getTime(String, Calendar)

```
Synopsis: public Time getTime(java.lang.String columnName,
                              java.util.Calendar cal)
              throws java.sql.SQLException;
```

Parameters

`columnName` - is the SQL name of the column

`cal` - the calendar to use in constructing the time

`return` - the column value; if the value is SQL NULL, the result is null

Exceptions

SQLException if a database-access error occurs.

Get the value of a column in the current row as a java.sql.Time object. Use the calendar to construct an appropriate millisecond value for the Time, if the underlying database doesn't store timezone information.

13.11.13.1.44 Method getTimestamp(int)

```
Synopsis: public Timestamp getTimestamp(int columnIndex)
              throws java.sql.SQLException;
```

Parameters

`columnIndex` - the first column is 1, the second is 2, ...

`return` - the column value; if the value is SQL NULL, the result is null

Exceptions

SQLException if a database-access error occurs.

Get the value of a column in the current row as a java.sql.Timestamp object.

13.11.13.1.45 Method getTimestamp(int, Calendar)

```
Synopsis: public Timestamp getTimestamp(int columnIndex,
                                        java.util.Calendar cal)
               throws java.sql.SQLException;
```

Parameters

`columnIndex` - the first column is 1, the second is 2, ...

`cal` - the calendar to use in constructing the timestamp

`return` - the column value; if the value is SQL NULL, the result is null

Exceptions

SQLException if a database-access error occurs.

Get the value of a column in the current row as a java.sql.Timestamp object. Use the calendar to construct an appropriate millisecond value for the Timestamp, if the underlying database doesn't store timezone information.

13.11.13.1.46 Method getTimestamp(String)

```
Synopsis: public Timestamp getTimestamp(java.lang.String columnName)
               throws java.sql.SQLException;
```

Parameters

`columnName` - is the SQL name of the column

`return` - the column value; if the value is SQL NULL, the result is null

Exceptions

SQLException if a database-access error occurs.

Get the value of a column in the current row as a java.sql.Timestamp object.

13.11.13.1.47 Method getTimestamp(String, Calendar)

```
Synopsis: public Timestamp getTimestamp(java.lang.String columnName,
                                        java.util.Calendar cal)
               throws java.sql.SQLException;
```

Parameters

`columnName` - is the SQL name of the column

`cal` - the calendar to use in constructing the timestamp

`return` - the column value; if the value is SQL NULL, the result is null

Exceptions

SQLException if a database-access error occurs.

Get the value of a column in the current row as a java.sql.Timestamp object. Use the calendar to construct an appropriate millisecond value for the Timestamp, if the underlying database doesn't store timezone information.

13.11.13.1.48 Method getUnicodeStream(int)

```
Synopsis: public InputStream getUnicodeStream(int columnIndex)
               throws java.sql.SQLException;
```

Parameters

`columnIndex` - the first column is 1, the second is 2, ...

`return` - a Java input stream that delivers the database column value as a stream of two byte Unicode characters. If the value is SQL NULL then the result is null.

Exceptions

SQLException if a database-access error occurs.

13.11.14 Deprecated

A column value can be retrieved as a stream of Unicode characters and then read in chunks from the stream. This method is particularly suitable for retrieving large LONGVARCHAR values. The JDBC driver will do any necessary conversion from the database format into Unicode.

Note: All the data in the returned stream must be read prior to getting the value of any other column. The next call to a get method implicitly closes the stream. . Also, a stream may return 0 for available() whether there is data available or not.

13.11.14.1 Method getUnicodeStream(String)

```
Synopsis: public InputStream getUnicodeStream(java.lang.String columnName)
                  throws java.sql.SQLException;
```

Parameters

`columnName` - is the SQL name of the column

`return` - a Java input stream that delivers the database column value as a stream of two byte Unicode characters. If the value is SQL NULL then the result is null.

Exceptions

SQLException if a database-access error occurs.

13.11.15 Deprecated

A column value can be retrieved as a stream of Unicode characters and then read in chunks from the stream. This method is particularly suitable for retrieving large LONGVARCHAR values. The JDBC driver will do any necessary conversion from the database format into Unicode.

Note: All the data in the returned stream must be read prior to getting the value of any other column. The next call to a get method implicitly closes the stream.

13.11.15.1 Method getURL(int)

```
Synopsis: public URL getURL(int columnIndex) throws java.sql.SQLException;
```

Parameters

`columnIndex` - the index of the column 1 is the first, 2 is the second,...

`return` - the column value as a `java.net.URL` object; if the value is SQL `NULL`, the value returned is `null` in the Java programming language

Exceptions

SQLException if a database access error occurs, or if a URL is malformed

Additional Information

Since 1.4

Retrieves the value of the designated column in the current row of this `ResultSet` object as a `java.net.URL` object in the Java programming language.

13.11.15.1.1 Method getURL(String)

```
Synopsis: public URL getURL(java.lang.String columnName)
                  throws java.sql.SQLException;
```

Parameters

`columnName` - the SQL name of the column

`return` - the column value as a `java.net.URL` object; if the value is SQL `NULL`, the value returned is `null` in the Java programming language

Exceptions

Exceptions

SQLException	if a database access error occurs or if a URL is malformed

Additional Information

Since	1.4

Retrieves the value of the designated column in the current row of this `ResultSet` object as a `java.net.URL` object in the Java programming language.

13.11.15.1.2 Method getWarnings()

```
Synopsis: public SQLWarning getWarnings() throws java.sql.SQLException;
```

Exceptions

SQLException	if a database-access error occurs.

The first warning reported by calls on this ResultSet is returned. Subsequent ResultSet warnings will be chained to this SQLWarning.

The warning chain is automatically cleared each time a new row is read.

Note: This warning chain only covers warnings caused by ResultSet methods. Any warning caused by statement methods (such as reading OUT parameters) will be chained on the Statement object.

13.11.15.1.3 Method insertRow()

```
Synopsis: public void insertRow() throws java.sql.SQLException;
```

Exceptions

SQLException	if a database access error occurs, if this method is called when the cursor is not on the insert row, or if not all of non-nullable columns in the insert row have been given a value

Inserts the contents of the insert row into this `ResultSet` objaect and into the database. The cursor must be on the insert row when this method is called.

13.11.15.1.4 Method isAfterLast()

```
Synopsis: public boolean isAfterLast() throws java.sql.SQLException;
```

Exceptions

SQLException	if a database-access error occurs.

Determine if the cursor is after the last row in the result set.

13.11.15.1.5 Method isBeforeFirst()

```
Synopsis: public boolean isBeforeFirst() throws java.sql.SQLException;
```

Exceptions

SQLException	if a database-access error occurs.

Determine if the cursor is before the first row in the result set.

13.11.15.1.6 Method isFirst()

```
Synopsis: public boolean isFirst() throws java.sql.SQLException;
```

Exceptions

SQLException	if a database-access error occurs.

Determine if the cursor is on the first row of the result set.

13.11.15.1.7 Method isLast()

```
Synopsis: public boolean isLast() throws java.sql.SQLException;
```

Exceptions

SQLException	if a database-access error occurs.

Determine if the cursor is on the last row of the result set. Note: Calling isLast() may be expensive since the JDBC driver might need to fetch ahead one row in order to determine whether the current row is the last row in the result set.

13.11.15.1.8 Method last()

```
Synopsis: public boolean last() throws java.sql.SQLException;
```

Exceptions

SQLException	if a database-access error occurs, or result set type is TYPE_FORWARD_ONLY.

Moves to the last row in the result set.

13.11.15.1.9 Method moveToCurrentRow()

```
Synopsis: public void moveToCurrentRow() throws java.sql.SQLException;
```

Exceptions

SQLException	if a database-access error occurs, or the result set is not updatable

Move the cursor to the remembered cursor position, usually the current row. Has no effect unless the cursor is on the insert row.

13.11.15.1.10 Method moveToInsertRow()

```
Synopsis: public void moveToInsertRow() throws java.sql.SQLException;
```

Exceptions

SQLException	if a database-access error occurs, or the result set is not updatable

Move to the insert row. The current cursor position is remembered while the cursor is positioned on the insert row. The insert row is a special row associated with an updatable result set. It is essentially a buffer where a new row may be constructed by calling the updateXXX() methods prior to inserting the row into the result set. Only the updateXXX(), getXXX(), and insertRow() methods may be called when the cursor is on the insert row. All of the columns in a result set must be given a value each time this method is called before calling insertRow(). UpdateXXX()must be called before getXXX() on a column.

13.11.15.1.11 Method next()

```
Synopsis: public boolean next() throws java.sql.SQLException;
```

Exceptions

SQLException	if a database-access error occurs.

A ResultSet is initially positioned before its first row; the first call to next makes the first row the current row; the second call makes the second row the current row, etc.

If an input stream from the previous row is open, it is implicitly closed. The ResultSet's warning chain is cleared when a new row is read.

13.11.15.1.12 Method previous()

```
Synopsis: public boolean previous() throws java.sql.SQLException;
```

Exceptions

SQLException	if a database-access error occurs, or result set type is TYPE_FORWAR_DONLY.

Moves to the previous row in the result set.

Note: previous() is not the same as relative(-1) since it makes sense to call previous() when there is no current row.

13.11.15.1.13 Method refreshRow()

```
Synopsis: public void refreshRow() throws java.sql.SQLException;
```

Exceptions

SQLException	if a database-access error occurs, or if called when on the insert row.

Refresh the value of the current row with its current value in the database. Cannot be called when on the insert row. The refreshRow() method provides a way for an application to explicitly tell the JDBC driver to refetch a row(s) from the database. An application may want to call refreshRow() when caching or prefetching is being done by the JDBC driver to fetch the latest value of a row from the database. The JDBC driver may actually refresh multiple rows at once if the fetch size is greater than one. All values are refetched subject to the transaction isolation level and cursor sensitivity. If refreshRow() is called after calling updateXXX(), but before calling updateRow() then the updates made to the row are lost. Calling refreshRow() frequently will likely slow performance.

13.11.15.1.14 Method relative(int)

```
Synopsis: public boolean relative(int rows) throws java.sql.SQLException;
```

Exceptions

SQLException	if a database-access error occurs, or there is no current row, or result set type is TYPE_FORWARD_ONLY.

Moves a relative number of rows, either positive or negative. Attempting to move beyond the first/last row in the result set positions the cursor before/after the the first/last row. Calling relative(0) is valid, but does not change the cursor position.

Note: Calling relative(1) is different than calling next() since is makes sense to call next() when there is no current row, for example, when the cursor is positioned before the first row or after the last row of the result set.

13.11.15.1.15 Method rowDeleted()

```
Synopsis: public boolean rowDeleted() throws java.sql.SQLException;
```

Exceptions

SQLException	if a database-access error occurs

Additional Information

See Also	java.sql.DatabaseMetaData.deletesAreDetected

Determine if this row has been deleted. A deleted row may leave a visible "hole" in a result set. This method can be used to detect holes in a result set. The value returned depends on whether or not the result set can detect deletions.

13.11.15.1.16 Method rowInserted()

```
Synopsis: public boolean rowInserted() throws java.sql.SQLException;
```

Exceptions

SQLException	if a database-access error occurs

Additional Information

See Also	java.sql.DatabaseMetaData.insertsAreDetected

Determine if the current row has been inserted. The value returned depends on whether or not the result set can detect visible inserts.

13.11.15.1.17 Method rowUpdated()

```
Synopsis: public boolean rowUpdated() throws java.sql.SQLException;
```

Exceptions

SQLException	if a database-access error occurs

Additional Information

See Also	java.sql.DatabaseMetaData.updatesAreDetected

Determine if the current row has been updated. The value returned depends on whether or not the result set can detect updates.

13.11.15.1.18 Method updateArray(int, Array)

```
Synopsis: public void updateArray(int columnIndex, java.sql.Array x)
                throws java.sql.SQLException;
```

Parameters

`columnIndex` - the first column is 1, the second is 2, ...

`x` - the new column value

Exceptions

SQLException	if a database access error occurs

Additional Information

Since	1.4

Updates the designated column with a `java.sql.Array` value. The updater methods are used to update column values in the current row or the insert row. The updater methods do not update the underlying database; instead the `updateRow` or `insertRow` methods are called to update the database.

13.11.15.1.19 Method updateArray(String, Array)

```
Synopsis: public void updateArray(java.lang.String columnName,
                                  java.sql.Array x)
                throws java.sql.SQLException;
```

Parameters

`columnName` - the name of the column

`x` - the new column value

Exceptions

SQLException	if a database access error occurs

Additional Information

Since	1.4

Updates the designated column with a `java.sql.Array` value. The updater methods are used to update column values in the current row or the insert row. The updater methods do not update the underlying database; instead the `updateRow` or `insertRow` methods are called to update the database.

13.11.15.1.20 Method updateAsciiStream(int, InputStream, int)

```
Synopsis: public void updateAsciiStream(int columnIndex, java.io.InputStream x,
                                        int length)
                throws java.sql.SQLException;
```

Parameters

`columnIndex` - the first column is 1, the second is 2, ...

`x` - the new column value

Parameters

length - the length of the stream

Exceptions

SQLException if a database-access error occurs

Update a column with an ascii stream value. The updateXXX() methods are used to update column values in the current row, or the insert row. The updateXXX() methods do not update the underlying database, instead the updateRow() or insertRow() methods are called to update the database.

13.11.15.1.21 Method updateAsciiStream(String, InputStream, int)

```
Synopsis: public void updateAsciiStream(java.lang.String columnName,
                                         java.io.InputStream x,
                                         int length)
              throws java.sql.SQLException;
```

Parameters

columnName - the name of the column

x - the new column value

length - of the stream

Exceptions

SQLException if a database-access error occurs

Update a column with an ascii stream value. The updateXXX() methods are used to update column values in the current row, or the insert row. The updateXXX() methods do not update the underlying database, instead the updateRow() or insertRow() methods are called to update the database.

13.11.15.1.22 Method updateBigDecimal(int, BigDecimal)

```
Synopsis: public void updateBigDecimal(int columnIndex, java.math.BigDecimal x)
              throws java.sql.SQLException;
```

Parameters

columnIndex - the first column is 1, the second is 2, ...

x - the new column value

Exceptions

SQLException if a database-access error occurs

Update a column with a BigDecimal value. The updateXXX() methods are used to update column values in the current row, or the insert row. The updateXXX() methods do not update the underlying database, instead the updateRow() or insertRow() methods are called to update the database.

13.11.15.1.23 Method updateBigDecimal(String, BigDecimal)

```
Synopsis: public void updateBigDecimal(java.lang.String columnName,
                                        java.math.BigDecimal x)
              throws java.sql.SQLException;
```

Parameters

columnName - the name of the column

x - the new column value

Exceptions

SQLException if a database-access error occurs

Update a column with a BigDecimal value. The updateXXX() methods are used to update column values in the current row, or the insert row. The updateXXX() methods do not update the underlying database, instead the updateRow() or insertRow() methods are called to update the database.

13.11.15.1.24 Method updateBinaryStream(int, InputStream, int)

```
Synopsis: public void updateBinaryStream(int columnIndex,
                                         java.io.InputStream x, int length)
              throws java.sql.SQLException;
```

Parameters

columnIndex - the first column is 1, the second is 2, ...

x - the new column value

length - the length of the stream

Exceptions

SQLException if a database-access error occurs

Update a column with a binary stream value. The updateXXX() methods are used to update column values in the current row, or the insert row. The updateXXX() methods do not update the underlying database, instead the updateRow() or insertRow() methods are called to update the database.

13.11.15.1.25 Method updateBinaryStream(String, InputStream, int)

```
Synopsis: public void updateBinaryStream(java.lang.String columnName,
                                         java.io.InputStream x,
                                         int length)
              throws java.sql.SQLException;
```

Parameters

columnName - the name of the column

x - the new column value

length - of the stream

Exceptions

SQLException if a database-access error occurs

Update a column with a binary stream value. The updateXXX() methods are used to update column values in the current row, or the insert row. The updateXXX() methods do not update the underlying database, instead the updateRow() or insertRow() methods are called to update the database.

13.11.15.1.26 Method updateBlob(int, Blob)

```
Synopsis: public void updateBlob(int columnIndex, java.sql.Blob x)
              throws java.sql.SQLException;
```

Parameters

columnIndex - the first column is 1, the second is 2, ...

x - the new column value

Exceptions

SQLException if a database access error occurs

Additional Information

Since 1.4

Updates the designated column with a `java.sql.Blob` value. The updater methods are used to update column values in the current row or the insert row. The updater methods do not update the underlying database; instead the `updateRow` or `insertRow` methods are called to update the database.

13.11.15.1.27 Method updateBlob(String, Blob)

```
Synopsis: public void updateBlob(java.lang.String columnName, java.sql.Blob x)
              throws java.sql.SQLException;
```

Parameters

`columnName` - the name of the column

`x` - the new column value

Exceptions

SQLException if a database access error occurs

Additional Information

Since 1.4

Updates the designated column with a `java.sql.Blob` value. The updater methods are used to update column values in the current row or the insert row. The updater methods do not update the underlying database; instead the `updateRow` or `insertRow` methods are called to update the database.

13.11.15.1.28 Method updateBoolean(int, boolean)

```
Synopsis: public void updateBoolean(int columnIndex, boolean x)
                throws java.sql.SQLException;
```

Parameters

`columnIndex` - the first column is 1, the second is 2, ...

`x` - the new column value

Exceptions

SQLException if a database-access error occurs

Update a column with a boolean value. The updateXXX() methods are used to update column values in the current row, or the insert row. The updateXXX() methods do not update the underlying database, instead the updateRow() or insertRow() methods are called to update the database.

13.11.15.1.29 Method updateBoolean(String, boolean)

```
Synopsis: public void updateBoolean(java.lang.String columnName, boolean x)
                throws java.sql.SQLException;
```

Parameters

`columnName` - the name of the column

`x` - the new column value

Exceptions

SQLException if a database-access error occurs

Update a column with a boolean value. The updateXXX() methods are used to update column values in the current row, or the insert row. The updateXXX() methods do not update the underlying database, instead the updateRow() or insertRow() methods are called to update the database.

13.11.15.1.30 Method updateByte(int, byte)

```
Synopsis: public void updateByte(int columnIndex, byte x)
                throws java.sql.SQLException;
```

Parameters

`columnIndex` - the first column is 1, the second is 2, ...

`x` - the new column value

Exceptions

SQLException if a database-access error occurs

Update a column with a byte value. The updateXXX() methods are used to update column values in the current row, or the insert row. The updateXXX() methods do not update the underlying database, instead the updateRow() or insertRow() methods are called to update the database.

13.11.15.1.31 Method updateByte(String, byte)

```
Synopsis: public void updateByte(java.lang.String columnName, byte x)
              throws java.sql.SQLException;
```

Parameters

columnName - the name of the column

x - the new column value

Exceptions

SQLException if a database-access error occurs

Update a column with a byte value. The updateXXX() methods are used to update column values in the current row, or the insert row. The updateXXX() methods do not update the underlying database, instead the updateRow() or insertRow() methods are called to update the database.

13.11.15.1.32 Method updateBytes(int, byte[])

```
Synopsis: public void updateBytes(int columnIndex, byte[] x)
              throws java.sql.SQLException;
```

Parameters

columnIndex - the first column is 1, the second is 2, ...

x - the new column value

Exceptions

SQLException if a database-access error occurs

Update a column with a byte array value. The updateXXX() methods are used to update column values in the current row, or the insert row. The updateXXX() methods do not update the underlying database, instead the updateRow() or insertRow() methods are called to update the database.

13.11.15.1.33 Method updateBytes(String, byte[])

```
Synopsis: public void updateBytes(java.lang.String columnName, byte[] x)
              throws java.sql.SQLException;
```

Parameters

columnName - the name of the column

x - the new column value

Exceptions

SQLException if a database-access error occurs

Update a column with a byte array value. The updateXXX() methods are used to update column values in the current row, or the insert row. The updateXXX() methods do not update the underlying database, instead the updateRow() or insertRow() methods are called to update the database.

13.11.15.1.34 Method updateCharacterStream(int, Reader, int)

```
Synopsis: public void updateCharacterStream(int columnIndex, java.io.Reader x,
                                            int length)
              throws java.sql.SQLException;
```

Parameters

columnIndex - the first column is 1, the second is 2, ...

x - the new column value

length - the length of the stream

Exceptions

SQLException if a database-access error occurs

Update a column with a character stream value. The updateXXX() methods are used to update column values in the current row, or the insert row. The updateXXX() methods do not update the underlying database, instead the updateRow() or insertRow() methods are called to update the database.

13.11.15.1.35 Method updateCharacterStream(String, Reader, int)

```
Synopsis: public void updateCharacterStream(java.lang.String columnName,
                                            java.io.Reader x,
                                            int length)
              throws java.sql.SQLException;
```

Parameters

columnName	- the name of the column
x	- the new column value
length	- of the stream

Exceptions

SQLException	if a database-access error occurs

Update a column with a character stream value. The updateXXX() methods are used to update column values in the current row, or the insert row. The updateXXX() methods do not update the underlying database, instead the updateRow() or insertRow() methods are called to update the database.

13.11.15.1.36 Method updateClob(int, Clob)

```
Synopsis: public void updateClob(int columnIndex, java.sql.Clob x)
              throws java.sql.SQLException;
```

Parameters

columnIndex	- the first column is 1, the second is 2, ...
x	- the new column value

Exceptions

SQLException	if a database access error occurs

Additional Information

Since	1.4

Updates the designated column with a `java.sql.Clob` value. The updater methods are used to update column values in the current row or the insert row. The updater methods do not update the underlying database; instead the `updateRow` or `insertRow` methods are called to update the database.

13.11.15.1.37 Method updateClob(String, Clob)

```
Synopsis: public void updateClob(java.lang.String columnName, java.sql.Clob x)
              throws java.sql.SQLException;
```

Parameters

columnName	- the name of the column
x	- the new column value

Exceptions

SQLException	if a database access error occurs

Additional Information

Since	1.4

Updates the designated column with a `java.sql.Clob` value. The updater methods are used to update column values in the current row or the insert row. The updater methods do not update the underlying database; instead the `updateRow` or `insertRow` methods are called to update the database.

13.11.15.1.38 Method updateDate(int, Date)

```
Synopsis: public void updateDate(int columnIndex, java.sql.Date x)
              throws java.sql.SQLException;
```

Parameters

columnIndex - the first column is 1, the second is 2, ...

x - the new column value

Exceptions

SQLException if a database-access error occurs

Update a column with a Date value. The updateXXX() methods are used to update column values in the current row, or the insert row. The updateXXX() methods do not update the underlying database, instead the updateRow() or insertRow() methods are called to update the database.

13.11.15.1.39 Method updateDate(String, Date)

```
Synopsis: public void updateDate(java.lang.String columnName, java.sql.Date x)
              throws java.sql.SQLException;
```

Parameters

columnName - the name of the column

x - the new column value

Exceptions

SQLException if a database-access error occurs

Update a column with a Date value. The updateXXX() methods are used to update column values in the current row, or the insert row. The updateXXX() methods do not update the underlying database, instead the updateRow() or insertRow() methods are called to update the database.

13.11.15.1.40 Method updateDouble(int, double)

```
Synopsis: public void updateDouble(int columnIndex, double x)
              throws java.sql.SQLException;
```

Parameters

columnIndex - the first column is 1, the second is 2, ...

x - the new column value

Exceptions

SQLException if a database-access error occurs

Update a column with a Double value. The updateXXX() methods are used to update column values in the current row, or the insert row. The updateXXX() methods do not update the underlying database, instead the updateRow() or insertRow() methods are called to update the database.

13.11.15.1.41 Method updateDouble(String, double)

```
Synopsis: public void updateDouble(java.lang.String columnName, double x)
              throws java.sql.SQLException;
```

Parameters

columnName - the name of the column

x - the new column value

Exceptions

SQLException if a database-access error occurs

Update a column with a double value. The updateXXX() methods are used to update column values in the current row, or the insert row. The updateXXX() methods do not update the underlying database, instead the updateRow() or insertRow() methods are called to update the database.

13.11.15.1.42 Method updateFloat(int, float)

```
Synopsis: public void updateFloat(int columnIndex, float x)
                throws java.sql.SQLException;
```

Parameters

columnIndex - the first column is 1, the second is 2, ...

x - the new column value

Exceptions

SQLException if a database-access error occurs

Update a column with a float value. The updateXXX() methods are used to update column values in the current row, or the insert row. The updateXXX() methods do not update the underlying database, instead the updateRow() or insertRow() methods are called to update the database.

13.11.15.1.43 Method updateFloat(String, float)

```
Synopsis: public void updateFloat(java.lang.String columnName, float x)
                throws java.sql.SQLException;
```

Parameters

columnName - the name of the column

x - the new column value

Exceptions

SQLException if a database-access error occurs

Update a column with a float value. The updateXXX() methods are used to update column values in the current row, or the insert row. The updateXXX() methods do not update the underlying database, instead the updateRow() or insertRow() methods are called to update the database.

13.11.15.1.44 Method updateInt(int, int)

```
Synopsis: public void updateInt(int columnIndex, int x)
                throws java.sql.SQLException;
```

Parameters

columnIndex - the first column is 1, the second is 2, ...

x - the new column value

Exceptions

SQLException if a database-access error occurs

Update a column with an integer value. The updateXXX() methods are used to update column values in the current row, or the insert row. The updateXXX() methods do not update the underlying database, instead the updateRow() or insertRow() methods are called to update the database.

13.11.15.1.45 Method updateInt(String, int)

```
Synopsis: public void updateInt(java.lang.String columnName, int x)
                throws java.sql.SQLException;
```

Parameters

columnName - the name of the column

x - the new column value

Exceptions

SQLException if a database-access error occurs

Update a column with an integer value. The updateXXX() methods are used to update column values in the current row, or the insert row. The updateXXX() methods do not update the underlying database, instead the updateRow() or insertRow() methods are called to update the database.

13.11.15.1.46 Method updateLong(int, long)

```
Synopsis: public void updateLong(int columnIndex, long x)
              throws java.sql.SQLException;
```

Parameters

columnIndex - the first column is 1, the second is 2, ...

x - the new column value

Exceptions

SQLException if a database-access error occurs

Update a column with a long value. The updateXXX() methods are used to update column values in the current row, or the insert row. The updateXXX() methods do not update the underlying database, instead the updateRow() or insertRow() methods are called to update the database.

13.11.15.1.47 Method updateLong(String, long)

```
Synopsis: public void updateLong(java.lang.String columnName, long x)
              throws java.sql.SQLException;
```

Parameters

columnName - the name of the column

x - the new column value

Exceptions

SQLException if a database-access error occurs

Update a column with a long value. The updateXXX() methods are used to update column values in the current row, or the insert row. The updateXXX() methods do not update the underlying database, instead the updateRow() or insertRow() methods are called to update the database.

13.11.15.1.48 Method updateNull(int)

```
Synopsis: public void updateNull(int columnIndex) throws java.sql.SQLException;
```

Parameters

columnIndex - the first column is 1, the second is 2, ...

Exceptions

SQLException if a database-access error occurs

Give a nullable column a null value. The updateXXX() methods are used to update column values in the current row, or the insert row. The updateXXX() methods do not update the underlying database, instead the updateRow() or insertRow() methods are called to update the database.

13.11.15.1.49 Method updateNull(String)

```
Synopsis: public void updateNull(java.lang.String columnName)
              throws java.sql.SQLException;
```

Parameters

columnName - the name of the column

Exceptions

SQLException if a database-access error occurs

Update a column with a null value. The updateXXX() methods are used to update column values in the current row, or the insert row. The updateXXX() methods do not update the underlying database, instead the updateRow() or insertRow() methods are called to update the database.

13.11.15.1.50 Method updateObject(int, Object)

```
Synopsis: public void updateObject(int columnIndex, java.lang.Object x)
                throws java.sql.SQLException;
```

Parameters

`columnIndex` - the first column is 1, the second is 2, ...

`x` - the new column value

Exceptions

SQLException if a database-access error occurs

Update a column with an Object value. The updateXXX() methods are used to update column values in the current row, or the insert row. The updateXXX() methods do not update the underlying database, instead the updateRow() or insertRow() methods are called to update the database.

13.11.15.1.51 Method updateObject(int, Object, int)

```
Synopsis: public void updateObject(int columnIndex, java.lang.Object x,
                                    int scale) throws java.sql.SQLException;
```

Parameters

`columnIndex` - the first column is 1, the second is 2, ...

`x` - the new column value

`scale` - For java.sql.Types.DECIMAL or java.sql.Types.NUMERIC types this is the number of digits after the decimal. For all other types this value will be ignored.

Exceptions

SQLException if a database-access error occurs

Update a column with an Object value. The updateXXX() methods are used to update column values in the current row, or the insert row. The updateXXX() methods do not update the underlying database, instead the updateRow() or insertRow() methods are called to update the database.

13.11.15.1.52 Method updateObject(String, Object)

```
Synopsis: public void updateObject(java.lang.String columnName,
                                    java.lang.Object x)
                throws java.sql.SQLException;
```

Parameters

`columnName` - the name of the column

`x` - the new column value

Exceptions

SQLException if a database-access error occurs

Update a column with an Object value. The updateXXX() methods are used to update column values in the current row, or the insert row. The updateXXX() methods do not update the underlying database, instead the updateRow() or insertRow() methods are called to update the database.

13.11.15.1.53 Method updateObject(String, Object, int)

```
Synopsis: public void updateObject(java.lang.String columnName,
                                    java.lang.Object x, int scale)
                throws java.sql.SQLException;
```

Parameters

`columnName` - the name of the column

`x` - the new column value

Parameters

`scale` - For java.sql.Types.DECIMAL or java.sql.Types.NUMERIC types this is the number of digits after the decimal. For all other types this value will be ignored.

Exceptions

SQLException if a database-access error occurs

Update a column with an Object value. The updateXXX() methods are used to update column values in the current row, or the insert row. The updateXXX() methods do not update the underlying database, instead the updateRow() or insertRow() methods are called to update the database.

13.11.15.1.54 Method updateRef(int, Ref)

```
Synopsis: public void updateRef(int columnIndex, java.sql.Ref x)
              throws java.sql.SQLException;
```

Parameters

`columnIndex` - the first column is 1, the second is 2, ...

`x` - the new column value

Exceptions

SQLException if a database access error occurs

Additional Information

Since 1.4

Updates the designated column with a `java.sql.Ref` value. The updater methods are used to update column values in the current row or the insert row. The updater methods do not update the underlying database; instead the `updateRow` or `insertRow` methods are called to update the database.

13.11.15.1.55 Method updateRef(String, Ref)

```
Synopsis: public void updateRef(java.lang.String columnName, java.sql.Ref x)
              throws java.sql.SQLException;
```

Parameters

`columnName` - the name of the column

`x` - the new column value

Exceptions

SQLException if a database access error occurs

Additional Information

Since 1.4

Updates the designated column with a `java.sql.Ref` value. The updater methods are used to update column values in the current row or the insert row. The updater methods do not update the underlying database; instead the `updateRow` or `insertRow` methods are called to update the database.

13.11.15.1.56 Method updateRow()

```
Synopsis: public void updateRow() throws java.sql.SQLException;
```

Exceptions

SQLException if a database-access error occurs, or if called when on the insert row

Update the underlying database with the new contents of the current row. Cannot be called when on the insert row.

13.11.15.1.57 Method updateShort(int, short)

```
Synopsis: public void updateShort(int columnIndex, short x)
              throws java.sql.SQLException;
```

Parameters

columnIndex - the first column is 1, the second is 2, ...

x - the new column value

Exceptions

SQLException if a database-access error occurs

Update a column with a short value. The updateXXX() methods are used to update column values in the current row, or the insert row. The updateXXX() methods do not update the underlying database, instead the updateRow() or insertRow() methods are called to update the database.

13.11.15.1.58 Method updateShort(String, short)

```
Synopsis: public void updateShort(java.lang.String columnName, short x)
              throws java.sql.SQLException;
```

Parameters

columnName - the name of the column

x - the new column value

Exceptions

SQLException if a database-access error occurs

Update a column with a short value. The updateXXX() methods are used to update column values in the current row, or the insert row. The updateXXX() methods do not update the underlying database, instead the updateRow() or insertRow() methods are called to update the database.

13.11.15.1.59 Method updateString(int, String)

```
Synopsis: public void updateString(int columnIndex, java.lang.String x)
              throws java.sql.SQLException;
```

Parameters

columnIndex - the first column is 1, the second is 2, ...

x - the new column value

Exceptions

SQLException if a database-access error occurs

Update a column with a String value. The updateXXX() methods are used to update column values in the current row, or the insert row. The updateXXX() methods do not update the underlying database, instead the updateRow() or insertRow() methods are called to update the database.

13.11.15.1.60 Method updateString(String, String)

```
Synopsis: public void updateString(java.lang.String columnName,
                                   java.lang.String x)
              throws java.sql.SQLException;
```

Parameters

columnName - the name of the column

x - the new column value

Exceptions

SQLException if a database-access error occurs

Update a column with a String value. The updateXXX() methods are used to update column values in the current row, or the insert row. The updateXXX() methods do not update the underlying database, instead the updateRow() or insertRow() methods are called to update the database.

13.11.15.1.61 Method updateTime(int, Time)

```
Synopsis: public void updateTime(int columnIndex, java.sql.Time x)
              throws java.sql.SQLException;
```

Parameters

columnIndex - the first column is 1, the second is 2, ...

x - the new column value

Exceptions

SQLException if a database-access error occurs

Update a column with a Time value. The updateXXX() methods are used to update column values in the current row, or the insert row. The updateXXX() methods do not update the underlying database, instead the updateRow() or insertRow() methods are called to update the database.

13.11.15.1.62 Method updateTime(String, Time)

```
Synopsis: public void updateTime(java.lang.String columnName, java.sql.Time x)
              throws java.sql.SQLException;
```

Parameters

columnName - the name of the column

x - the new column value

Exceptions

SQLException if a database-access error occurs

Update a column with a Time value. The updateXXX() methods are used to update column values in the current row, or the insert row. The updateXXX() methods do not update the underlying database, instead the updateRow() or insertRow() methods are called to update the database.

13.11.15.1.63 Method updateTimestamp(int, Timestamp)

```
Synopsis: public void updateTimestamp(int columnIndex, java.sql.Timestamp x)
              throws java.sql.SQLException;
```

Parameters

columnIndex - the first column is 1, the second is 2, ...

x - the new column value

Exceptions

SQLException if a database-access error occurs

Update a column with a Timestamp value. The updateXXX() methods are used to update column values in the current row, or the insert row. The updateXXX() methods do not update the underlying database, instead the updateRow() or insertRow() methods are called to update the database.

13.11.15.1.64 Method updateTimestamp(String, Timestamp)

```
Synopsis: public void updateTimestamp(java.lang.String columnName,
                                      java.sql.Timestamp x)
              throws java.sql.SQLException;
```

Parameters

columnName - the name of the column

x - the new column value

Exceptions

SQLException if a database-access error occurs

Update a column with a Timestamp value. The updateXXX() methods are used to update column values in the current row, or the insert row. The updateXXX() methods do not update the underlying database, instead the updateRow() or insertRow() methods are called to update the database.

13.11.15.1.65 Method wasNull()

```
Synopsis: public boolean wasNull() throws java.sql.SQLException;
```

Exceptions

SQLException	if a database-access error occurs.

A column may have the value of SQL NULL; wasNull reports whether the last column read had this special value. Note that you must first call getXXX on a column to try to read its value and then call wasNull() to find if the value was the SQL NULL.

13.11.16 Class OPLPooledConnection

A PooledConnection object is a connection object that provides hooks for connection pool management. A PooledConnection object represents a physical connection to a data source.

13.11.16.1 Synopsis

```
public class OPLPooledConnection implements PooledConnection,Cloneable {
  // Public Methods
  public void finalize() throws java.lang.Throwable;
  public void addConnectionEventListener(
                javax.sql.ConnectionEventListener parm);
  public void removeConnectionEventListener(
                javax.sql.ConnectionEventListener parm);
  public void close() throws java.sql.SQLException;
  public void closeAll() throws java.sql.SQLException;
  public Connection getConnection() throws java.sql.SQLException;
}
```

Inheritance Path

java.lang.Object

|

openlink.javax.OPLPooledConnection

13.11.16.2 Members

13.11.16.2.1 Method addConnectionEventListener(ConnectionEventListener)

```
Synopsis: public void addConnectionEventListener(
                        javax.sql.ConnectionEventListener parm);
```

Add an event listener.

13.11.16.2.2 Method close()

```
Synopsis: public void close() throws java.sql.SQLException;
```

Exceptions

java.sql.SQLException	if a database-access error occurs

Close the physical connection.

13.11.16.2.3 Method closeAll()

```
Synopsis: public void closeAll() throws java.sql.SQLException;
```

Exceptions

java.sql.SQLException	if a database-access error occurs

Close all the Statement objects that have been opened by this PooledConnection object.

13.11.16.2.4 Method getConnection()

```
Synopsis: public Connection getConnection() throws java.sql.SQLException;
```

Exceptions

java.sql.SQLException	if a database-access error occurs

Create an object handle for this physical connection. The object returned is a temporary handle used by application code to refer to a physical connection that is being pooled.

13.11.16.2.5 Method removeConnectionEventListener(ConnectionEventListener)

```
Synopsis: public void removeConnectionEventListener(
                          javax.sql.ConnectionEventListener parm);
```

Remove an event listener.

13.11.17 Class OPLPoolStatistic

13.11.17.1 Synopsis

```
public class OPLPoolStatistic implements Cloneable {
  // Public Methods
  public int getHits();
  public int getMisses();
  public long getMaxWaitTime();
  public long getMinWaitTime();
  public long getCumWaitTime();
  public int getCacheSize();
  public int getConnsInUse();
  public int getConnsUnUsed();
  public String getName();
  public String toString();
}
```

Inheritance Path

java.lang.Object

|

openlink.javax.OPLPoolStatistic

13.11.17.2 Members

13.11.17.2.1 Method getCacheSize()

```
Synopsis: public int getCacheSize();
```

Returns the connection pool size.

13.11.17.2.2 Method getConnsInUse()

```
Synopsis: public int getConnsInUse();
```

Returns the amount of used connections in the connection pool size.

13.11.17.2.3 Method getConnsUnUsed()

```
Synopsis: public int getConnsUnUsed();
```

Returns the amount of unused connections in the connection pool size.

13.11.17.2.4 Method getCumWaitTime()

```
Synopsis: public long getCumWaitTime();
```

Returns the sum of all waiting time for the connection pool.

13.11.17.2.5 Method getHits()

```
Synopsis: public int getHits();
```

Returns the amount of the connection pool hits.

13.11.17.2.6 Method getMaxWaitTime()

```
Synopsis: public long getMaxWaitTime();
```

Returns the maximal waiting time for the connection pool.

13.11.17.2.7 Method getMinWaitTime()

```
Synopsis: public long getMinWaitTime();
```

Returns the minimal waiting time for the connection pool.

13.11.17.2.8 Method getMisses()

```
Synopsis: public int getMisses();
```

Returns the amount of the connection pool misses.

13.11.17.2.9 Method getName()

```
Synopsis: public String getName();
```

Returns the name of ConectionPoolDataSource.

13.11.18 Class OPLRowSetMetaData

The RowSetMetaData interface extends ResultSetMetaData with methods that allow a metadata object to be initialized.

13.11.18.1 Synopsis

```
public class OPLRowSetMetaData implements RowSetMetaData,Serializable {
  // Public Constructors
  public OPLRowSetMetaData(java.sql.ResultSetMetaData rsmd)
      throws java.sql.SQLException;
  // Public Methods
  public void setAutoIncrement(int column, boolean property)
      throws java.sql.SQLException;
  public void setCaseSensitive(int column, boolean property)
      throws java.sql.SQLException;
  public void setCatalogName(int column, java.lang.String catalogName)
      throws java.sql.SQLException;
  public void setColumnCount(int columnCount) throws java.sql.SQLException;
  public void setColumnDisplaySize(int column, int size)
      throws java.sql.SQLException;
  public void setColumnLabel(int column, java.lang.String label)
      throws java.sql.SQLException;
  public void setColumnName(int column, java.lang.String columnName)
```

```
        throws java.sql.SQLException;
    public void setColumnType(int column, int SQLType)
        throws java.sql.SQLException;
    public void setColumnTypeName(int column, java.lang.String typeName)
        throws java.sql.SQLException;
    public void setCurrency(int column, boolean property)
        throws java.sql.SQLException;
    public void setNullable(int column, int property)
        throws java.sql.SQLException;
    public void setPrecision(int column, int precision)
        throws java.sql.SQLException;
    public void setScale(int column, int scale) throws java.sql.SQLException;
    public void setSchemaName(int column, java.lang.String schemaName)
        throws java.sql.SQLException;
    public void setSearchable(int column, boolean property)
        throws java.sql.SQLException;
    public void setSigned(int column, boolean property)
        throws java.sql.SQLException;
    public void setTableName(int column, java.lang.String tableName)
        throws java.sql.SQLException;
    public int getColumnCount() throws java.sql.SQLException;
    public boolean isAutoIncrement(int column) throws java.sql.SQLException;
    public boolean isCaseSensitive(int column) throws java.sql.SQLException;
    public boolean isSearchable(int column) throws java.sql.SQLException;
    public boolean isCurrency(int column) throws java.sql.SQLException;
    public int isNullable(int column) throws java.sql.SQLException;
    public boolean isSigned(int column) throws java.sql.SQLException;
    public int getColumnDisplaySize(int column) throws java.sql.SQLException;
    public String getColumnLabel(int column) throws java.sql.SQLException;
    public String getColumnName(int column) throws java.sql.SQLException;
    public String getSchemaName(int column) throws java.sql.SQLException;
    public int getPrecision(int column) throws java.sql.SQLException;
    public int getScale(int column) throws java.sql.SQLException;
    public String getTableName(int column) throws java.sql.SQLException;
    public String getCatalogName(int column) throws java.sql.SQLException;
    public int getColumnType(int column) throws java.sql.SQLException;
    public String getColumnTypeName(int column) throws java.sql.SQLException;
    public boolean isReadOnly(int column) throws java.sql.SQLException;
    public boolean isWritable(int column) throws java.sql.SQLException;
    public boolean isDefinitelyWritable(int column) throws java.sql.SQLException;
    public String getColumnClassName(int column) throws java.sql.SQLException;
  }
```

Inheritance Path

java.lang.Object

|

openlink.javax.OPLRowSetMetaData

13.11.18.2 Members

13.11.18.2.1 Method getCatalogName(int)

```
Synopsis: public String getCatalogName(int column)
              throws java.sql.SQLException;
```

Parameters

`column` - the first column is 1, the second is 2, ...

`return` - column name or "" if not applicable

Exceptions

SQLException if a database access error occurs

Gets the designated column's table's catalog name.

13.11.18.2.2 Method getColumnClassName(int)

```
Synopsis: public String getColumnClassName(int column)
              throws java.sql.SQLException;
```

Exceptions

SQLException	if a database access error occurs

Additional Information

Since	1.2
See Also	What Is in the JDBC 2.0 API

Returns the fully-qualified name of the Java class whose instances are manufactured if the method `ResultSet.getObject` is called to retrieve a value from the column. `ResultSet.getObject` may return a subclass of the class returned by this method.

13.11.18.2.3 Method getColumnCount()

```
Synopsis: public int getColumnCount() throws java.sql.SQLException;
```

Exceptions

SQLException	if a database access error occurs

Returns the number of columns in this RowSet

13.11.18.2.4 Method getColumnDisplaySize(int)

```
Synopsis: public int getColumnDisplaySize(int column)
              throws java.sql.SQLException;
```

Parameters

`column` - the first column is 1, the second is 2, ...

`return` - the normal maximum number of characters allowed as the width of the designated column

Exceptions

SQLException	if a database access error occurs

Indicates the designated column's normal maximum width in characters.

13.11.18.2.5 Method getColumnLabel(int)

```
Synopsis: public String getColumnLabel(int column)
              throws java.sql.SQLException;
```

Parameters

`column` - the first column is 1, the second is 2, ...

`return` - the suggested column title

Exceptions

SQLException	if a database access error occurs

Gets the designated column's suggested title for use in printouts and displays.

13.11.18.2.6 Method getColumnName(int)

```
Synopsis: public String getColumnName(int column) throws java.sql.SQLException;
```

Parameters

`column` - the first column is 1, the second is 2, ...

`return` - column name

Exceptions

SQLException	if a database access error occurs

Get the designated column's name.

13.11.18.2.7 Method getColumnType(int)

```
Synopsis: public int getColumnType(int column) throws java.sql.SQLException;
```

Parameters

`column` - the first column is 1, the second is 2, ...

`return` - SQL type from java.sql.Types

Exceptions

SQLException if a database access error occurs

Additional Information

See Also java.sql.Types

Retrieves the designated column's SQL type.

13.11.18.2.8 Method getColumnTypeName(int)

```
Synopsis: public String getColumnTypeName(int column)
              throws java.sql.SQLException;
```

Parameters

`column` - the first column is 1, the second is 2, ...

`return` - type name used by the database. If the column type is a user-defined type, then a fully-qualified type name is returned.

Exceptions

SQLException if a database access error occurs

Retrieves the designated column's database-specific type name.

13.11.18.2.9 Method getPrecision(int)

```
Synopsis: public int getPrecision(int column) throws java.sql.SQLException;
```

Parameters

`column` - the first column is 1, the second is 2, ...

`return` - precision

Exceptions

SQLException if a database access error occurs

Get the designated column's number of decimal digits.

13.11.18.2.10 Method getScale(int)

```
Synopsis: public int getScale(int column) throws java.sql.SQLException;
```

Parameters

`column` - the first column is 1, the second is 2, ...

`return` - scale

Exceptions

SQLException if a database access error occurs

Gets the designated column's number of digits to right of the decimal point.

13.11.18.2.11 Method getSchemaName(int)

```
Synopsis: public String getSchemaName(int column) throws java.sql.SQLException;
```

Parameters

Parameters

`column` - the first column is 1, the second is 2, ...

`return` - schema name or "" if not applicable

Exceptions

SQLException if a database access error occurs

Get the designated column's table's schema.

13.11.18.2.12 Method getTableName(int)

```
Synopsis: public String getTableName(int column) throws java.sql.SQLException;
```

Parameters

`column` - the first column is 1, the second is 2, ...

`return` - table name or "" if not applicable

Exceptions

SQLException if a database access error occurs

Gets the designated column's table name.

13.11.18.2.13 Method isAutoIncrement(int)

```
Synopsis: public boolean isAutoIncrement(int column)
                throws java.sql.SQLException;
```

Parameters

`column` - the first column is 1, the second is 2, ...

`return` - true if so

Exceptions

SQLException if a database access error occurs

Indicates whether the column is automatically numbered, thus read-only.

13.11.18.2.14 Method isCaseSensitive(int)

```
Synopsis: public boolean isCaseSensitive(int column)
                throws java.sql.SQLException;
```

Parameters

`column` - the first column is 1, the second is 2, ...

`return` - `true` if so; `false` otherwise

Exceptions

SQLException if a database access error occurs

Indicates whether a column's case matters.

13.11.18.2.15 Method isCurrency(int)

```
Synopsis: public boolean isCurrency(int column) throws java.sql.SQLException;
```

Parameters

`column` - the first column is 1, the second is 2, ...

`return` - `true` if so; `false` otherwise

Exceptions

SQLException if a database access error occurs

Indicates whether the designated column is a cash value.

13.11.18.2.16 Method isDefinitelyWritable(int)

```
Synopsis: public boolean isDefinitelyWritable(int column)
                throws java.sql.SQLException;
```

Parameters

`column` - the first column is 1, the second is 2, ...

`return` - `true` if so; `false` otherwise

Exceptions

SQLException if a database access error occurs

Indicates whether a write on the designated column will definitely succeed.

13.11.18.2.17 Method isNullable(int)

```
Synopsis: public int isNullable(int column) throws java.sql.SQLException;
```

Parameters

`column` - the first column is 1, the second is 2, ...

`return` - the nullability status of the given column; one of `columnNoNulls`, `columnNullable` or `columnNullableUnknown`

Exceptions

SQLException if a database access error occurs

Indicates the nullability of values in the designated column.

13.11.18.2.18 Method isReadOnly(int)

```
Synopsis: public boolean isReadOnly(int column) throws java.sql.SQLException;
```

Parameters

`column` - the first column is 1, the second is 2, ...

`return` - `true` if so; `false` otherwise

Exceptions

SQLException if a database access error occurs

Indicates whether the designated column is definitely not writable.

13.11.18.2.19 Method isSearchable(int)

```
Synopsis: public boolean isSearchable(int column) throws java.sql.SQLException;
```

Parameters

`column` - the first column is 1, the second is 2, ...

`return` - `true` if so; `false` otherwise

Exceptions

SQLException if a database access error occurs

Indicates whether the designated column can be used in a where clause.

13.11.18.2.20 Method isSigned(int)

```
Synopsis: public boolean isSigned(int column) throws java.sql.SQLException;
```

Parameters

`column` - the first column is 1, the second is 2, ...

`return` - `true` if so; `false` otherwise

Exceptions

SQLException if a database access error occurs

Indicates whether values in the designated column are signed numbers.

13.11.18.2.21 Method isWritable(int)

```
Synopsis: public boolean isWritable(int column) throws java.sql.SQLException;
```

Parameters

`column` - the first column is 1, the second is 2, ...

`return` - `true` if so; `false` otherwise

Exceptions

SQLException if a database access error occurs

Indicates whether it is possible for a write on the designated column to succeed.

13.11.18.2.22 Method setAutoIncrement(int, boolean)

```
Synopsis: public void setAutoIncrement(int column, boolean property)
              throws java.sql.SQLException;
```

Parameters

`column` - the first column is 1, the second is 2, ...

`property` - is either true or false (default is false).

Exceptions

SQLException if a database-access error occurs.

Specify whether the is column automatically numbered, thus read-only.

13.11.18.2.23 Method setCaseSensitive(int, boolean)

```
Synopsis: public void setCaseSensitive(int column, boolean property)
              throws java.sql.SQLException;
```

Parameters

`column` - the first column is 1, the second is 2, ...

`property` - is either true or false (default is false).

Exceptions

SQLException if a database-access error occurs.

Specify whether the column is case sensitive.

13.11.18.2.24 Method setCatalogName(int, String)

```
Synopsis: public void setCatalogName(int column, java.lang.String catalogName)
              throws java.sql.SQLException;
```

Parameters

`column` - the first column is 1, the second is 2, ...

`catalogName` - column's catalog name.

Exceptions

SQLException if a database-access error occurs.

Specify the column's table's catalog name, if any.

13.11.18.2.25 Method setColumnCount(int)

```
Synopsis: public void setColumnCount(int columnCount)
              throws java.sql.SQLException;
```

Parameters

`columnCount` - number of columns.

Exceptions

SQLException if a database-access error occurs.

Set the number of columns in the RowSet.

13.11.18.2.26 Method setColumnDisplaySize(int, int)

```
Synopsis: public void setColumnDisplaySize(int column, int size)
              throws java.sql.SQLException;
```

Parameters

`column` - the first column is 1, the second is 2, ...

`size` - size of the column

Exceptions

SQLException if a database-access error occurs.

Specify the column's normal max width in chars.

13.11.18.2.27 Method setColumnLabel(int, String)

```
Synopsis: public void setColumnLabel(int column, java.lang.String label)
              throws java.sql.SQLException;
```

Parameters

`column` - the first column is 1, the second is 2, ...

`label` - the column title

Exceptions

SQLException if a database-access error occurs.

Specify the suggested column title for use in printouts and displays, if any.

13.11.18.2.28 Method setColumnName(int, String)

```
Synopsis: public void setColumnName(int column, java.lang.String columnName)
              throws java.sql.SQLException;
```

Parameters

`column` - the first column is 1, the second is 2, ...

`columnName` - the column name

Exceptions

SQLException if a database-access error occurs.

Specify the column name.

13.11.18.2.29 Method setColumnType(int, int)

```
Synopsis: public void setColumnType(int column, int SQLType)
              throws java.sql.SQLException;
```

Parameters

`column` - the first column is 1, the second is 2, ...

Parameters

`SQLType` - column's SQL type.

Exceptions

SQLException if a database-access error occurs.

Additional Information

See Also java.sql.Types

Specify the column's SQL type.

13.11.18.2.30 Method setColumnTypeName(int, String)

```
Synopsis: public void setColumnTypeName(int column, java.lang.String typeName)
        throws java.sql.SQLException;
```

Parameters

`column` - the first column is 1, the second is 2, ...

`typeName` - data source specific type name.

Exceptions

SQLException if a database-access error occurs.

Specify the column's data source specific type name, if any.

13.11.18.2.31 Method setCurrency(int, boolean)

```
Synopsis: public void setCurrency(int column, boolean property)
        throws java.sql.SQLException;
```

Parameters

`column` - the first column is 1, the second is 2, ...

`property` - is either true or false (default is false).

Exceptions

SQLException if a database-access error occurs.

Specify whether the column is a cash value.

13.11.18.2.32 Method setNullable(int, int)

```
Synopsis: public void setNullable(int column, int property)
        throws java.sql.SQLException;
```

Parameters

`column` - the first column is 1, the second is 2, ...

`property` - is either one of columnNoNulls, columnNullable or columnNullableUnknown (default is columnNullableUnknown).

Exceptions

SQLException if a database-access error occurs.

Specify whether the column's value can be set to NULL.

13.11.18.2.33 Method setPrecision(int, int)

```
Synopsis: public void setPrecision(int column, int precision)
        throws java.sql.SQLException;
```

Parameters

`column` - the first column is 1, the second is 2, ...

`precision` - number of decimal digits.

Exceptions

Exceptions

SQLException if a database-access error occurs.

Specify the column's number of decimal digits.

13.11.18.2.34 Method setScale(int, int)

```
Synopsis: public void setScale(int column, int scale)
              throws java.sql.SQLException;
```

Parameters

column - the first column is 1, the second is 2, ...

scale - number of digits to right of decimal point.

Exceptions

SQLException if a database-access error occurs.

Specify the column's number of digits to right of the decimal point.

13.11.18.2.35 Method setSchemaName(int, String)

```
Synopsis: public void setSchemaName(int column, java.lang.String schemaName)
              throws java.sql.SQLException;
```

Parameters

column - the first column is 1, the second is 2, ...

schemaName - the schema name

Exceptions

SQLException if a database-access error occurs.

Specify the column's table's schema, if any.

13.11.18.2.36 Method setSearchable(int, boolean)

```
Synopsis: public void setSearchable(int column, boolean property)
              throws java.sql.SQLException;
```

Parameters

column - the first column is 1, the second is 2, ...

property - is either true or false (default is false).

Exceptions

SQLException if a database-access error occurs.

Specify whether the column can be used in a where clause.

13.11.18.2.37 Method setSigned(int, boolean)

```
Synopsis: public void setSigned(int column, boolean property)
              throws java.sql.SQLException;
```

Parameters

column - the first column is 1, the second is 2, ...

property - is either true or false (default is false).

Exceptions

SQLException if a database-access error occurs.

Speicfy whether the column is a signed number.

13.11.18.2.38 Method setTableName(int, String)

```
Synopsis: public void setTableName(int column, java.lang.String tableName)
                 throws java.sql.SQLException;
```

Parameters

`column` - the first column is 1, the second is 2, ...

`tableName` - column's table name.

Exceptions

SQLException if a database-access error occurs.

Specify the column's table name, if any.

13.12 New Features

- Unicode Driver support. Database agents support the latest releases of all supported database engines this includes:

  1. Oracle 8i & 9i
  2. Microsoft SQL Server 2000
  3. DB/2 v7
  4. ODBC Agent
  5. JDBC Agent
  6. Informix 9
  7. Sybase Adaptive Server 12.5
  8. Progress 9 (SQL-92)
- Supports JDBC 3.0 (JDK 1.4). Support of JDBC 3.0 (JDK 1.4) is now available, with new JDBC Client drivers opljdbc3.jar & megathin3.jar for providing this connectivty. A new Sample program is also available for demonstrating the used of Cached RowSet in JDBC 3.0
- Significant SQL Server and SYBASE Driver Enhancements. Our Drivers are now built to communicate directly with Microsoft SQL Server and SYBASE ASE using the TDS protocol (the native wire protocol for both database servers). This also implies that no additional software is required post installation in order for our ODBC Drivers to communicate with these Drivers (this applies to the Single Tier format Drivers only). The use of TDS has also enabled us to double the performance of both our Single Tier and Multi-Tier Drivers for these databases.
- Array Optimisations on Select Queries. All drivers now support the SQLSetStmtAttr(SQL_ARRAY_SIZE) call for batch select statements, providing improved performance when re-execute select statements with bound paramters.
- Deferred Fetching. The Release 4 OpenLink driver family brings marked communications layer improvements. Central to these are improved implementation of the `SQLGetData` and `SQLPutData` ODBC function calls.

  Wholesale vs. Piecemeal Data Transfer. In previous releases of the drivers, parameter data at query execution was assembled in the OpenLink driver's client component and transferred to its server component in a single network hop. (The client and server components apply to both the Single- and Multi-Tier drivers; they refer to different layers within the driver entity.) Similarly, when fetching from a "long", or large binary data column, data was transferred from the driver's server component to the client component in a single network transfer. The only way data could manipulated in a piecemeal fashion, was within the ODBC application from the driver's client component (client side only).

  The Release 4 driver family now allow transferring parameter data in parts over the network between the client and server driver components. Once transferred, the fragmented column data are re-pieced together in their entirety within the client and server portions of the driver.

  Deferred Fetching. When fetching, data from columns with "long" data are only transferred between the server and client components if one of the following applies to that column:

  It has been "bound" by the application via the SQLBindCol API call

  It has been retrieved via the SQLGetData API call

  This mechanism is referred to as deferred fetching. In this method, as "long" column data is not reassembled within the OpenLink ODBC client itself (rather, within the application), driver memory overhead incurred is dramatically reduced. Deferred fetching applies to the following "long" database column types:

ODBC agent

- SQL_LONGVARCHAR
- SQL_LONGVARBINARY

DB/2 agent

- SQL_BLOB
- SQL_CLOB
- SQL_DBCLOB
- If the long data compatibility option has been specified in the DB2 database
- SQL_LONGVARCHAR
- SQL_LONGVARBINARY
- SQL_LONGVARGRAPHIC

Oracle agent

- SQLT_BLOB
- SQLT_CLOB

Sybase agent

- CS_IMAGE_TYPE
- CS_TEXT_TYPE

However, if a table contains a column defined as one of these "long" types, but the actual data stored in the column only fills a small proportion of the available space, deferred fetching is of no benefit. In these cases, performance may be improved by switching off the deferred fetching mechanism using the control in the OpenLink Generic Client data source setup dialog.

14 Chapter 12. Distributed Transaction Processing (XA)

Abstract
The X/Open Distributed Transaction Processing (DTP) model defines a specification for the management of transactions whose operations are distributed among different computers or among different databases from different vendors.

Table of Contents

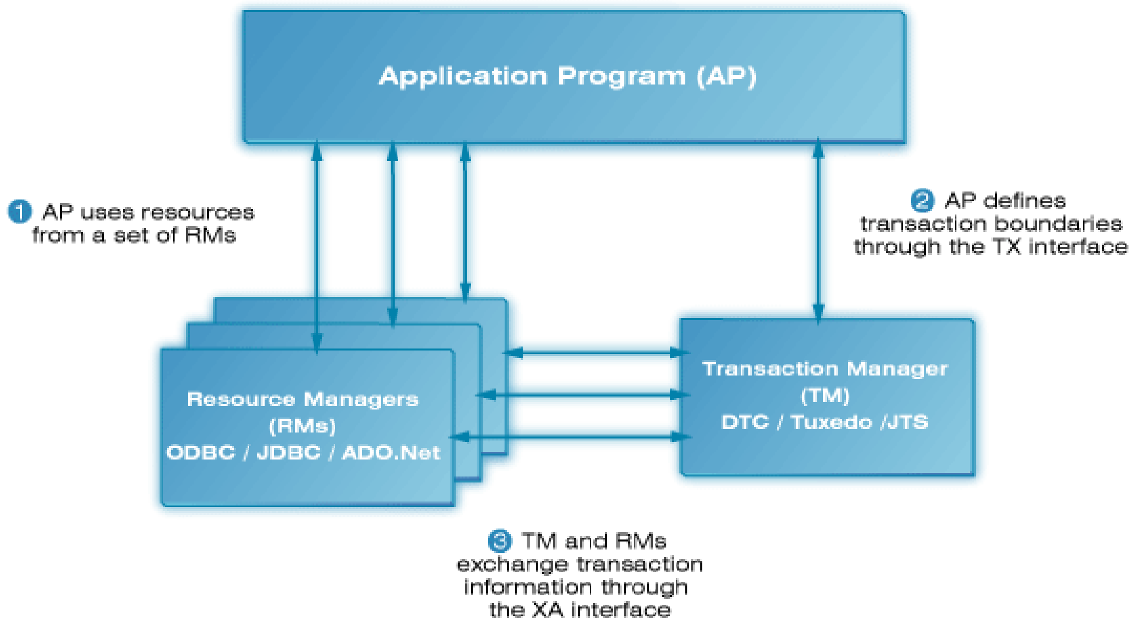
14.1 Distributed Transaction Processing (DTP)

14.1.1 X/Open DTP Model

DTP can be broken down into 3 core components:

- An Application Program (AP) for defining transaction boundires and specifies actions that constitute a transaction.
- Resource Managers (RMs, such as databases or Data Access Drivers ie ODBC, JDBC, ADO.Net for accessing them)provide access to shared resources.
- A Transaction Manager (TM) that assigns identifiers to a transactions, monitors the progress, and takes responsibility for transaction completion and fault recovery.

Figure 12.1. DTP-XA.gif


Application Program (AP)
1 AP uses resources from a set of RMs
2 AP defines transaction boundaries through the TX interface
Resource Managers (RMs) ODBC / JDBC / ADO.Net
Transaction Manager (TM) DTC / Tuxedo /JTS
3 TM and RMs exchange transaction information through the XA interface

The TX standard defines the API used for communication between an application program and a transaction manager. The XA (eXtended Architecture) standard defines the two-phase commit protocol and the API used for communication between a transaction manager and a resource manager. A resource manager is a system service that manages durable data, e.g. a DBMS or file system. A transaction manager manages distributed transactions, which may span multiple resource managers. It manages two-phase commit, coordinating the decision to commit distributed transactions or roll them back, and coordinates failure recovery.

Whenever an application has a single function that needs to access or update the data in multiple transactional resources, it should use a distributed transaction. You can use a separate simple transaction on each of the resources, but this approach is error-prone. If the transaction in one resource commits successfully but another fails and must roll back, the first transaction can no longer be rolled back, so the state of the application becomes inconsistent. If one resource commits successfully but the system crashes before the other resource can commit successfully, the application again is inconsistent. An application moving data from one database to another requires a distributed transaction. Otherwise, the data may be duplicated (if the insert completes and the delete fails) or lost (if the delete completes and the insert fails).

14.1.1.1 Distributed Transaction Coordinator (DTC)

The Microsoft COM and MTS interfaces were unified into a single run-time layer and renamed COM+ for Windows 2000. COM+ includes the transaction support that was created for MTS. Both include a system service, DTC, for coordinating distributed transactions and the two-phase commit protocol. Every machine running COM+ has a DTC system service.

For distributed transactions, each computer has a local transaction manager ie. the DTC on that machine. When a transaction does work at multiple computers, the transaction managers interact with other transaction managers via either a superior or subordinate relationship. These relationships are relevant only for a particular transaction. The root transaction manager, also referred to as the global commit coordinator, is the transaction manager on the system that initiates a transaction.

DTC introduces the notion of a resource dispenser which does two things. It manages a pool of connections to a resource manager (i.e. a DBMS in the context of Data Access) and it automatically enlists the resource manager in a component's transaction. The ODBC Driver Manager is a typical resource dispenser, as is an ADO .NET Data Provider.

14.1.1.2 Tuxedo

BEA Tuxedo provides the framework, or middleware, for building scalable multi-tier client/server applications in heterogeneous (dissimilar), distributed environments that extend from the Web to the Enterprise. Using BEA Tuxedo, users can develop, manage, and deploy distributed applications independently of the underlying hardware, operating system, network, and database environment.

In contrast to classic 2-tier client/server configuration of SQL servers, the Tuxedo brings 3-tier paradigm (clients, services, resource managers).

At the foundation of BEA Tuxedo ATMI is a proven, reliable transaction processor, also known as a transaction processing (TP) monitor. A transaction processor is an example of a 3-tier client/server architecture, where the transaction processor supports the application logic (represented by "services" between the GUI front-end and the back-end resource managers. Examples of resource managers are SQL databases, message queues, legacy applications, and other back-end services.

Tuxedo's Application-to-Transaction Monitor Interface consists of the Tuxedo transaction processor and infrastructure, and the ATMI API. The ATMI API is used to create a Tuxedo client and server.

A Tuxedo ATMI client collects a user request and forwards it to a server offering the requested service. The client "joins" a distributed Tuxedo application by calling the ATMI client initialization routine. The client can then exchange information with the server, sending and receiving data in typed buffers, and define transaction boundaries The client "leaves" the application by issuing an ATMI termination function.

A Tuxedo server process oversees a set of services, calling them automatically for clients that request them. ATMI clients do not call servers by name; they call services. The server is not written as a complete program (i.e. with a standard main). Instead, the server's services are coded as a set of functions that are compiled with Tuxedo binaries to produce a server executable.

14.1.1.3 Java Transaction API (JTA)

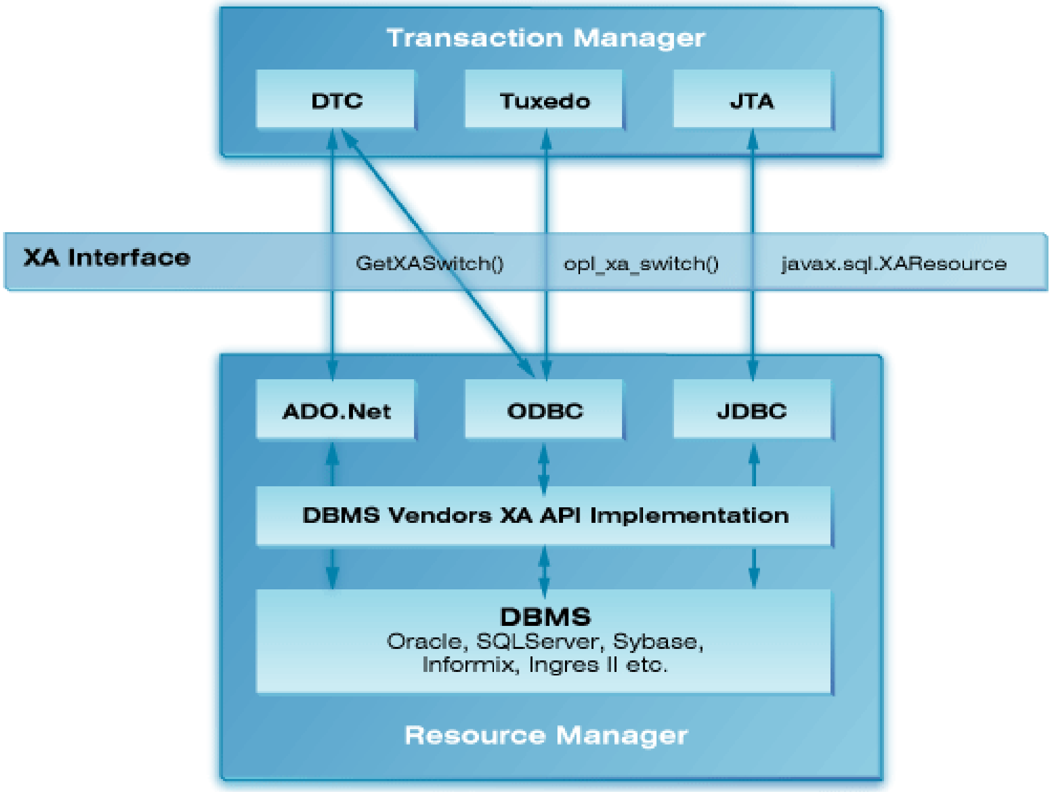
The JTA, or Java Transaction API, is a Java Enterprise API for managing distributed transactions. It defines a Java binding for the standard XA API for distributed transactions (XA is a standard defined by the Open Group). Using the JTA, one can write a program that communicates with a distributed transaction service and uses that service to coordinate a distributed transaction that accesses and update data on two or more networked computer resources, or databases in the case of JDBC.

The XA specification defines an interface between the transaction manager (TM) and resource manager (RM) in a distributed transaction system. This is a generic interface and it does not directly address the use of distributed transactions from Java. The Java mapping of the XA interface is defined in Sun Microsystems Java Transaction API (JTA) and JDBC 2.0/3.0 specifications.

14.1.2 OpenLink Resource Manager XA Interface Implementation

The OpenLink DataAccess drivers (ODBC, JDBC, ADO.Net) form part of the Resource Manager in the overal DTP model, implementing the XA (eXtended Architecture) interface for the comminication between itself as the upper layer of Resource Manager proess and the Transaction Manager (DTC, Tuxedo, JTS), and also providing the binding to the Database Vendors XA API implementation, which forms the lower layer of the Resoure Manager:

Figure 12.2. UDA-XA.gif



Distributed transactions are supported in the OpenLink Multi-Tier and Single-Tier ODBC, JDBC and ADO.Net drivers/providers for Oracle, SQLServer, Sybase, Informix and Ingres II.

14.1.2.1 ODBC

The OpenLink Multi-Tier and Single-Tier ODBC drivers support distributed transactions using the Microsoft Distributed Transaction Co-ordinator (on Windows only) or Tuxedo as the Transaction Manager of the distributed transaction process.

14.1.2.1.1 Distributed Transaction Co-ordinator (DTC)

An OpenLink XA compliant ODBC driver acts as a proxy for an XA capable database (RM) and as such can be thought of as a resource manager. The OpenLink XA compliant ODBC drivers implement and export the function GetXASwitch() which provide the entry points for the DBMS specific XA functions which the DTC can use to communicate directly with the DBMS should an for of distributed transaction recovery need to the performed.

14.1.2.1.1.1 Transaction Enlistment

After initiating a transaction, an application enlists the resource managers it wants to participate in the transaction. Enlistment is done through the ODBC 3.0 connection attribute function call:

SQLSetConnectAttr(SQL_ATTR_ENLIST_IN_DTC)

14.1.2.1.2 Tuxedo

An OpenLink XA compliant ODBC driver acts as a proxy for an XA capable database (RM) and as such can be thought of as a resource manager. The driver exports a function opl_xa_switch:

```
        xa_switch_t * opl_xa_switch (void)
```

which TP monitors can use to retrieve the xa_switch_t structure containing pointers to the XA functions.

14.1.2.1.2.1 Defining the OpenLink Resource Manager in Tuxedo

When integrating a new resource manager into Tuxedo, the file %TUXDIR%/udataobj/RM must be updated to include information about the resource manager. To integrate the OpenLink Lite Driver for Oracle into Tuxedo on Windows, define an*rm_alias* entry in the RM file by adding a line similar to:

OPL_ORA_XA;(*opl_xa_switch());"c:\program files\openlink\lite32\ntl5o9zz.lib

where, in this example, OPL_ORA_XA is the*rm_alias*.. ntl5o9zz.lib is an import library for the ODBC driver associated with the DSN identified by the xa_info string; in this example the OpenLink Windows Lite Driver for Oracle (ntl5o9zz.dll).

To create an*rm_alias*entry in the RM file for the OpenLink Generic Client add a line similar to:

OPL_XA;(*opl_xa_switch());"c:\program files\openlink\olod5032u.lib

On Unix the*rm_alias*is created in a similar way. A line adding an alias for the OpenLink Generic Client is added to $TUXDIR/udataobj/RM as follows:

OPL_XA:(*opl_xa_switch()):/home/openlink/lib/oplodbcu.so

Where /home/openlink is the openlink installation directory.

14.1.2.1.2.2 xa_info, OPENINFO and CLOSEINFO String Formats

The XA functions xa_open and xa_close open or close a connection to a resource manager for the calling thread. Both functions accept an*xa_info*argument.*xa_info*is a string containing instance-specific information for the resource manager. For OpenLink ODBC drivers, the *xa_info*string takes the form of an ODBC Data Source Name. The DSN must be defined in your ODBC.INI file or registry hive using the ODBC Administrator.

In a Tuxedo environment, the xa_info strings required by xa_open and xa_close for a particular RM instance are identified by the OPENINFO and CLOSEINFO entries in the UBBCONFIG configuration file. (An example UBBCONFIG file is given below.) OPENINFO and CLOSEINFO entries must be defined for each server group. These entries must be quoted strings of the form "*rm_alias*:*DSN*". (OPENINFO and CLOSEINFO should not contain the usual*xa_info*string required by the target database. This string is stored as part of the DSN definition when the ODBC data source is configured. The OpenLink ODBC driver retrieves the stored*xa_info*string using the DSN it receives through xa_open or xa_close.)

14.1.2.1.2.3 Using ODBC in an XA Global Transaction

In order for a service to use ODBC functions in the context of a global transaction, the ODBC functions must use the XA connection created when the service connects to the resource manager with which it is linked. (In the sample application, each service connects to a database using tpopen(), when the service is first initialized through tpsvrinit()). OpenLink XA-enabled ODBC drivers provide two functions, SQLGetXaEnv() and SQLGetXaConnect(), which make an XA connection available for use with ODBC. If the XA connection is enlisted in a global transaction, e.g. the client has called tpbegin() before calling the service, then any 'work' done by ODBC functions in the service will be controlled by that global transaction.

SQLGetXaEnv() returns the ODBC environment handle associated with the XA connection. SQLGetXaConnect() returns the ODBC connection handle associated with the XA connection. The prototypes for these functions are:

```
SQLRETURN SQLGetXaEnv(SQLHENV *phenv);
    SQLRETURN SQLGetXaConnect(SQLHDBC *phdbc);
```

When using OpenLink ODBC drivers in a global transaction context, you should not create an environment handle using SQLAllocHandle(), or use SQLSetEnvAttr(). (The SQLHENV associated with the XA connection is created with SQL_ATTR_ODBC_VERSION set to SQL_OV_ODBC3.) If you create an ODBC connection using this SQLHENV, it will not be enlisted in the global transaction.

14.1.2.2 JDBC

The OpenLink Multi-Tier JDBC drivers support distributed transactions using the Java Transaction API (JTA) as the Transaction Manager of the distributed transaction process.

14.1.2.2.1 Java Transaction API (JTA)

The OpenLink UDA Generic JDBC clients (opljdbc3.jar & opljdbc2.jar) support the industry standard XA specification for distributed transaction processing. The XA specification defines an interface between the transaction manager (TM) and resource manager (RM) in a distributed transaction system. This is a generic interface and it does not directly address the use of distributed transactions from Java. The Java mapping of the XA interface is defined in Sun Microsystems Java Transaction API (JTA) and JDBC 2.0/3.0 specifications. The OpenLink UDA Generic JDBC client drivers support the JTA architecture by providing the implementation of JTA resource manager interfaces.

The OpenLink UDA JDBC 3.0 driver provides the openlink.javax.OPLXid, openlink.javax.OPLXADataSource, and openlink.javax.OPLXAConnection classes which implement the interfaces javax.transaction.xa.Xid, javax.transaction.xa.XADataSource, javax.sql.XAConnection, and javax.sql.XAResource respectively. The use if these interfaces is usually transparent for applications and the application developer shouldn't bother with them. They are used only by the JTS transaction manager which normally runs as a part of the J2EE server.

14.1.2.2.1.1 JDBC 3.0 (JDK 1.4 & 1.5)

JTA support is built into the JRE for JDK 1.4 and 1.5, thus the UDA Generic JDBC client (opljdbc3.jar) simply needs to be added to the CLASSPATH and the XA classes will be available for use.

14.1.2.2.1.2 JDBC 2.0 (JDK 1.2 & 1.3)

JTA support was added by Sun Microsystems as an extension to the JDK 1.2 and 1.3 JRE's resulting in the following jar files having to be added to you CLASSPATH in additon to the UDA Generic JDBC client (opljdbc2.jar) :

```
jdbc2_0-stdext.jar
jta-spec1_0_1.jar
jndi.jar
```

14.1.2.2.1.3 JDBC XA Documentaton

Reference documentation for the OpenLink JDBC XA implementation in the UDA JDBC Client.

14.1.3 Configuring Databases for XA support

14.1.3.1 Oracle

XA support has been tested against Oracle version 9i and 10g

14.1.3.1.1 Granting SELECT Privilege on V$XATRANS$ & DBA_PENDING_TRANSACTIONS Views

In the event that the TM needs to perform recovery, xa_recover will fail if you do not grant the SELECT privilege to the V$XATRANS$ view for all Oracle accounts that XA applications will use. If the view does not already exist in your Oracle installation then it can be manually loaded using the following SQL script which should be included in your Oracle installation:

```
$ORACLE_HOME/rdbms/admin/xaview.sql
```

The example below shows an extract from an Oracle XA Library trace file (for details of how to enable Oracle XA Library tracing in an OpenLink driver see here). Oracle returns error *ORA-00942 : table or view does not exist* because user scott does not have the necessary SELECT privilege on the V$XATRANS$ view.

```
ORACLE XA: Version 10.1.0.0.0. RM name = 'Oracle_XA'.
113956.2352:536.536.1:
xaoopen: xa_info=ORACLE_XA+Threads=true+SesTm=60+Acc=P/scott/tiger+DB=DB01BA5BF8+SQLNET=ORCL+DbgFl=0x1+Lo
113956.2352:536.536.1:
xaolgn_help: version#: 168821248 banner: Personal Oracle Database 10g Release 10.1.0.2.0 - Production
With the Partitioning, OLAP and Data Mining options
113956.2352:536.536.1:
xaoopen: return 0
113959.2352:536.536.1:
xaorecover: xids=0xf8f8f8, count=10, rmid=1, flags=0x1000000
113959.2352:536.536.1:
ORA-00942: table or view does not exist
113959.2352:536.536.1:
xaorecover: xaofetch rtn -3.
113959.2352:536.536.1:
xaoclose: xa_info=DSN=au49-ora10-carlv;UID=scott;PWD=tiger;+DB=DB01BA5BF8, rmid=1769209857, flags=0x0
113959.2352:536.536.1:
xaoclose: rtn 0
```

The syntax for adding the grant privilege to the 'scott' user would thus be:

```
grant select on V$XATRANS$ to scott;
```

In addtion, the "ORA-00942: table or view does not exist" error also occurs if select privileges are not granted on the DBA_PENDING_TRANSACTIONS view as detailed in this Oracle MetaLink Article, requiring the following command to be executed as a DBA user:

```
GRANT SELECT ON DBA_PENDING_TRANSACTIONS TO PUBLIC
```

Note that the V$XATRANS$ did not exist nor did the DBA_PENDING_TRANSACTIONS view have appropriate select privileges in Unix Oracle installations, although they are included in Windows installations by default.

14.1.3.2 Informix

XA support has been tested against versions 9.x and 10.x

14.1.3.2.1 Enable Database Logging

For XA to be supported with an Informix database, the database must be capable of supporting transactions, which for Informix means the database must have logging enabled.

14.1.3.2.2 Useful Links

- Old IBM document (1996), but may contain some useful information: http://www.umiacs.umd.edu/~jhu/DOCS/TX4.2/html/erzhae/erzhae18.htm

14.1.3.3 Sybase

XA support has been tested against Sybase version 12.5.1 and above.

The following instructions need to performed to set up a working XA environment on your database server:

- Ensure your Syabse Server is licensed for Distributed Transaction Management. This can be checked by executing the following query against you Sybase Server:

  ```
  select license_enabled('ASE_DTM')
  ```

The query will return "1", if you have the ASE_DTM license installed.

- Run "sp_configure 'enable DTM',1" , via isql to enable transactions.
- Run "sp_configure 'enable xact coordination',1" , via isql
- Run "grant role dtm_tm_role to USER_NAME" .
- To prevent deadlocks when running transactions, enable row level lock by default, by running "sp_configure 'lock scheme',0,datarows" via isql.

. Note, you must restart Adaptive Server for this changes to take effect.

14.1.3.3.1 Setting the Timeout for Detached Transactions

On the Sybase server, you can set the dtm detach timeout period, which sets the amount of time (in minutes) that a distributed transaction branch can remain in the detached state (without an associated execution thread). After this period, the DBMS automatically rolls back the transaction. The dtm detach timeout period applies to all transactions on the database server. It cannot be set for each transaction. For example, to automatically rollback transactions after being detached for 10 minutes, use the following command:

"sp_configure 'dtm detach timeout period', 10"

You should set the dtm detach timeout period higher than the transaction timeout to prevent the database server from rolling back the transaction before the transaction times out.

14.1.3.4 SQLServer

XA support was tested against the following Microsoft SQLServer versions:

MSSQL 6.5 RTM (WinNt4 Sp3)

MSSQL 7.0EE SP4 (Win2k AdvSrv Sp3)

MSSQL 2000 SP3a (Win2k Sp3)

Before using the Microsoft SQLServer XA features you must install and register the Microsoft SQLServer extented stored procedures, for this :

1) copy xp_oplxa.dll (xp_oplxa65.dll for MSSQL 6.5) to the MSSQLServer_Root/binn directory

2) execute the command :

isql.exe -Usa -Psa_pwd -ig:xp_oplxa.sql

or

isql.exe -Usa -Psa_pwd -ig:xp_oplxa65.sql (for MSSQL 6.5)

14.1.3.5 Ingres II

XA is supported with IngresII version 2.0 and higher and Ingres II DBMS does not require any additional configuration for XA support to be enabled.

The Ingres II XA support was tested against the following versions:

IngresII v2.0

IngresII v2.6

IngresII v3.0 (Open release)

14.1.4 OpenLink XA Connection String Formats

To connect to a database's XA interface or some other XA compliant RM, a TM must connect using xa_open(). The xa_open() call accepts as one of its arguments an xa_info string. When acting as an RM proxy, OpenLink drivers automatically construct the database dependent xa_info string from other connection information received through the driver's data access API. The constructed string is sufficient to open an XA connection, but may omit optional settings. These optional settings are database dependent, details of which are given below. They can be entered in the 'XA Info' text box in the driver's setup dialog.

14.1.4.1 ODBC

14.1.4.1.1 Oracle

For full details of the Oracle syntax for the xa_info string, refer to the "Oracle Application Developer's Guide - Fundamentals : Chapter 20 Working With Transaction Monitors With Oracle XA : Developing and Installing Applications That Use the XA Libraries : Defining the xa_open String".

The OpenLink driver builds an xa_info string of the form:

```
Oracle_XA+Threads=true+SesTm=60+SqlNet=serviceName+DB=dbName+Acc=P/username/password
```

Fields in the xa_info string automatically inserted by the OpenLink driver should not be present in the 'XA Info' string. You should not include clauses such as "DB=dbname" or "Acc=P/username/password" .

*serviceName*takes the value of the 'Connection String' textbox in the Single-Tier ODBC driver setup dialog or the 'NET x Service Name' textbox in the Multi-Tier ODBC driver setup dialog. If a value isn't specified in the respective setup dialog, *serviceName*defaults to the value of the ORACLE_SID environment variable.

In addition to the tracing facilities provided by the OpenLink driver, Oracle's own XA interface also supports tracing. Ordinarily a trace file is only created at this level when an error occurs, but it is possible to trace all calls to the Oracle XA interface by setting the DbgFl entry appropriately in the xa_info string. To enable tracing by the Oracle XA Library, use an XaInfo string similar to: DbgFl=0x1+LogDir=c:/ For full details of the Oracle XA Library trace facilities see the "Oracle Application Developer's Guide - Fundamentals : Chapter 20 Working With Transaction Monitors With Oracle XA : Troubleshooting XA Applications".

14.1.4.1.2 Informix

The OpenLink driver builds an xa_info string of the form:

```
DB=dbname@dbserver;USER=username;PASSWD=password;CON=sessionid;
```

No optional xa_info elements are supported.

14.1.4.1.3 Sybase

No optional xa_info elements are supported.

14.1.4.1.4 SQL Server

No optional xa_info elements are supported.

14.1.4.1.5 Ingres

No optional xa_info elements are supported.

14.1.5 Tracing XA calls in OpenLink ODBC Drivers

When logging is enabled in an XA-enabled OpenLink ODBC driver, an XA log file is opened in addition to the usual ODBC log file. For instance, when using a Windows Lite driver, if you specify a log file name of c:\ntl6o10zu.log in the Lite setup dialog, an XA log file named c:\ntl6o10zu.log-xa will also be created. Because all DSNs which use the same OpenLink driver share the same log file name, if two applications use the same driver simultaneously, the log output from one application will be lost. The last application to connect takes ownership of the log file. This problem can be overcome by including meta-characters in the log file name. This is an OpenLink v6 feature which allows applications to have their own log file. The meta-characters substitute additional information into the log file name. They can also be used when configuring logging for a OpenLink ODBC Lite driver or an OpenLink ODBC generic multi-tier client, on both Windows or Unix.)

Supported meta-characters are:

- $$ - inserts $ (Windows & Unix)
- $P - current process id (Windows & Unix)
- $U - inserts user name (Windows & Unix)
- $H - inserts C:\ (Windows) or user's home directory (Unix)
- $T - timestamp (yyyymmdd-hhmmss) (Windows & Unix)

For instance, when tracing XA, it is useful to include $P, e.g. c:\ntl6i10zu_$P.log, so that each process using the driver gets its own XA log file. This is important when using the MS DTC as the transaction monitor, as the DTC loads the ODBC driver at the same time as the application when connecting to an RM through ODBC.

14.1.6 Enabling XA Transactions on Windows XP SP2 and Windows Server 2003

In Windows 2000, the MSDTC runs as LocalSystem, a special account that is granted all possible privileges to the local computer on which it resides. However, if someone compromises the DTC or any other service running under LocalSystem, that person then has full Administrator access to the computer. To avoid this potential security problem, starting with Windows XP SP2 and Windows Server 2003, Microsoft changed the account under which MSDTC service runs to be "NT AUTHORITY\NetworkService". A service running as NetworkService has the same network access as a service running as System (i.e. the operating system), but has significantly reduced local access. (NetworkService services access local resources as members of the local Users group.) Microsoft also introduced other MSDTC security enhancements. As a result, to enable XA transactions for these environment, some additional steps are needed after installing OpenLink XA/MSDTC enabled drivers.

(If you are using the distributed transactions support in COM+/Enterprise Services components with OpenLink drivers, you must enable XA transactions.)

1) Enable XA transactions

To turn on support for XA transactions, follow these steps:

1. Open Component Services.
2. Expand the tree view to locate the computer where you want to turn on support for XA transactions (for example, My Computer).
3. Right-click the computer name, and then click Properties.
4. Click the MSDTC tab, and then click Security Configuration.
5. Under Security Settings, click to select the check box for XA Transactions to turn on this support.

2) Identify in the registry the XA DLLs you plan to use

Windows Server 2003 and XP SP2 provide a registry entry for specifying the XA DLLs that the MSDTC can use. For each XA enabled OpenLink ODBC driver you plan to use, you must create a registry named-value under the registry key HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\MSDTC\XADLL. In your registry named-value, Name is the file name of the XA DLL (in the format dllname.dll), Type is String (REG_SZ), and the value is the full path name (including the file name) of the DLL file.

e.g.

```
        [HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\MSDTC\XADLL]
        Key name: ntl6o10zu.dll
        Data: C:\program files\openlink\bin\ntl6o10zu.dll
```

3) Allow the NetworkService account access to the folder containing the XA DLL

Since NetworkService a very restricted account, you need to provide this account with access to the folder where the XA DLL is located; for instance by modifying the access control list (ACL) for the folder. The ACL for an object is generally found on the Security tab of the object's property sheet.

4) Ensure the system PATH includes the folders of any DLLs loaded by the XA DLL

The MSDTC must be able to locate any DLLs on which the XA DLL depends. The folders containing these DLLs must be included in the global PATH environment variable. (The required DLLs and their paths can be identified easily withDependencyWalkerafter enabling the 'Full Paths' option on the 'View' menu.) Correct the path using Control panel, System Properties, Environment Variables, System Variables, Path. A reboot may be necessary for the change to take effect.

In Windows versions supporting the XADLL registry entry, it is not necessary to include the location of the XA DLL itself in the system PATH, as the registry entry includes the DLL's full path. In all earlier versions of Windows, the system PATH must include the XA DLL's location.

14.1.6.1 Useful Links

Florin Lazar's Weblog - MSDTC Posts

Microsoft Windows XP Professional Resource Kit Documentation(See Part III Security, Chapter 16 Authorization and Access Control)

14.1.7 Using OpenLink Drivers with Tuxedo on Unix

See also "Using the OpenLink Oracle Lite Driver with Tuxedo on Windows" for additional information on configuring Tuxedo.

14.1.7.1 Defining an OpenLink Resource Manager in Tuxedo

When integrating an OpenLink Unix driver into Tuxedo, the file %TUXDIR%/udataobj/RM must be updated to include information about the driver. Add a line similar to one of the examples below:

14.1.7.1.1 OpenLink ODBC Generic Client for Unix (Multithreaded):

```
        OPL_XA:(*opl_xa_switch()):/home/openlink/lib/oplodbcu.so ;; Unicode Driver
        OPL_XA:(*opl_xa_switch()):/home/openlink/lib/oplodbc.so ;; Ansi Driver
```

14.1.7.1.2 OpenLink Unix ODBC Lite Driver for SQLServer or Sybase (Multithreaded):

```
        OPL_XA:(*opl_xa_switch()):/home/openlink/lib/sql_mt_lu.so ;; Unicode Driver
        OPL_XA:(*opl_xa_switch()):/home/openlink/lib/sql_mt_lt.so ;; Ansi Driver
```

14.1.7.1.3 OpenLink Unix ODBC Lite Driver for Informix 9 (Multithreaded):

```
        OPL_XA:(*opl_xa_switch()):/home/openlink/lib/inf9_mt_lu.so -lthxa -lthsql -lthasf -lthgen -lthos -lifgls
        OPL_XA:(*opl_xa_switch()):/home/openlink/lib/inf9_mt_lt.so -lthxa -lthsql -lthasf -lthgen -lthos -lifgls
```

14.1.7.1.4 OpenLink Unix ODBC Lite Driver for Oracle 10g (Multithreaded):

```
        OPL_XA:(*opl_xa_switch()):/home/openlink/lib/ora100_mt_lu.so -lclntsh -lnnz10 ;; Unicode Driver
        OPL_XA:(*opl_xa_switch()):/home/openlink/lib/ora100_mt_lt.so -lclntsh -lnnz10 ;; Ansi Driver
```

14.1.7.1.5 OpenLink Unix ODBC Lite Driver for Ingres II (Multithreaded):

```
        OPL_XA:(*opl_xa_switch()):/home/openlink/lib/ingii_mt_lt.so ;; Ansi Driver
```

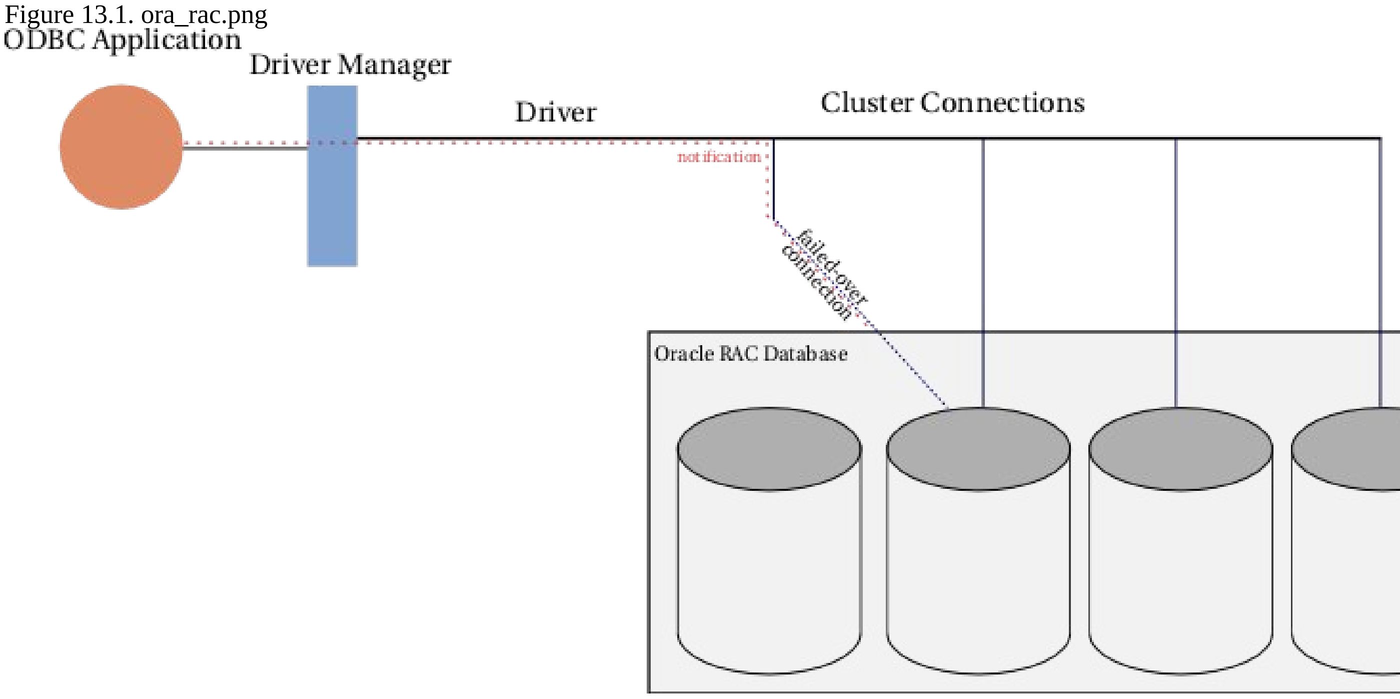
Note that due a limitation in the current Ingres II OpenAPI? XA implementation our Ingres II Lite driver cannot be used with Tuxedo as detailed in the OpenLink Ingres II XA Implementation Wiki document, in which case the MT Ingres II agent is the only available solution currently

where /home/openlink is the openlink installation directory and OPL_XA is the rm_alias. Any additional database libraries on which the driver depends (as shown by ldd) must be included after the driver shared library name. The example above for the Oracle Lite driver is for a driver linked against the Oracle 10g Instant Client.

15 Chapter 13. Real Application Cluster (RAC) / TAF Support

Abstract
Transparent Application Failover (TAF) is a protocol within Oracle whereby, if a connection to a database node fails, it can be re-established against an alternative node.

Figure 13.1. ora_rac.png



Once a broken connection has failed-over, an application can continue without any special action on its part. However, TAF does not restore all facets of a connection. For instance, TAF does not:

- restore active transactions - any active transactions are rolled back at the time of failure because TAF cannot preserve active transactions after failover: the application instead receives an error message until a ROLLBACK is submitted;
- restore session properties set using 'ALTER SESSION';
- maintain the state of server-side program variables, such as PL/SQL package states.

If any of these situations apply to a failed connection, an application may need to take action following failover to return the connection fully to the desired state. In order to do this, the application may request to be notified when failover has occurred. The OpenLink Lite Driver for Oracle 10g and the OpenLink Generic Multi-Tier Client provide this facility through TAF event notifications.

The Oracle 10g Lite Driver and Multi-Tier Agent also allow configuration of the failover retry interval and the maximum number of failover retries, in the event that failover is not successful on the first attempt.

Table of Contents

15.1 What you have to do to use it

15.1.1 Configuration

15.1.1.1 Multi-Tier

To use OpenLink's TAF event notification features or configure the failover retry parameters, you must enable OpenLink's TAF support. For the Oracle 10g Multi-Tier Agent, you must set the environment variable OPL_TAF_ENABLE to 'Y' or 'y'. With the agent, this is done by creating an entry in the appropriate 'Environment' section of the OpenLink Request Broker's rulebook, or by navigating through the web-based Administrative Assistant (normally running on <http://servername:8000/>) to configure an agent by form or wizard.

15.1.1.2 TAF Retry Configuration

When OpenLink's TAF support is enabled, in the event that a failover attempt fails, the Lite driver or Multi-Tier agent will instruct Oracle to retry. By default, the maximum number of failover attempts is 10 and the interval between retry attempts is 10 seconds. The default values can be overridden.

If using the Oracle 10g Lite Driver for Windows, enter the new values in the 'Max. Retries' and 'Retry Interval' text fields. For the Oracle 10g Lite Driver for Unix or the Multi-Tier Oracle 10g agent, override the defaults by setting the environment variables OPL_TAF_MAX_RETRIES and OPL_TAF_RETRY_INTERVAL. The retry settings operate independently of TAF event call-backs. When OpenLink's TAF support is enabled, they are effective irrespective of whether an application registers an event call-back.

15.1.2 Programmatic Considerations

OpenLink ODBC drivers notify ODBC applications that failover has occurred using two mechanisms: event call-backs and SQLSTATEs.

15.1.2.1 TAF Event Notification: Call-backs

An application can register a failover call-back routine by setting the proprietary connection attribute SQL_ATTR_EVENT_CALL-BACK (1280) on an open connection. The routine's address is supplied as the value of SQLSetConnectAttr ?'s *ValuePtr* argument. The same call-back routine can be registered for more than one connection. The call-back routine's signature must take the form:

```
void (*call-back) (oplevent_t oplEvent, SQLHANDLE handle, SQLUSMALLINT
eventInfo)
```

where:

- *oplEvent:* is an integer indicating the type of event which took place.
- *context:* is an application-supplied ODBC handle identifying the context the event is taking place in.
- *eventInfo:* is an unsigned integer code providing more information about the event. The values returned depend on the type of event. 0 signifies 'no further information available'

oplEvent is of type oplevent_t, an enumerated type enumerating the types of events reported to event call-backs. oplevent_t is defined as follows:

```
typedef enum {
 OPL_EV_NONE = 0, OPL_EV_FAILOVER_SUCCESS = 16, OPL_EV_FAILOVER_ABORT = 17 }
 oplevent_t;
```

At the moment, only failover events are supported through the OPL_EV_FAILOVER_xxx event class. Other types of event may be supported in the future using this call-back mechanism. OPL_EV_FAILOVER_SUCCESS indicates that failover was successful, OPL_EV_FAILOVER_ABORT that failover was aborted.

context is the ODBC handle (HDBC) of the connection for which the application wishes to receive event notifications. This handle should be supplied to the ODBC driver when the call-back is registered, using another Oracle-specific connection attribute, SQL_ATTR_EVENT_CONTEXT (1281). If this attribute is not set, the call-back receives SQL_NULL_HANDLE for the *handle* argument. (If other event types are supported in the future, this attribute may accept other types of ODBC handle, e.g. handles of type SQL_HANDLE_STMT, depending on the scope of the event.)

eventInfo is reserved for future use. All failover events currently return 0.

15.1.2.2 TAF Event Notification: SQLSTATEs

As well as an ODBC application being informed of failover through a call-back routine, it also receives notification through SQLSTATEs. After failover completes, the first ODBC call to return, on the affected connection, can return one of two proprietary SQLSTATES, IM500 or IM501:

- IM500 denotes successful failover completion. If the ODBC call generated no other errors, the call returns SQL_SUCCESS_WITH_INFO with SQLSTATE IM500 and is accompanied by the informational message "Failover completed". As usual, the SQLSTATE and diagnostic message can be retrieved using SQLGetDiagRec ?();

- IM501 ("Failover aborted") indicates failover was unsuccessful. A function returning this SQLSTATE returns with return code SQL_ERROR.

If the ODBC call returning the failover SQLSTATE fails for some reason, a diagnostic record holding the failover SQLSTATE and message is appended to any diagnostic records already generated by the failing call. In this case, even if failover was successful, a SQLSTATE IM500 may be accompanied by a function return code of SQL_ERROR. For instance, if a transaction was open at the time failover took place, SQLExecute may return SQL_ERROR with two diagnostic records, for example:

- DiagRec ? #1 Native error: 25402 SQLSTATE: HY000 Message: [OpenLink][ODBC][Oracle Server]ORA-25402: transaction must roll back
- DiagRec ? #2 Native error: 0 SQLSTATE: IM500 Message: [OpenLink][ODBC][Oracle Server]Failover completed.

Note: Failover notification using TAF-specific SQLSTATEs cannot be used independently of failover call-backs. The use of these SQLSTATEs is only triggered when an application registers a failover call-back.

15.2 Related Links

- Oracle
- TAF in Oracle 10g
- OCI FAQs (including TAF)
- Oracle 9i RAC/TAF

16 Chapter 14. OpenLink ODBC Driver Manager (iODBC SDK)

Abstract
This document provides information on linking your ODBC Applications with iODBC. iODBC is an alternative ODBC Driver Manager solution for developing ODBC components and applications for non-Windows systems (e.g. Mac OS Classic, Mac OS X, Linux....). iODBC is a free project licensed exclusively under the LGPL and BSD licenses, developed and maintained by OpenLink Software. iODBC permits non-Windows systems to communicate with databases via ODBC.

The ODBC API consists of a set of functions to enable any C/C++ applications linked against it to access and manage data. The ODBC specification is currently maintained by Microsoft Corporation.

iODBC is compliant with the actual Microsoft ODBC version 3.51.

iODBC SDKs are different for each platform. iODBC SDKs are available free of charge from the iODBC web site, released under the GNU Library General Public License (LGPL). The SDKs are made up of include files (.h), libraries for the corresponding platform, and a sample application for testing and demonstration (odbctest). The sample application is an Interactive Dynamic SQL Interpreter. Its sources are included for your use as you see fit.

There are no drivers included with the iODBC SDK, but you can find many on the OpenLink web site and from other third party middleware vendors.

You can find below a set of URLs for the right iODBC SDK regarding the platform (which is the operating system and the CPU) you are targeting at iodbc.org.

Table of Contents

16.1 iODBC SDK on Unix

Unlike Windows, Unix has historically never come with a system-wide ODBC installation, therefore there is greater scope for setting everything up the way you wish.

Also on Unix, there are two main-stream implementations of ODBC: UnixODBC and iODBC. The former comes with a QT-based implementation of a GUI, while iODBC provides a GTK+-based administrator.

The iODBC package is available in several formats - source tarball, source RPM, and a choice of binary components. Installing the binary packages is a simple exercise - for example,

```
sudo rpm -Uhv libiodbc-3.0.6-2.i386-glibc21.rpm
```

will install the iODBC runtime library.

There follows an alternative, walking through an unpacking, configure and build of the complete sources from tarball:

```
zsh, purple  6:10PM C/ % tar xvpfz libiodbc-3.0.6.tar.gz
libiodbc-3.0.6/
```

```
libiodbc-3.0.6/admin/
libiodbc-3.0.6/admin/Makefile.am
libiodbc-3.0.6/admin/Makefile.in
[snip]
zsh, purple  6:10PM C/ % cd libiodbc-3.0.6
zsh, purple  6:10PM libiodbc-3.0.6/ % ls
AUTHORS       LICENSE       NEWS           bin/          etc/        mac/
COPYING       LICENSE.BSD   README         configure*    include/    samples/
```

Configuration, starting with a check of interesting options specific to iODBC:

```
zsh, purple  6:10PM libiodbc-3.0.6/ % ./configure --help
Defaults for the options are specified in brackets.
Configuration:
  -h, --help              display this help and exit
      --help=short        display options specific to this package
      --help=recursive    display the short help of all the included packages
[snip]
--enable-fast-install=PKGS  optimize for fast installation default=yes
--disable-libtool-lock  avoid locking (might break parallel builds)
--enable-gui            build GUI applications (default),
--disable-gui           build GUI applications
--disable-gtktest       Do not try to compile and run a test GTK program
--enable-odbc3          build ODBC 3.x compatible driver manager (default)
--disable-odbc3         build ODBC 2.x compatible driver manager
[snip]
zsh, purple 6:10PM libiodbc-3.0.6/ % ./configure --prefix=/usr/local/stow/iodbc --with-gtk --enable-gui &
checking for a BSD-compatible install... /bin/install -c
checking whether build environment is sane... yes
checking for gawk... gawk
```

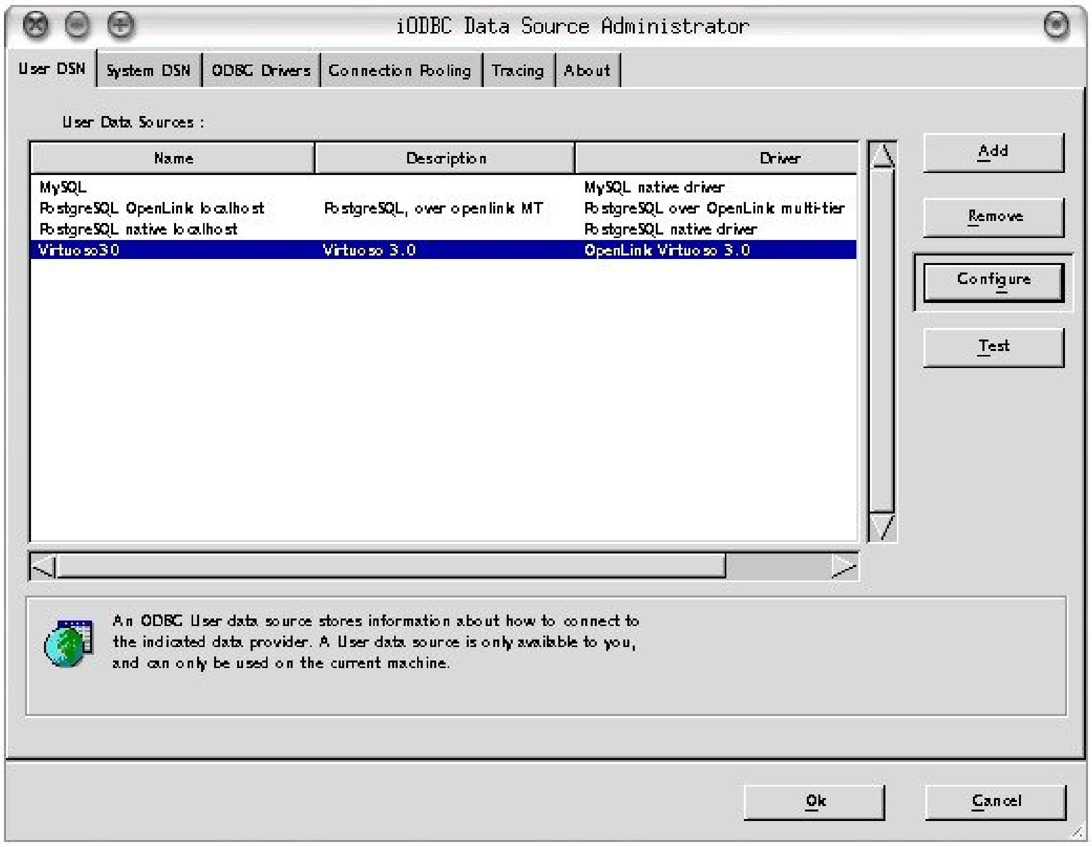
Installation, starting with becoming root by sudo, then stow-ing the package correctly into /usr/local:

```
zsh, purple  6:15PM libiodbc-3.0.6/ % sudo -s
Password:
zsh, purple  6:17PM libiodbc-3.0.6/ # make install
Making install in admin
make[1]: Entering directory `/home/tim/C/libiodbc-3.0.6/admin'
[snip]
make[2]: Nothing to be done for `install-exec-am'.
make[2]: Nothing to be done for `install-data-am'.
make[2]: Leaving directory `/home/tim/C/libiodbc-3.0.6'
make[1]: Leaving directory `/home/tim/C/libiodbc-3.0.6'
zsh, purple  6:18PM libiodbc-3.0.6/ # cd /usr/local/stow/
zsh, purple  6:18PM stow/ # chmod -R og=rX iodbc/
zsh, purple  6:18PM stow/ # stow iodbc/
zsh, purple  6:18PM stow/ # ^D
zsh, purple  6:18PM libiodbc-3.0.6/ % ls /usr/local/stow/iodbc/bin/
iodbc-config*  iodbcadm-gtk*  odbctest*
```

16.2 Configuring Data Sources

The graphical (GTK+-based) configuration screen makes setting up your data-sources (DSNs) quite intuitive, especially if you're used to the setup screens on Windows:

Figure 14.1. iODBC GTK Administrator

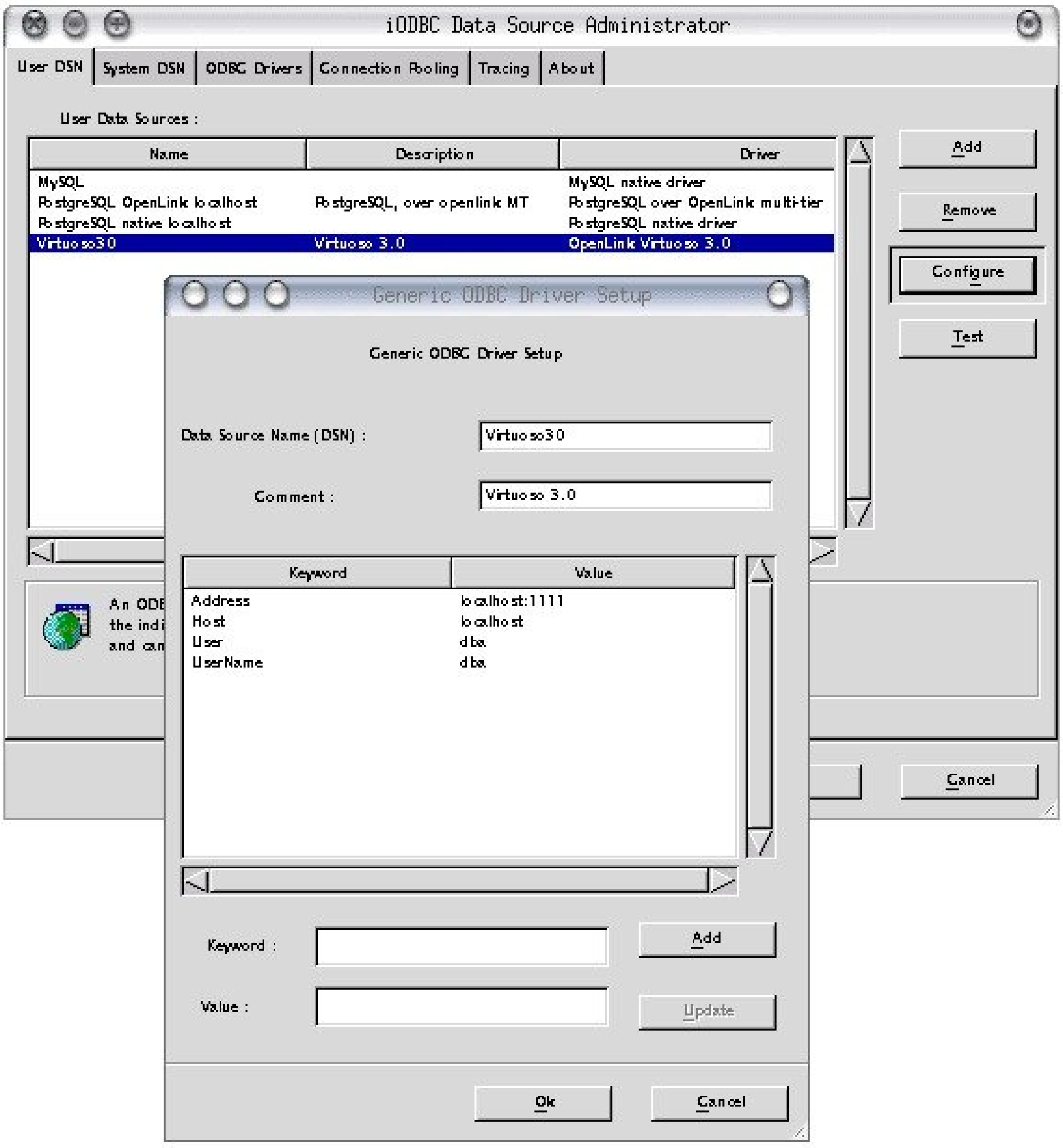


User Data Sources :

Name	Description	Driver
MySQL		MySQL native driver
PostgreSQL OpenLink localhost	PostgreSQL, over openlink MT	PostgreSQL over OpenLink multi-tier
PostgreSQL native localhost		PostgreSQL native driver
Virtuoso30	Virtuoso 3.0	OpenLink Virtuoso 3.0

An ODBC User data source stores information about how to connect to the indicated data provider. A User data source is only available to you, and can only be used on the current machine.

Figure 14.2. iODBC GTK Administrator



User Data Sources :

Name	Description	Driver
MySQL		MySQL native driver
PostgreSQL OpenLink localhost	PostgreSQL, over openlink MT	PostgreSQL over OpenLink multi-tier
PostgreSQL native localhost		PostgreSQL native driver
Virtuoso30	Virtuoso 3.0	OpenLink Virtuoso 3.0

Generic ODBC Driver Setup

Data Source Name (DSN) : Virtuoso30

Comment : Virtuoso 3.0

Keyword	Value
Address	localhost:1111
Host	localhost
User	dba
UserName	dba

Keyword :

Value :

An example of adding a new datasource follows:

16.2.1 The Configuration Files

The iODBC library searches for its DSN through a set few files:

$ODBCINI - the environment variable, if set

~/.odbc.ini - in your home-directory, if it exists

/etc/odbc.ini - a system-wide default

The format of this file is very simple; there are 3 sections, one for ODBC itself (setting up tracing), one for a list of DSNs, and one for the definitions of those DSNs, thus:

```
[ODBC]
Debug         = 1
Trace         = 0
DebugFile     = /home/tim/temp/odbc-debugfile.log
TraceFile     = /home/tim/temp/odbc-tracefile.log
TraceAutoStop = 1

[ODBC Data Sources]
Virtuoso30 = OpenLink Virtuoso 3.0

[Virtuoso30]
Description = Virtuoso 3.0
```

```
Driver      = /opt/opl/virtuoso-o12/lib/virtodbc.so
Address     = localhost:1111
UserName    = dba
User        = dba
```

Each DSN configured has an entry in the `ODBC Data Sources' section, and a complete definition in a paragraph section of its own.

There is also an ODBCINSTINI file; this contains descriptions of the ODBC drivers available.

```
[ODBC Drivers]
OpenLink Generic = installed
[OpenLink Generic]
Driver = /opt/openlink/odbcsdk/lib/oplodbc.so
```

16.2.2 Making a Test Connection

To test that a DSN connects correctly, you can use the supplied `odbctest' utility.

```
zsh, purple  3:58PM bin/ % ls
iodbc-config*  iodbcadm-gtk*  odbctest*
zsh, purple  3:58PM bin/ % echo $ODBCINI
/home/tim/.odbc.ini
zsh, purple  3:58PM bin/ % ./odbctest
iODBC Demonstration program
This program shows an interactive SQL processor
Enter ODBC connect string (? shows list):
Progress9.x(solaris)           | OpenLink Generic ODBC Driver
Progress9.x(solaris)           | OpenLink Generic ODBC Driver
pgsqlPurple                    | PostgreSQL native driver
pgsqlPurpleOpl                 | PostgreSQL using OpenLink driver
pgsqlPurpleVirtDemo            | Virtuoso database driver
SQLServer                      | OpenLink Generic ODBC Driver
Enter ODBC connect string (? shows list): DSN=pgsqlPurpleOpl
Driver: 04.50.0801 OpenLink Generic ODBC Driver (oplodbc.so)
SQL&gt;select count(*) from timtest;
count
-----------
100
 result set 1 returned 1 rows.
```

Any of the DSN attributes can be overridden in the connect-string, which takes the form

```
DSN=dsn_name[;attr=value]*
```

The attributes themselves depend on the database driver behind the DSN; normally they control the username (where the attribute could be called`userid' or `uid') and password (if specified) used to connect to the database, some form of server hostname specification (`host=' or `server='), and a means to identify a database instance on that server (`database='). A driver may also have custom attributes, such as FetchBufferSize, Port, etc.

16.2.3 Compiling Sample Program

To compile the sample "odbctest" application, follow these steps:

1. Navigate to the directory:

   ```
   odbcsdk/examples
   ```
2. Type the following commands to use the default Makefile and odbctest.c code:

   ```
   make clean
   make
   ```

16.2.4 Developing ODBC Applications

The ODBC SDK only implements an interface for the 'C' programming language.

To write an ODBC application,you must perform the following tasks:

1. Include the files "sql.h" and "sqlext.h" in your 'C' program(s).
2. Link the application with the following driver libraries:

   ```
   lib/libiodbc.a
   lib/libiodbc.so
   ```
3. At runtime, the ODBC drivers required for the connection are as follows (must be included in your shared library path):

   ```
   libiodbc.so.2.1.2
   libiodbc.so.2
   ```
4. Compile the program in the same directory as the "Makefile" file using the syntax:

   ```
   make
   ```

Note: Some UNIX systems also need -lsocket, -lnsl_s or both.

16.2.5 Further Reading:

"Data Management: SQL Call Level Interface (CLI)"

from X/Open in conjunction with SQL Access Group

ISBN: 1-872630-63-4

X/Open Document Number: S203

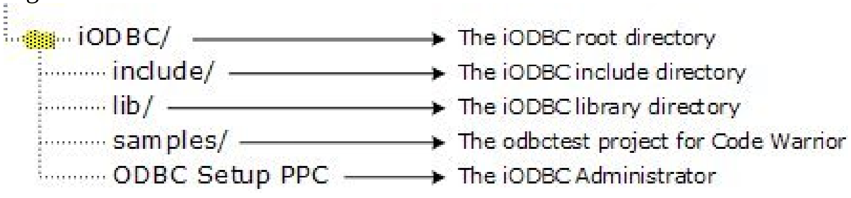
Microsoft ODBC API documentation: http://www.microsoft.com/data/odbc/default.htm

16.3 Linking iODBC and ODBC Applications on Mac OS

16.3.1 Mac OS Classic

The package provided for this platform is a compressed Macintosh format (.sit.hqx). After uncompressing it on your hard drive, you will find below the tree of the iODBC SDK as installed on Mac OS Classic :

Figure 14.3. iODBC SDK Tree for Mac OS Classic


iODBC/
The iODBC root directory
include/
The iODBC include directory
lib/
The iODBC library directory
samples/
The odbctest project for Code Warrior
ODBC Setup PPC
The iODBC Administrator

iODBC is cut down in three parts, and each one has its proper set of include files and libraries.

The iODBC Driver Manager is the core of the ODBC driver manager. Its main task is to load run the right function with the right ODBC driver.

The iODBC Configuration Manager is the ODBC configuration manager. Its main task is to access and manage the DSN configuration files.

The iODBC GUI Manager is the GUI manager. Its main task is to give to the user the ability to access all the ODBC functionality through a GUI.

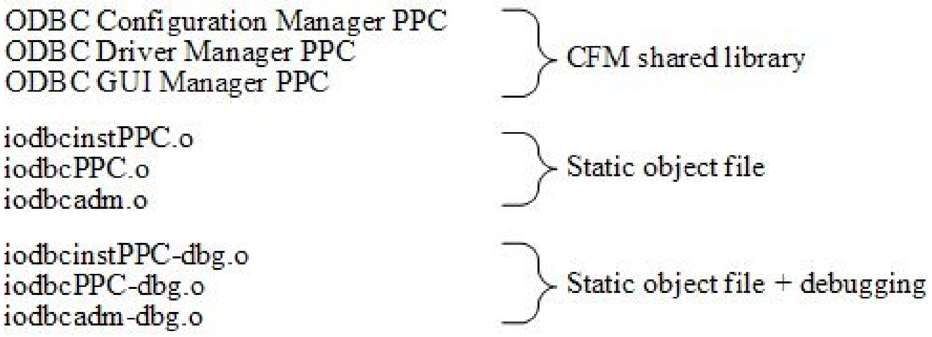
Under the include directory, you will find all the iODBC header files that can be used in your source. The following are the main header files: `isql.h, isqlext.h, isqltypes.h, iodbcadm.h, iodbcinst.h`

These are generally only a proxy for the other header files provided in the same directory: `sql.h, sqlext.h, sqltypes.h, odbcinst.h`

To access all these include files, simply add to your compiler an include path that points to `iODBC/include`. This is compiler dependent, so you may have to look into your compiler's documentation. For Code Warrior, you can do this via the Project Preferences panel.

Under the lib directory, you will find all the iODBC libraries needed to link against your application. All the parts explained before, are shipped through three different forms: CFM shared library, static object file, and static object file with debugging information.

Figure 14.4. iODBC SDK libraries on Mac



You will have to link your application with only one set of libraries. If you choose the CFM shared library set, ensure that at runtime these files are in the System-Extensions folder of your Mac Classic system.

To access the right set of library files, add a library path that points to `iODBC/lib`. This is compiler dependent, consult your compiler documentation. For Code Warrior, you can do this through the Project Preferences panel.

16.3.2 Mac OS X

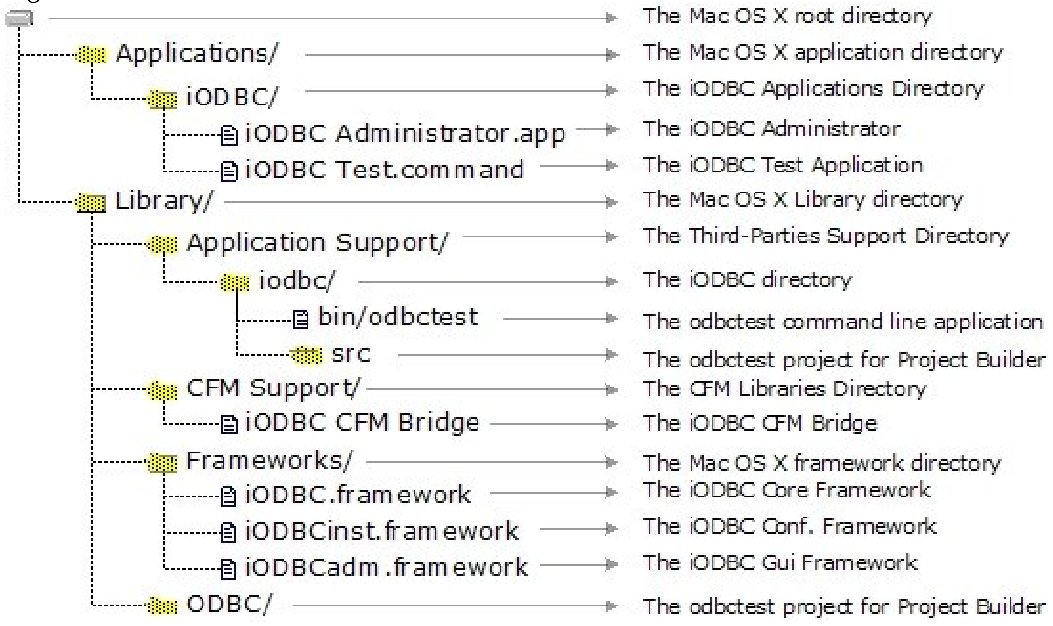
The package provided for this platform is a Macintosh Installer package (.mpkg) inside a disk-image (.dmg) file. After mounting the disk-image by double clicking on it, you will have to launch the Macintosh Installer package (.mpkg). You will be guided through the installation via the Macintosh Installer application.

16.3.3 See Also:

iODBC follows the Programmer and System guidelines defined by Apple, this document can be found at: developer.apple.com

You will find the iODBC SDK installation in the following directory tree structure as installed on Mac OS X:

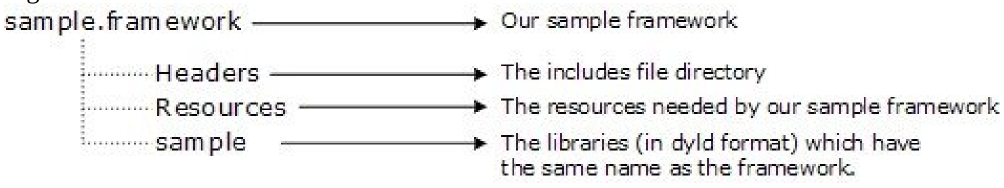
Figure 14.5. iODBC SDK tree on Mac OS X


The Mac OS X root directory
Applications/
The Mac OS X application directory
iODBC/
The iODBC Applications Directory
iODBC Administrator.app
The iODBC Administrator
iODBC Test.command
The iODBC Test Application
Library/
The Mac OS X Library directory
Application Support/
The Third-Parties Support Directory
iodbc/
The iODBC directory
bin/odbctest
The odbctest command line application
src
The odbctest project for Project Builder
CFM Support/
The CFM Libraries Directory
iODBC CFM Bridge
The iODBC CFM Bridge
Frameworks/
The Mac OS X framework directory
iODBC.framework
The iODBC Core Framework
iODBCinst.framework
The iODBC Conf. Framework
iODBCadm.framework
The iODBC Gui Framework
ODBC/
The odbctest project for Project Builder

In Mac OS X, "frameworks" are the central components of all SDKs. They provide everything needed by developers from include files to shared libraries with which you need to link your application. All frameworks support is well managed with Project Builder, shipped by Apple. Simply drag and drop via the provided GUI the corresponding framework you want use. If you want to use a framework through the command line compiler, see below.

The framework is basically a directory, structured as below:

Figure 14.6. iODBC SDK Frameworks on Mac OS X


sample.framework
Our sample framework
Headers
The includes file directory
Resources
The resources needed by our sample framework
sample
The libraries (in dyld format) which have the same name as the framework.

When you need to include one of the framework header files, you have to prepend your header file name with the framework name as below :

```
#include <samples/my_sample_header.h>
```

iODBC is broken down into three frameworks, each having its own proper set of includes files and libraries.

The iODBC.framework is the core of the ODBC driver manager. Its main task is to load run the right function with the right ODBC driver.

The iODBCinst.framework is the ODBC configuration manager. Its main task is to access and manage the DSN files.

The iODBCadm.framework is the GUI manager. Its main task is to give to the user the ability to access all the ODBC functionality through a GUI.

First of all, you have to be sure that the compiler and linker you are using can manage frameworks correctly.

16.3.4 Note:

The iODBC SDK is shipped with the `odbctest` sample application including sources and project file, which is intended as an example of an application utilizing iODBC.

Under Project Builder, you have to put all the frameworks that your application will use into your project. Implicitly, all these frameworks will be used. Also, you have to be sure that the machine where the application will be running includes all the frameworks with which you are linking. In certain cases, iODBC can be shipped with your application and all the frameworks.

This is the most straightforward way, but you can also directly use the compiler and command line linker shipped with Project Builder. In such case, you will have to add certain options to let the compiler and linker know that you want to use a specific framework. However, the sources do not require any changes. During the compilation phase, you will have to use the " `F`" option to add a path to let the compiler search for frameworks under the designated path. The correct syntax is " `Fdir`", where `dir` is the path you want to add. You have to put one path, and there is no limit on the number of `F` options to pass on the command line. For iODBC, you have to add only the following two directories:

```
/Library/Frameworks
/System/Library/Frameworks
```

A complete cc compilation command line may look like:

```
cc  c " F/Library/Frameworks" " F/System/Library/Frameworks" mysources.c
```

During the linkage phase, you will have to use the same option as above, as well as specifying at the end of the command line all the frameworks with which you are linking, preceded by the option `framework`.

A complete `cc` linkage command line may look like:

```
cc  o "myapplication" " F/Library/Frameworks" " F/System/Library/Frameworks"  \
       framework "iODBC"  framework "iODBCinst"  framework "iODBCadm"
```

16.3.5 References

iODBC web site

OpenLink Software web site

ODBC web site

Apple's Developer web site

16.3.6 Porting Mac OS Classic ODBC applications to Mac OS X

Carbon is a framework included with Apple's Mac OS X operating system that is derived from the original Macintosh Toolbox programming interface. Carbon on Mac OS X is an entirely new implementation, with a superset of the original Toolbox functionality, and many changes and additions to run under the Mac OS X operating system.

A subset of Carbon is also available on previous versions of the Mac OS, all the way back to Mac OS 8.1 or Mac OS 8.6 depending on the functionality required. This enables developers to create applications from one set of source code that will run on both the original Macintosh operating system as well as Mac OS X. Carbon applications can even take advantage of special Mac OS X features when running on Mac OS X, and continue to run as they have otherwise on earlier versions of the Mac OS.

Cocoa is a rich set of object-oriented frameworks that allow for the most rapid development of applications on Mac OS X. Using Cocoa, developers can create full-featured applications in a fraction of the time they would need for procedural languages. Applications from UNIX and other OS platforms can also be brought to Mac OS X quickly by using Cocoa to build state-of-the-art Aqua user interfaces while retaining most existing core code.

The Cocoa application environment is designed specifically for Mac OS X-only native applications. It is comprised of a set of object-oriented frameworks that support rapid development and high productivity. The Cocoa frameworks include a full featured set of classes designed to create robust and powerful Mac OS X applications. The object-oriented design simplifies application development and debugging.

Figure 14.7. iODBC SDK on Mac

16.3.7 See Also:

Cocoa

Carbon

16.3.7.1 Migration of iODBC applications to Carbon

The iODBC CFM Bridge is the library that Carbon applications must be linked against. This bridge is responsible for redirecting the ODBC API calls to the right iODBC version applicable to platform that the application is running on, beit Mac OS Classic or Mac OS X.

The iODBC CFM Bridge must be used exclusively. If you link against the iODBC CFM Bridge, ensure that no others iODBC libraries are linked in your project. The iODBC CFM Bridge linked applications require a valid iODBC installation on the target platform in order run successfully.

Any CFM libraries used must be included in your System-Extensions folder of your Mac Classic system in order to successfully support carbon applications at runtime.

16.3.7.2 Migration of iODBC applications to Cocoa

Under Project Builder, you have to put all the frameworks that your application will use in your project. Implicitly, all these frameworks will be used. Also, you have to be sure that the machine where you will run the application include all the frameworks you link with. In such cases, iODBC can be shipped with your application including all the frameworks.

This is the most straight forward way, but you can also directly use the compiler and linker command line shipped with

Project Builder. In a such case, you will have to add some options to inform the compiler and linker that you want to use a specific framework. The sources sources do not have to change.

During the compilation phase, use the `-Fdir` option on the compiler command line to add a search path to the framework path, where dir is the path you want to add. You may only specify one path per `-F` option, but you not limited to the number of `-F` options used. iODBC compilations only require `-F/Library/Frameworks` and `-F/System/Library/Frameworks` options. Hence, complete `cc`= compilation command line may look like:

```
cc -c "-F/Library/Frameworks" "-F/System/Library/Frameworks" ... mysources.c
```

During the linkage phase, you will have to use the same option as above, and to specify at the end of the command line all the framework you want to link with, preceded with the option -framework . A complete cc linkage command line may look like:

```
cc -o "myapplication" "-F/Library/Frameworks" "-F/System/Library/Frameworks" ... \
  -framework "iODBC" -framework "iODBCinst" -framework "iODBCadm"
```

17 Chapter 15. Sample Applications

Abstract
Information on installing and using the supplied samples.

Table of Contents

A number of sample applications are bundled with your Universal Data Access Driver Suite installation for the following purposes:

- To simplifying the process of getting the product up and running
- To accelerate the support case creation and resolution process
- To demonstrate the Data Access Driver Suite's unique product features highlighting the benefits it brings to your organization
- To demonstrate application programming techniques that can used to aid and assist your ODBC and JDBC programmers

The Universal Data Access Driver Suite services are consumed primarily via ODBC and JDBC applications (OLE-DB applications connect via ODBC Data Providers for OLE-DB), thus separate ODBC & JDBC sample applications (including source code) have been packaged and integrated into the installer. The current list of sample applications include:

- C++ Demo. an ODBC based Interactive SQL processor written in C++.
- ODBC Bench Test. a 32 Bit C++ program based on the industry standard TPC-A benchmark (we will be extending this program to include the TPC-C and TPC-D benchmarks also). This program helps you compare the performance of Virtuoso against other backend database engines as well as compare the performance of various ODBC Drivers connecting to any ODBC compliant backend database.
- ODBCTEST. ODBC based Interactive SQL processor written in 'C' for Linux & UNIX
- JDBCDemo. a JDBC sample application that demonstrates an SQL query.
- Rowset Demo. a JDBC application that demonstrates OpenLink's JDBC Driver with the OpenLink Scrollable Cursor RowSet Extension to the JDBC specification.
- ScrollDemo2. a JDBC 2.0 sample application that demonstrates OpenLink's support of Scrollable Cursors and its ability to perform scrollable cursor operations across heterogeneous databases.
- JBench. a Java and JDBC based adaptations of the industry standard TPC-A benchmarks. This program helps you compare the performance of OpenLink's Virtuoso against other backend database engines, it also helps you to compare the performance of various JDBC Drivers connecting to any JDBC compliant backend database.

17.1 Binary & Source File Locations

17.1.1 ODBC Demonstration Applications

The binary executables of these sample applications reside under the following directory structure:

```
<OPENLINK_INSTALLATION_DIRECTORY>\samples\odbc
```

The source code of some of these sample applications, when available, reside under the following directory structure, for example:

```
<OPENLINK_INSTALLATION_DIRECTORY>\samples\odbc\cppdemo
```

17.1.2 JDBC Demonstration Applications

The binary executables (Java class files), and sources for these sample applications reside under the following directory structure:

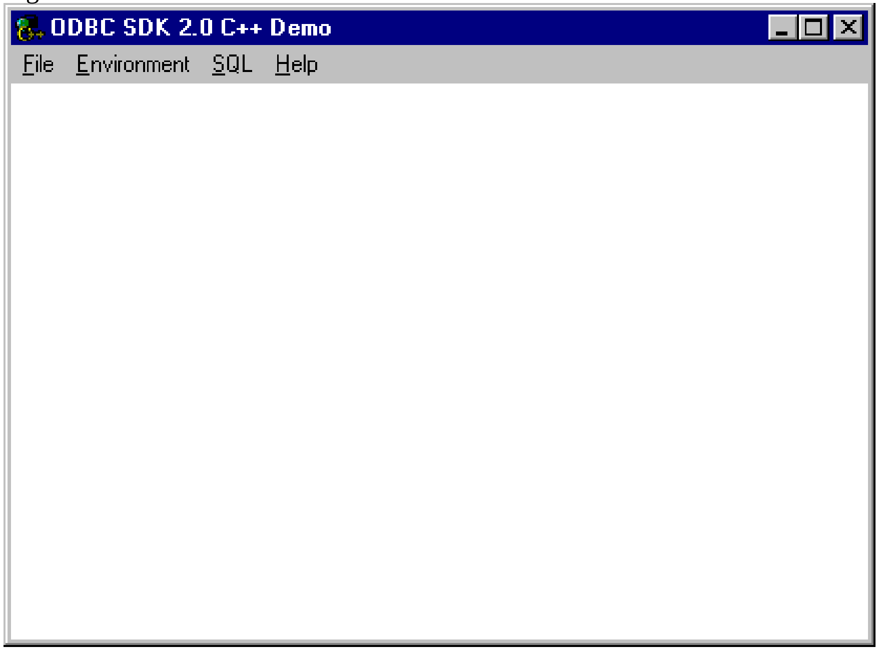
```
<OPENLINK_INSTALLATION_DIRECTORY>\samples\jdbc\<JDK_Version>\<Demo_name>
```

17.2 Windows 95/98/NT/2000 Based ODBC Sample Applications

17.2.1 C++ Demo

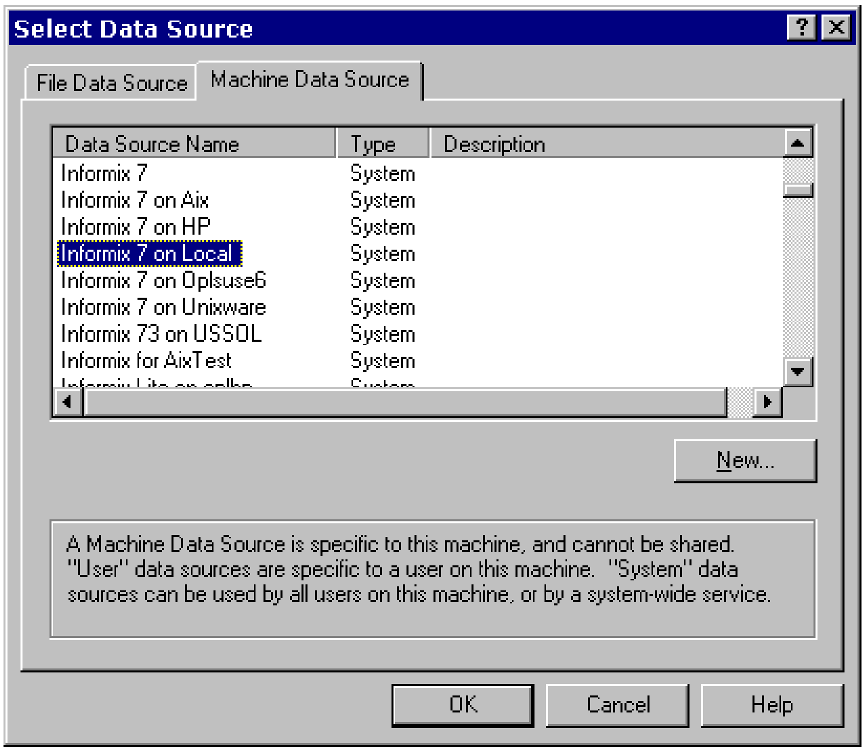
1. Go to the OpenLink Data Access Drivers "Start Menu" item, then click on the "C++ Demo 32 Bit" menu item.

   Figure 15.1.

2. Follow the Environment->Open Connection menu path. Selecting the "Open Connection" menu item results in the ODBC Driver Manager presenting you with a list of ODBC DSNs on your machine as depicted by the screen capture below:

   Figure 15.2.

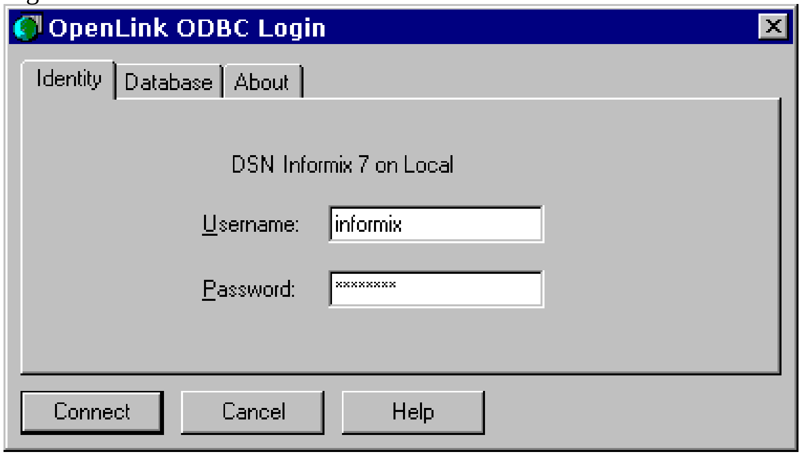


Data Source Name	Type	Description
Informix 7	System	
Informix 7 on Aix	System	
Informix 7 on HP	System	
Informix 7 on Local	System	
Informix 7 on Oplsuse6	System	
Informix 7 on Unixware	System	
Informix 73 on USSOL	System	
Informix for AixTest	System	

A Machine Data Source is specific to this machine, and cannot be shared. "User" data sources are specific to a user on this machine. "System" data sources can be used by all users on this machine, or by a system-wide service.

3. Select the ODBC DSN that you wish to connect to, (in this case "Informix 7 on Local" has been chosen. This will connect you to the Informix 7 database.)
4. You are then presented with a Login Dialog by the OpenLink Driver for ODBC, enter a valid user name and password into the appropriate fields.

Figure 15.3.



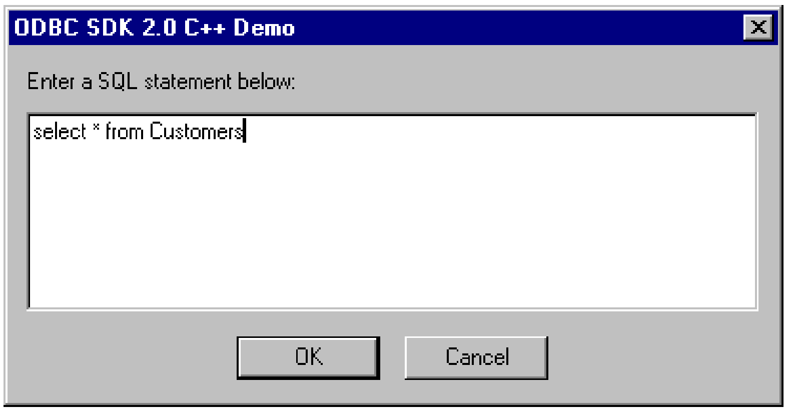
DSN Informix 7 on Local

Username: informix

Password: ********

5. At this point you will be connected to the chosen datasource, you can now use the SQL-->Execute SQL menu path to open up the Interactive SQL input dialog. Enter a valid SQL statement (see example in screen shot) and then click on the "OK" button.

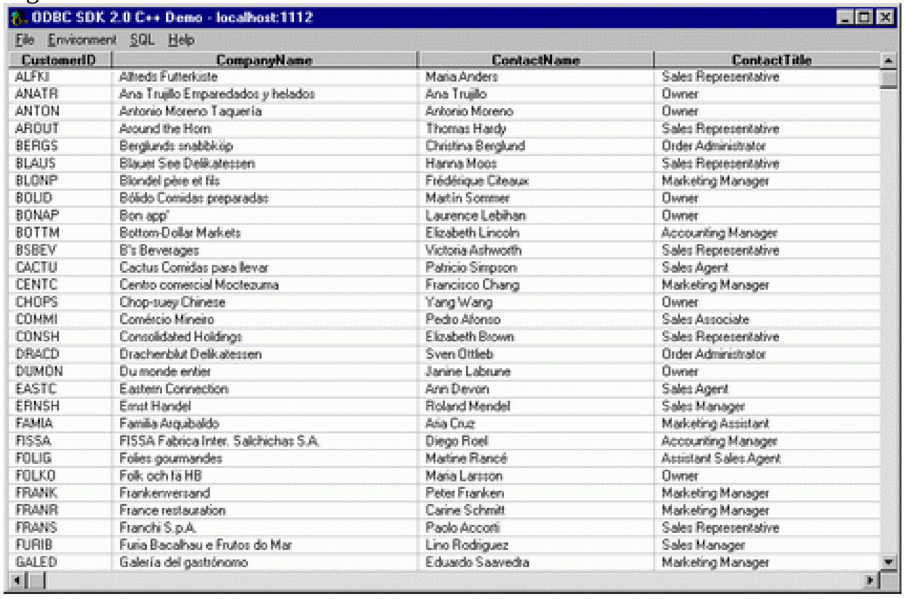
Figure 15.4.



```
select * from Customers
```

6. You will be presented with the results of your query.

Figure 15.5.



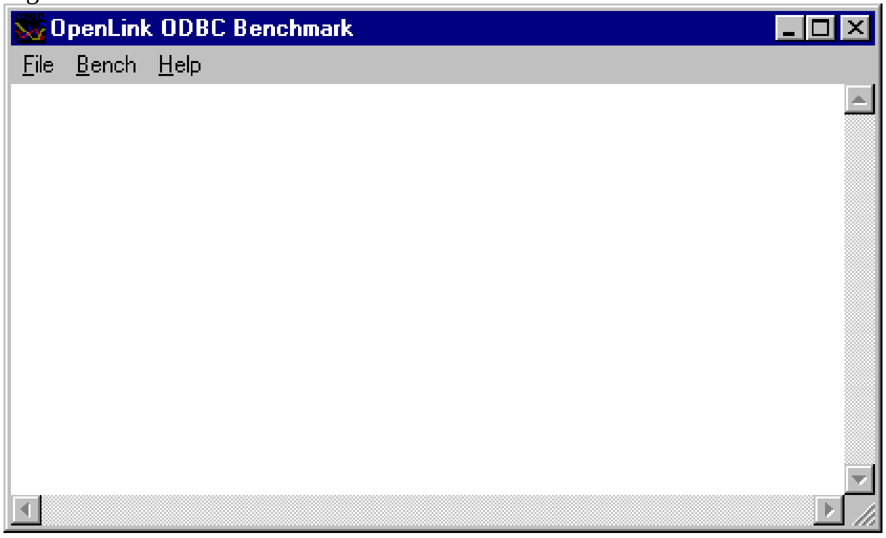
CustomerID	CompanyName	ContactName	ContactTitle
ALFKI	Alfreds Futterkiste	Maria Anders	Sales Representative
ANATR	Ana Trujillo Emparedados y helados	Ana Trujillo	Owner
ANTON	Antonio Moreno Taquería	Antonio Moreno	Owner
AROUT	Around the Horn	Thomas Hardy	Sales Representative
BERGS	Berglunds snabbköp	Christina Berglund	Order Administrator
BLAUS	Blauer See Delikatessen	Hanna Moos	Sales Representative
BLONP	Blondel père et fils	Frédérique Citeaux	Marketing Manager
BOLID	Bólido Comidas preparadas	Martin Sommer	Owner
BONAP	Bon app'	Laurence Lebihan	Owner
BOTTM	Bottom-Dollar Markets	Elizabeth Lincoln	Accounting Manager
BSBEV	B's Beverages	Victoria Ashworth	Sales Representative
CACTU	Cactus Comidas para llevar	Patricio Simpson	Sales Agent
CENTC	Centro comercial Moctezuma	Francisco Chang	Marketing Manager
CHOPS	Chop-suey Chinese	Yang Wang	Owner
COMMI	Comércio Mineiro	Pedro Afonso	Sales Associate
CONSH	Consolidated Holdings	Elizabeth Brown	Sales Representative
DRACD	Drachenblut Delikatessen	Sven Ottlieb	Order Administrator
DUMON	Du monde entier	Janine Labrune	Owner
EASTC	Eastern Connection	Ann Devon	Sales Agent
ERNSH	Ernst Handel	Roland Mendel	Sales Manager
FAMIA	Familia Arquibaldo	Aria Cruz	Marketing Assistant
FISSA	FISSA Fabrica Inter. Salchichas S.A.	Diego Roel	Accounting Manager
FOLIG	Folies gourmandes	Martine Rancé	Assistant Sales Agent
FOLKO	Folk och fä HB	Maria Larsson	Owner
FRANK	Frankenversand	Peter Franken	Marketing Manager
FRANR	France restauration	Carine Schmitt	Marketing Manager
FRANS	Franchi S.p.A.	Paolo Accorti	Sales Representative
FURIB	Furia Bacalhau e Frutos do Mar	Lino Rodriguez	Sales Manager
GALED	Galería del gastrónomo	Eduardo Saavedra	Marketing Manager

7. You exit this demo by following the Environment-->Close Connection menu path.

17.2.2 ODBC Bench Test 32

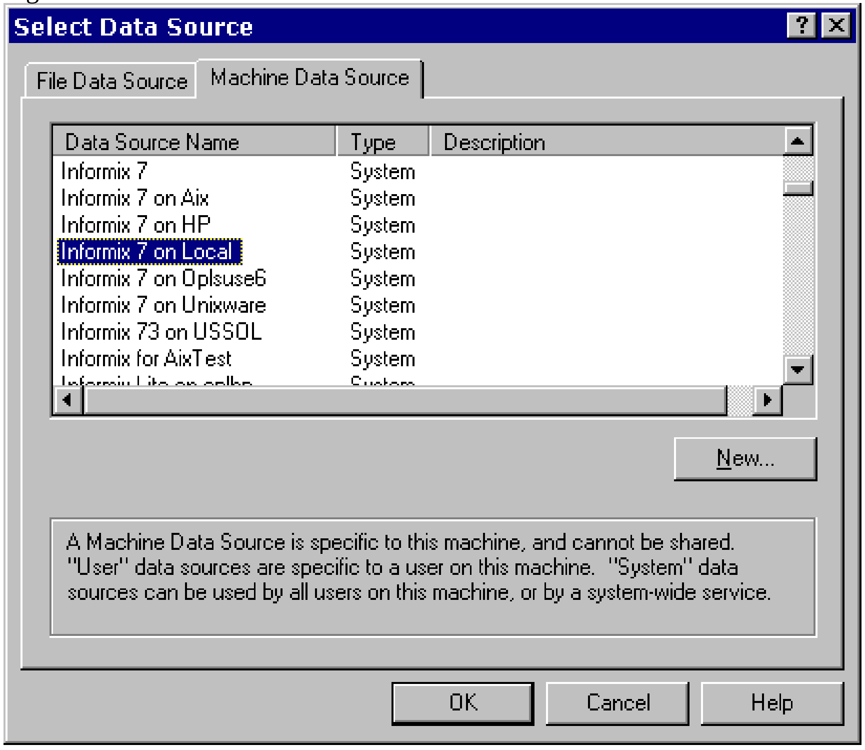
1. Go to the OpenLink Data Access Drivers "Start Menu" item, then click on the "ODBC Bench Test 32 Bit" menu item. You will be presented with the "Bench Test" interface.

Figure 15.6.



2. Follow the File->Connect menu path which initializes the ODBC Driver Manager, which in turn presents you with a list of ODBC DSNs installed on your machine. Select the DSN that you want to benchmark, remember that by benchmarking a DSN you are benchmarking the ODBC Driver that serves the DSN in question and the backend database engine that serves the ODBC Driver. Choose the name of the datasource you want to benchmark.

   Figure 15.7.

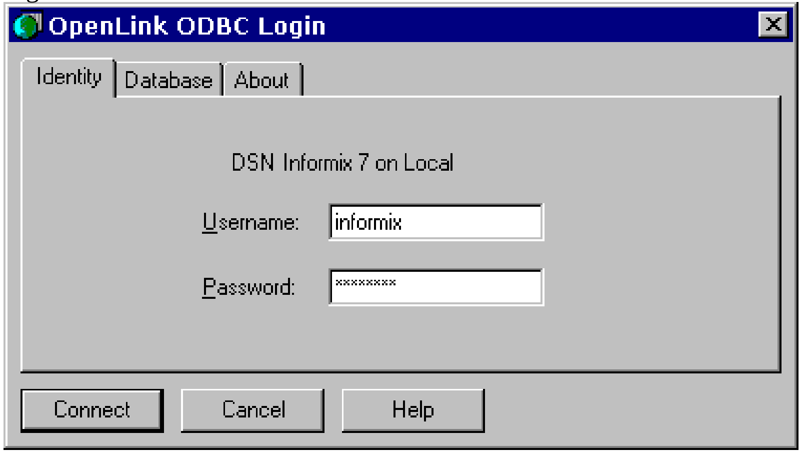
   

Data Source Name	Type	Description
Informix 7	System	
Informix 7 on Aix	System	
Informix 7 on HP	System	
Informix 7 on Local	System	
Informix 7 on Oplsuse6	System	
Informix 7 on Unixware	System	
Informix 73 on USSOL	System	
Informix for AixTest	System	

   A Machine Data Source is specific to this machine, and cannot be shared. "User" data sources are specific to a user on this machine. "System" data sources can be used by all users on this machine, or by a system-wide service.

3. You will then be presented with a Login Dialog by the OpenLink Generic Driver for ODBC, enter a valid user name and password into the appropriate fields.

   Figure 15.8.

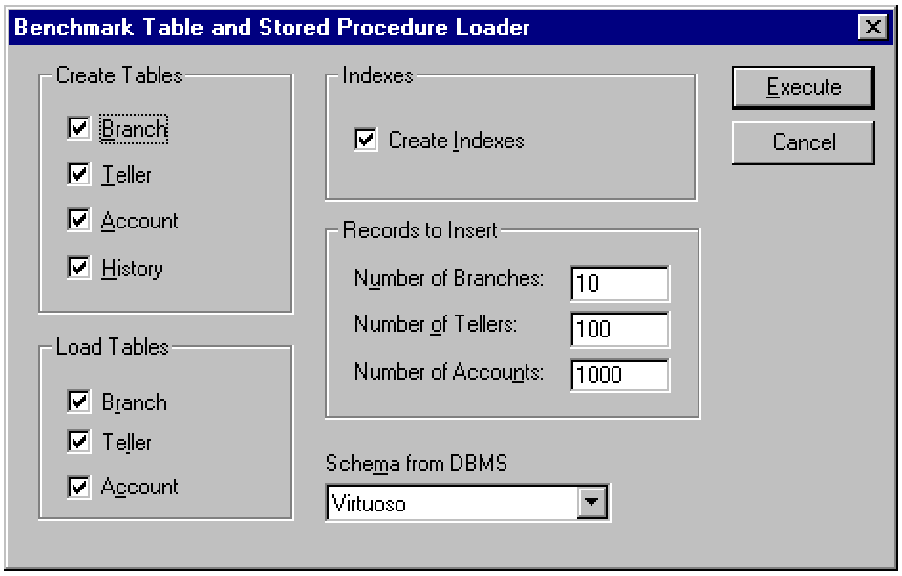
   DSN Informix 7 on Local

   Username: informix

   Password: ********

4. Now follow the Bench-->Load Tables menu path and you will be presented with a dialog that enables you to configure key elements of your benchmark. Click the "Execute" button to commence the process of setting up your database for the benchmark tests. Please make sure you select the appropriate schema for the DBMS that you are connecting to so that the benchmark tables are created properly.

   Figure 15.9.



Create Tables
- [✓] Branch
- [✓] Teller
- [✓] Account
- [✓] History

Indexes
- [✓] Create Indexes

Records to Insert

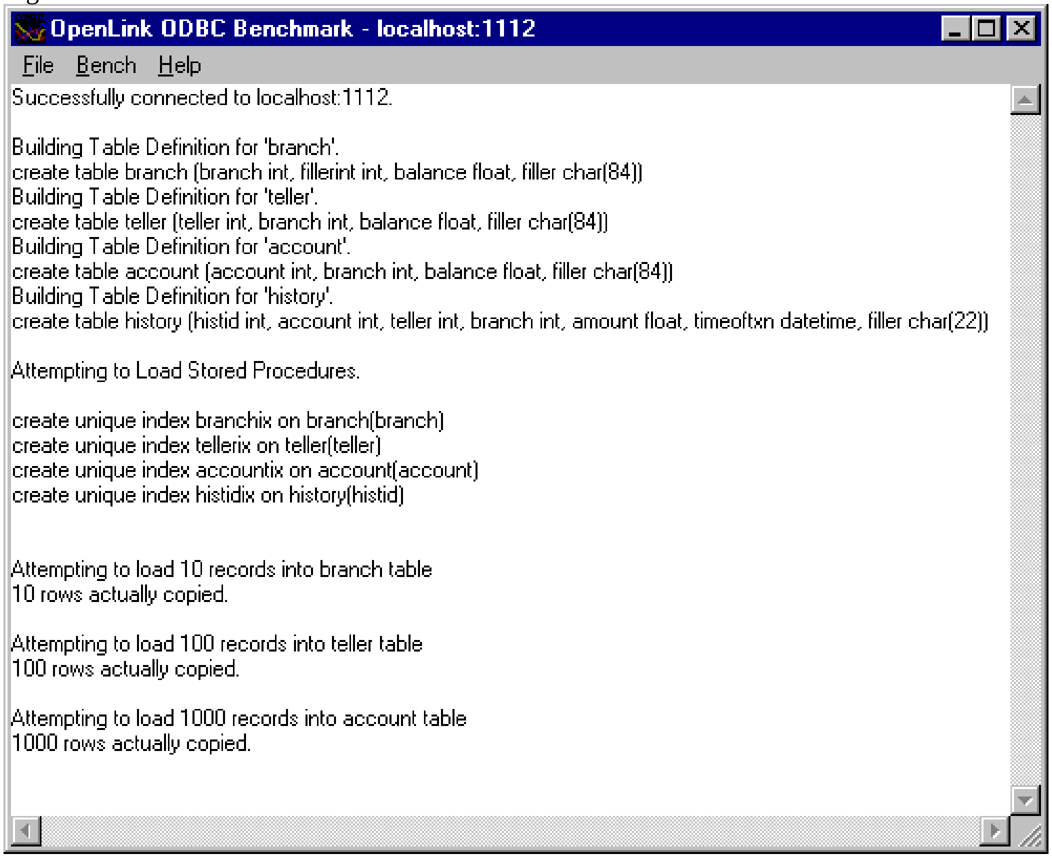
Number of Branches:	10
Number of Tellers:	100
Number of Accounts:	1000

Load Tables
- [✓] Branch
- [✓] Teller
- [✓] Account

Schema from DBMS: Virtuoso

5. As the process of loading data occurs, all the way up to completion, the benchmark program will provide status information into the benchmark output pane as shown below:

   Figure 15.10.

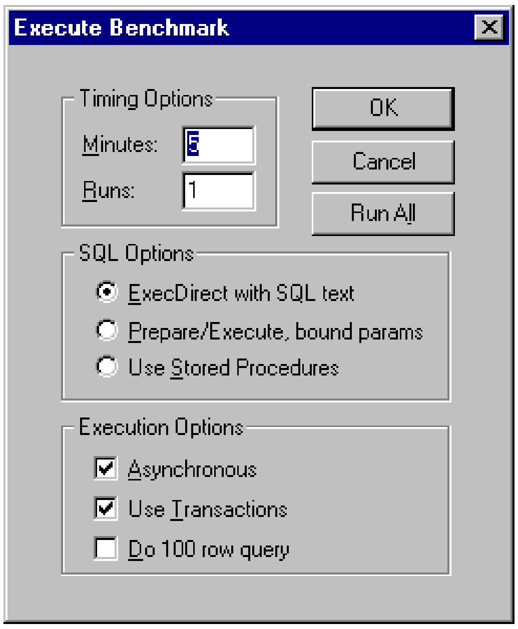


```
Successfully connected to localhost:1112.

Building Table Definition for 'branch'.
create table branch (branch int, fillerint int, balance float, filler char(84))
Building Table Definition for 'teller'.
create table teller (teller int, branch int, balance float, filler char(84))
Building Table Definition for 'account'.
create table account (account int, branch int, balance float, filler char(84))
Building Table Definition for 'history'.
create table history (histid int, account int, teller int, branch int, amount float, timeoftxn datetime, filler char(22))

Attempting to Load Stored Procedures.

create unique index branchix on branch(branch)
create unique index tellerix on teller(teller)
create unique index accountix on account(account)
create unique index histidix on history(histid)


Attempting to load 10 records into branch table
10 rows actually copied.

Attempting to load 100 records into teller table
100 rows actually copied.

Attempting to load 1000 records into account table
1000 rows actually copied.
```

6. Now that all the benchmark data has been loaded into your database, follow the Bench-->Run Benchmark menu path and then configure your actual benchmark session parameters:

   These benchmark parameters fall into 3 categories, Timing Options, SQL Options, and Execution Options.

   Figure 15.11.


Execute Benchmark

Timing Options
Minutes: 5
Runs: 1

SQL Options
(●) ExecDirect with SQL text
( ) Prepare/Execute, bound params
( ) Use Stored Procedures

Execution Options
[✓] Asynchronous
[✓] Use Transactions
[ ] Do 100 row query

17.2.2.1

17.2.2.1.1 Timing Options:

These setting allow you to configure the duration related aspects of this benchmark program

Minutes. this is the duration of each benchmark run

Runs. this controls how many iterations of the benchmarks you actually run (the default is one benchmark iteration with a duration of 5 minutes)

17.2.2.1.2 SQL Options:

These settings allow you to configure how your benchmark's SQL instructions are actually handled.

ExecDirect with SQL Text. this means that no form of repetitive SQL execution optimization is being applied (SQL statements are prepared and executed repetitively)

Prepare/Execute Bound Params. this means that the Parameter Binding SQL execution optimization is being applied (SQL is prepared once but executed many times without the overhead of re-preparing statements prior to execution)

Use Stored Procedures. this means that the Stored Procedure SQL optimization is being applied (benchmark instructions are stored within database being benchmarked)

17.2.2.1.3 Execution Options:

These settings allow you to configure the tone of your benchmark, for instance it could have Transaction scoping and a mix of record retrieval queries, or it could simply be input and update intensive with a minimal amount of record retrieval queries (the case when the 100 row query checkbox is unchecked a typical OLTP scenario)

Asynchronous. execute the benchmark instructions asynchronously

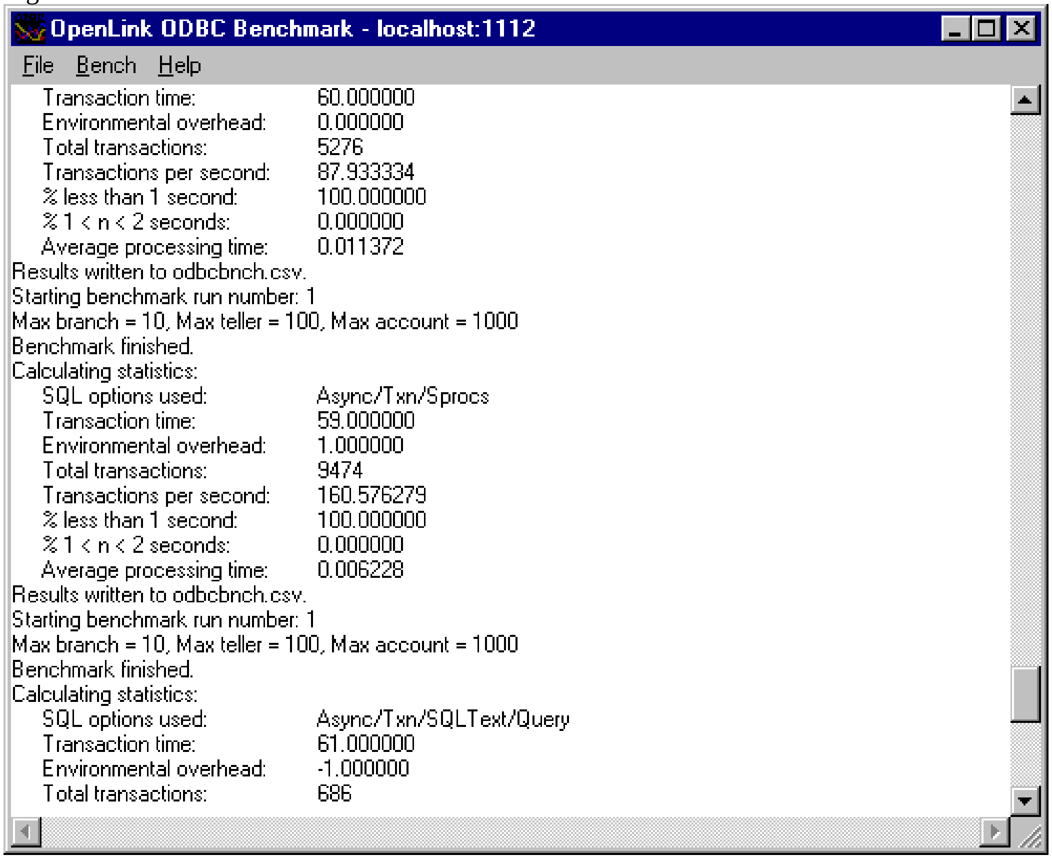
Use Transactions. make the benchmark use transaction control (instructions are scoped to transaction blocks)

Do 100 row Query. perform a simulation of a 100 record retrieval as part of the benchmark activity.

17.2.2.2

1. Click on the "Run All" button if you would like all the different benchmark type combinations to be performed.
2. When benchmark run complete benchmark data is written to the benchmark program's output pane.

Figure 15.12.



```
    Transaction time:            60.000000
    Environmental overhead:      0.000000
    Total transactions:          5276
    Transactions per second:     87.933334
    % less than 1 second:        100.000000
    % 1 < n < 2 seconds:         0.000000
    Average processing time:     0.011372
Results written to odbcbnch.csv.
Starting benchmark run number: 1
Max branch = 10, Max teller = 100, Max account = 1000
Benchmark finished.
Calculating statistics:
    SQL options used:            Async/Txn/Sprocs
    Transaction time:            59.000000
    Environmental overhead:      1.000000
    Total transactions:          9474
    Transactions per second:     160.576279
    % less than 1 second:        100.000000
    % 1 < n < 2 seconds:         0.000000
    Average processing time:     0.006228
Results written to odbcbnch.csv.
Starting benchmark run number: 1
Max branch = 10, Max teller = 100, Max account = 1000
Benchmark finished.
Calculating statistics:
    SQL options used:            Async/Txn/SQLText/Query
    Transaction time:            61.000000
    Environmental overhead:      -1.000000
    Total transactions:          686
```

The key pieces of benchmark data that you need to look out for are:

Total Transactions. total number of transactions completed during the benchmark run

Transactions Per Second. number of transaction completed per second for the benchmark run

Information from this benchmark is automatically written to an Excel format CSV (the file c:\odbcbnch.csv) which makes it easy for you to graph and pivot data collated from several benchmark runs. A later version of this demo will actually write the benchmark data into an ODBC DSN that you provide thereby offering even more flexibility and accessibility to benchmark data.

17.2.3 Linux & UNIX Based ODBC Sample Applications

17.2.3.1 ODBCTEST

This is a simple 'C' based and ODBC compliant Interactive SQL processor.

1. Run the script openlink.sh to set up your environment:

   ```
   openlink.sh
   ```
2. Start ODBCTEST by executing the following command:

   ```
   odbctest
   ```

   Or pass a DSN connect string, for example:

   ```
   odbctest DSN=marketing;UID=name;PWD=pwd;
   ```
3. At the SQL command prompt enter "?" for a list of ODBC DSNs on your machine or enter a valid ODBC Connect String e.g.

   If you have a DSN named "Marketing" you would enter: DSN=Marketing

17.2.3.2 ODBC Benchmark Application

The TPC-A, TPC-C, and TPC-D benchmarks are currently under development, please monitor our Web site (http://www.openlinksw.com) for updates on these applications.

17.2.4 Mac OS X

17.2.4.1 ODBCTEST:

This is a simple 'C' based and ODBC compliant Interactive SQL processor.

1. Open a Terminal session, and start ODBCTEST by executing the following command:

   ```
   /Library/iodbc/bin/odbctest
   ```
2. At the SQL command prompt enter "?" for a list of ODBC DSNs on your machine or enter a valid ODBC Connect String. If you have a DSN named "Marketing" you would enter:

   ```
   DSN=Marketing;UID=username;PWD=password
   ```

   Note: If there is no password, you must include a semicolon at the end:

   ```
   DSN=Marketing;UID=sa;PWD=;
   ```
3. Any valid SQL or ODBC command may be executed through this interface. The following example shows a connection to Microsoft SQLServer 2000, making a simple query against the sample Northwind database:

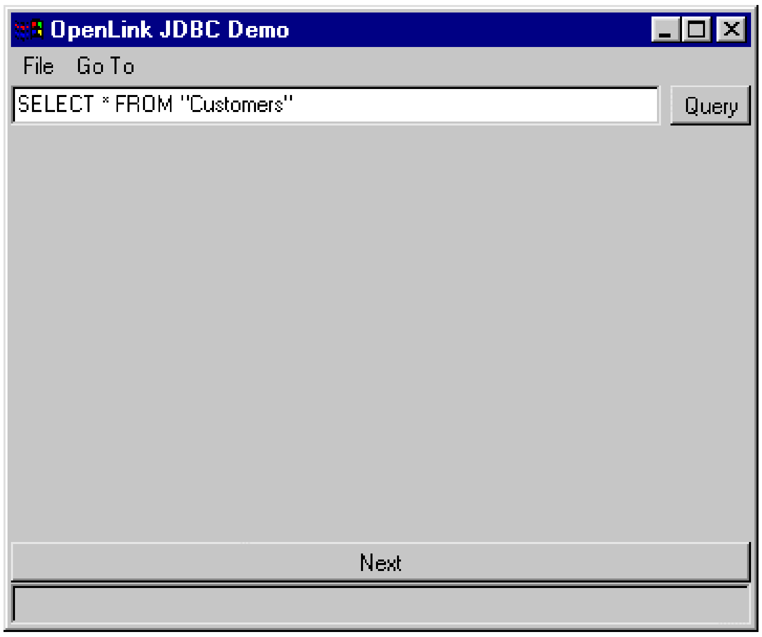
   ```
   [localhost:~] openlink% /Library/iodbc/bin/odbctest
   iODBC Demonstration program
   This program shows an interactive SQL processor
   Enter ODBC connect string (? shows list, or DSN=...): DSN=user_tthib_sql2k
   SQL>select au_lname, au_fname, state from authors where au_id < '333-33-3333'
   au_lname                                |au_fname            |state
   ----------------------------------------+--------------------+-----
   White                                   |Johnson             |CA
   Green                                   |Marjorie            |CA
   Carson                                  |Cheryl              |CA
   O'Leary                                 |Michael             |CA
   Straight                                |Dean                |CA
    5 row(s) fetched.
   SQL>quit
   Again (y/n) ? n
   Have a nice day.
   [localhost:~] openlink%
   ```

17.2.5 JDBC Sample Applications &Applets

17.2.5.1 JDBCDemo Java Application

1. Go to the OpenLink Data Access Drivers "Start Menu" program group and then follow the JDBC Samples menu path to the appropriate JDK, and then JDBCDemo. This will execute a DOS batch program that initialises the Java demo application.

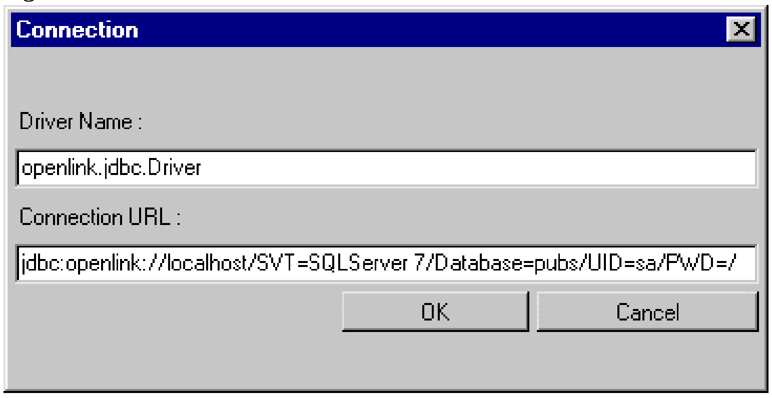
   Figure 15.13. JDBC Demo



SELECT * FROM "Customers"

2. Set the JDBC Driver Name and URL settings for the connection to your database. The "Driver Name" field identifies the jdbc Driver. The "Connection URL" field requires a valid JDBC URL.

   Figure 15.14. JDBC Demo

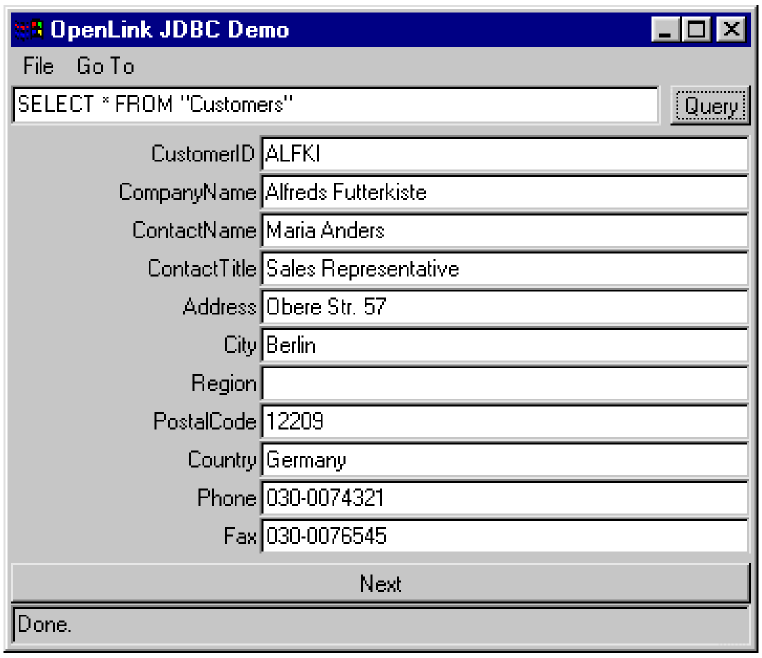


Driver Name : openlink.jdbc.Driver

Connection URL : jdbc:openlink://localhost/SVT=SQLServer 7/Database=pubs/UID=sa/PWD=/

3. Enter a valid SQL statement and then click the "Query" button, the example below uses a SQL statement requesting all records from the "Customers" table.

   Figure 15.15. JDBC Demo



SELECT * FROM "Customers"

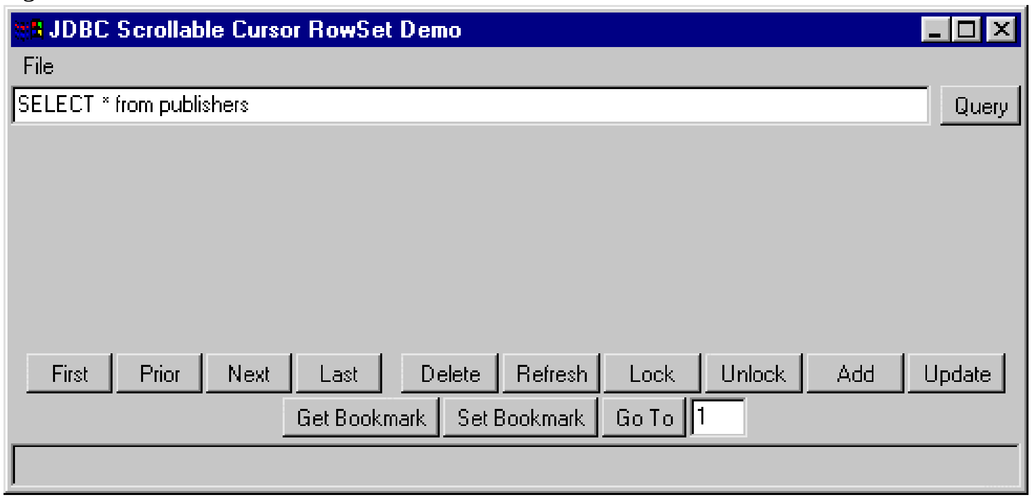
CustomerID	ALFKI
CompanyName	Alfreds Futterkiste
ContactName	Maria Anders
ContactTitle	Sales Representative
Address	Obere Str. 57
City	Berlin
Region	
PostalCode	12209
Country	Germany
Phone	030-0074321
Fax	030-0076545

Done.

17.2.5.2 Rowset Java Application

1. Go to the OpenLink Data Access Drivers "Start Menu" program group and then follow the JDBC Samples menu path to the appropriate JDK, and then Rowset Demo. This will execute a DOS batch program that initialises the Java demo application.

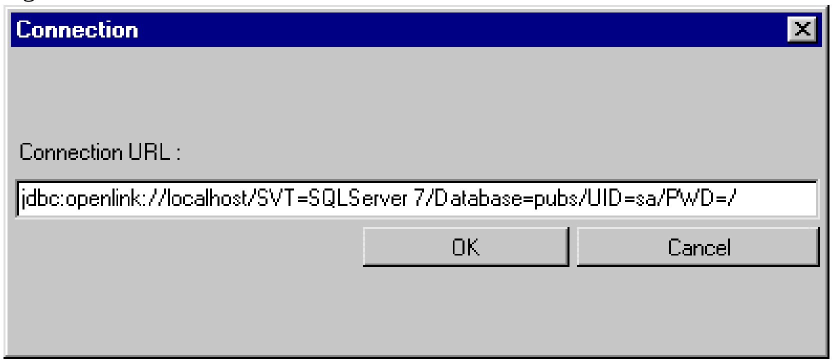
   Figure 15.16. Rowset Demo

   SELECT * from publishers

2. Set the JDBC URL settings for the connection to your database. The "Connection URL" field requires a valid JDBC URL.

   Figure 15.17. Rowset Demo

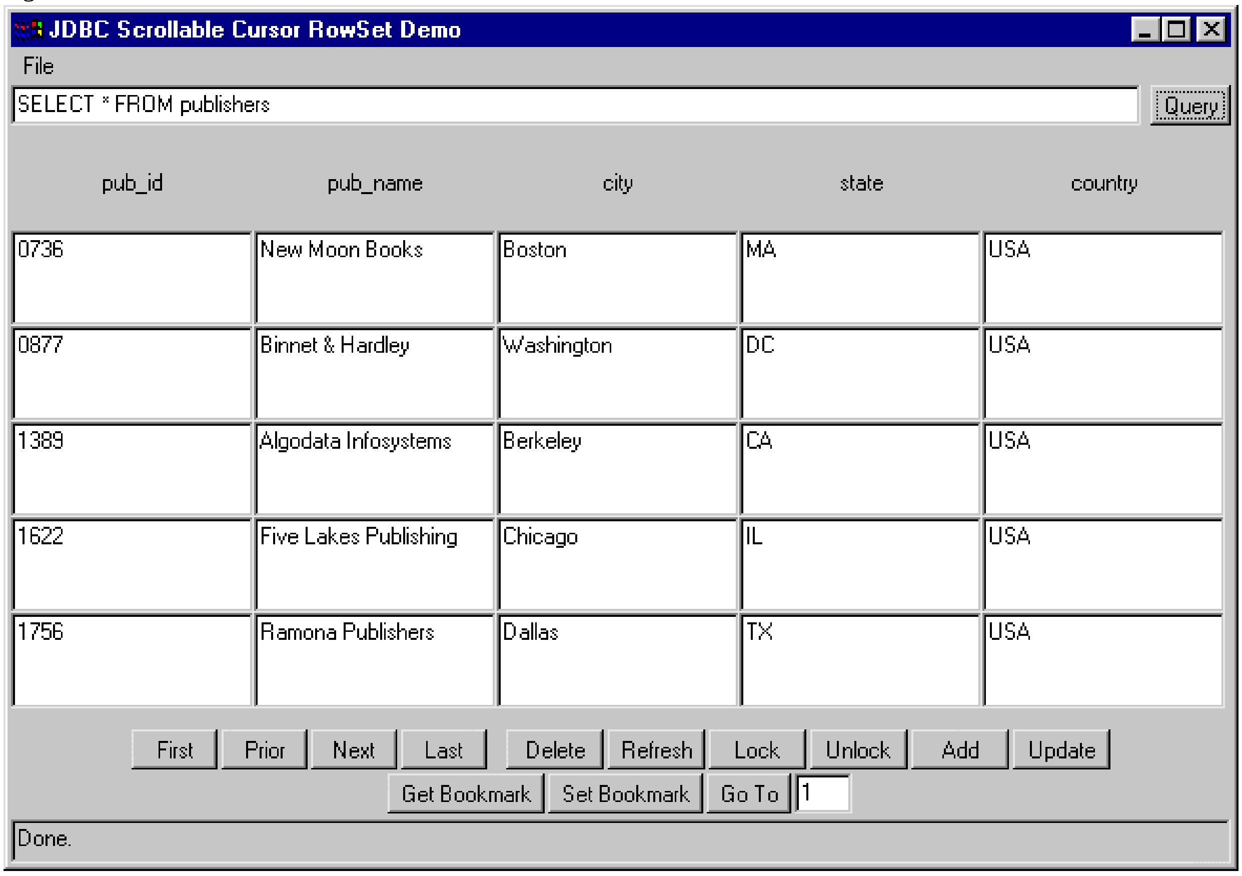
   

   Connection URL :

   jdbc:openlink://localhost/SVT=SQLServer 7/Database=pubs/UID=sa/PWD=/

3. Enter a valid SQL statement and then click the "Query" button. The first rowset of the resultset will be shown.

Figure 15.18. Rowset Demo



SELECT * FROM publishers

pub_id	pub_name	city	state	country
0736	New Moon Books	Boston	MA	USA
0877	Binnet & Hardley	Washington	DC	USA
1389	Algodata Infosystems	Berkeley	CA	USA
1622	Five Lakes Publishing	Chicago	IL	USA
1756	Ramona Publishers	Dallas	TX	USA

1

Done.

4. Click on the "Next" button to retrieve the next rowset, "Prior" to go back a rowset, "First" to move directly to the first rowset, and "Last" to move to the last rowset.

   "Delete", "Refresh", "Lock", "Unlock", "Add" and Update" work on the principle of a current row in the rowset. To indicate the current row to the applet, you must click on any column of a particular row and then press a key. The status panel will then display "Current Row = 2" or similar, indicating that the current row is set.

   Clicking "Delete" will attempt to delete the row from the database. Clicking on "Refresh" will now retrieve the latest values from the database to refresh that row. However this may not be successful dependent on the restrictions place on the resultset by the underlying database.

   "Lock" and "Unlock" will attempt to perform those operations on the current row, which again, may or may not be successful, dependent on the restrictions of the underlying database.

   "Add" will attempt to add the contents of the current row as a new row to the relevant tables forming the resultset. "Update" will attempt to update the modified contents of the current row to the database. Again the success of these two operations depend on the restrictions placed on the underlying database.

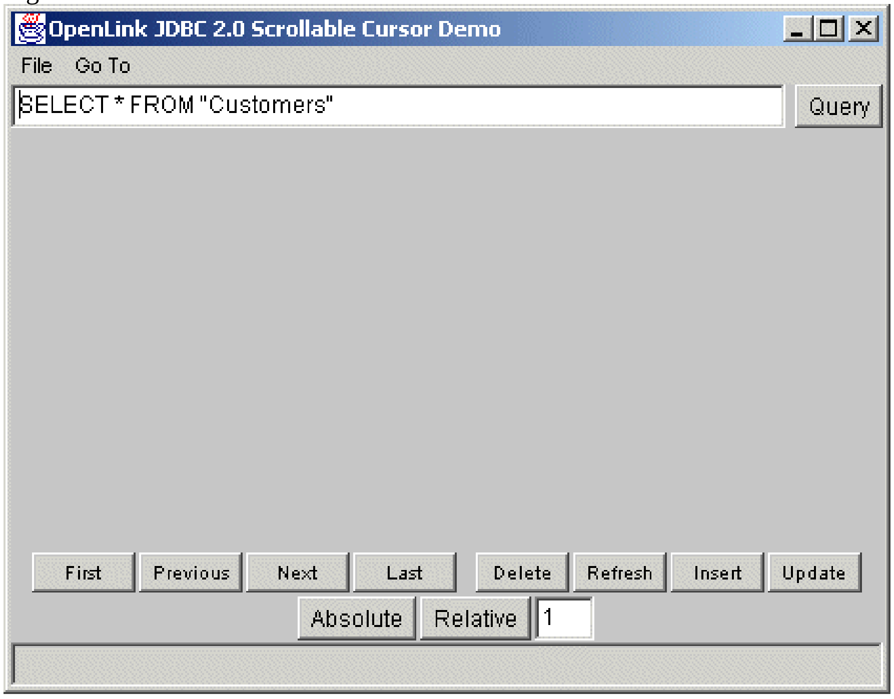
   "Get Bookmark" will attempt to retrieve a bookmark for the current rowset position, which "Set Bookmark" will attempt to return to.

   To move to a particular position in the resultset, enter that position into the edit box next to the "Goto" button and click the "Goto" button.

17.2.5.3 ScrollDemo2 Java Application

1. Go to the OpenLink Data Access Drivers "Start Menu" program group and then follow the JDBC Samples-->ScrollDemo2 (JDK1.2) menu path, this will execute a DOS batch program that initializes the Java demo application.

Figure 15.19.

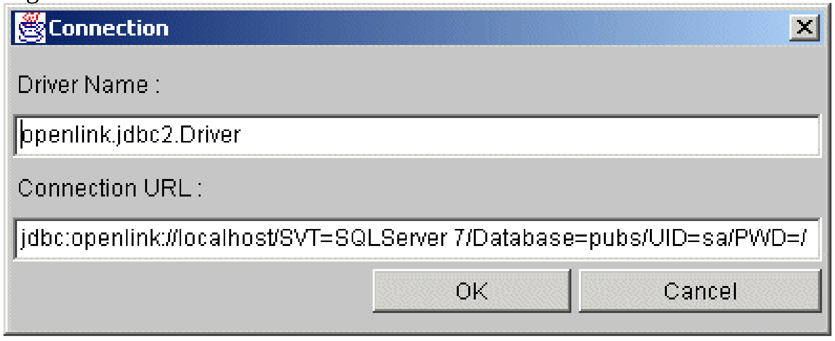


SELECT * FROM "Customers"

1

2. Set the JDBC Driver Name and URL settings for your connection to a database of your choice. The "Driver Name" field identifies the OpenLink Driver. If it is left blank, then it defaults to "openlink.jdbc2.Driver", which is the OpenLink Driver for JDBC 2.0. The "Connection URL" field requires a valid OpenLink JDBC URL.

Figure 15.20.

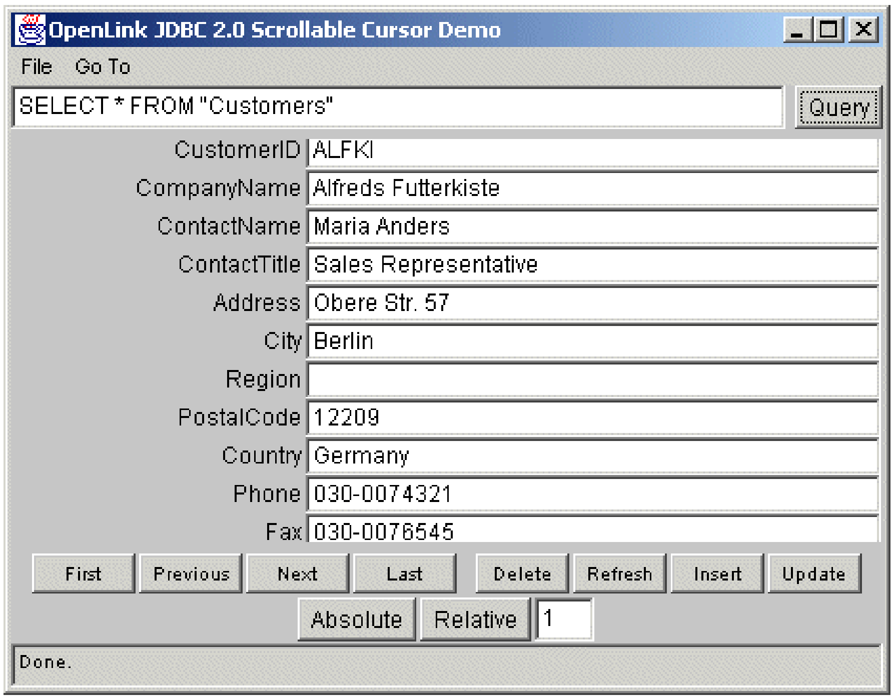


Driver Name : openlink.jdbc2.Driver

Connection URL : jdbc:openlink://localhost/SVT=SQLServer 7/Database=pubs/UID=sa/PWD=/

3. Enter a valid SQL statement and then click the "Query" button.

Figure 15.21.



SELECT * FROM "Customers"

CustomerID	ALFKI
CompanyName	Alfreds Futterkiste
ContactName	Maria Anders
ContactTitle	Sales Representative
Address	Obere Str. 57
City	Berlin
Region	
PostalCode	12209
Country	Germany
Phone	030-0074321
Fax	030-0076545

1

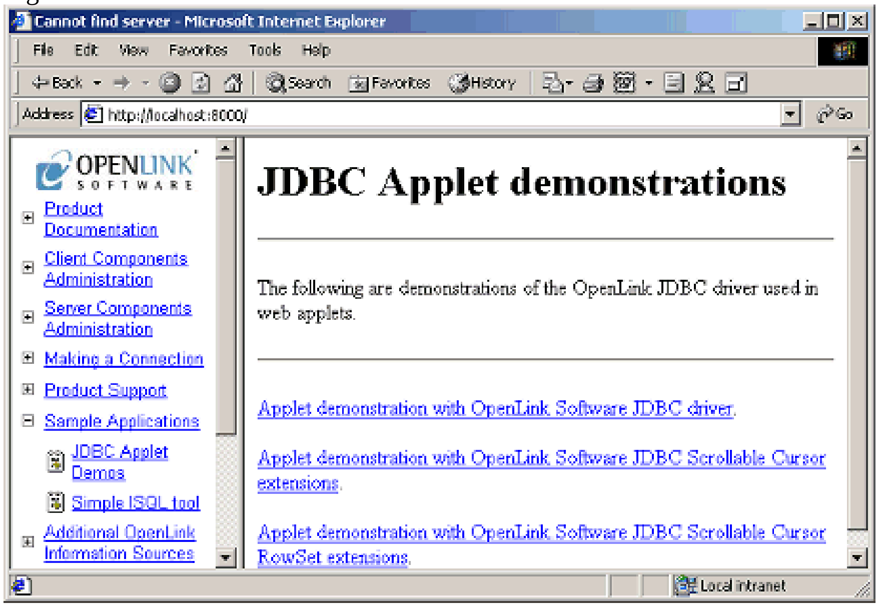
Done.

4. You can now use the navigational buttons to Scroll backwards and forwards, each of these navigational buttons highlights OpenLink's full implementation of the JDBC 2.0 Scrollable Cursors specifications.

17.2.5.4 ScrollDemo2 Java Applet

1. Start the OpenLink Admin Assistant and then follow the Sample Applications-->JDBC Applet Demos menu path.

Figure 15.22.



JDBC Applet demonstrations

The following are demonstrations of the OpenLink JDBC driver used in web applets.

Applet demonstration with OpenLink Software JDBC driver.

Applet demonstration with OpenLink Software JDBC Scrollable Cursor extensions.

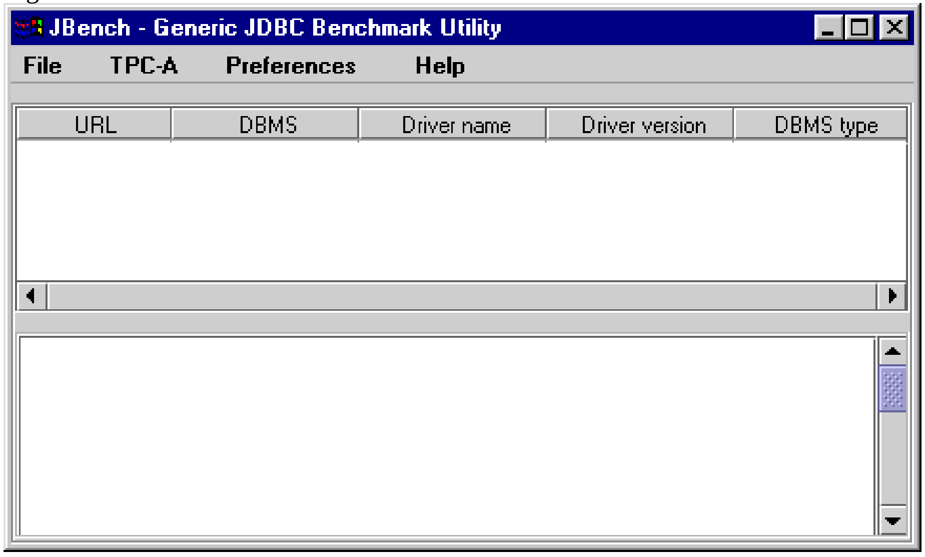
Applet demonstration with OpenLink Software JDBC Scrollable Cursor RowSet extensions.

2. Click on the "ScrollDemo2" hyperlink which initializes the ScrollDemo2 applet, if you do not have a Java 1.2 or Java 2.0 compliant browser you will not be able to run this Applet demo. The other way to experience this demo is to run the Application version which uses your operating systems Java Virtual Machine (JVM) instead of a JVM inherently linked to a Web Browser.

17.2.5.5 JBench Application

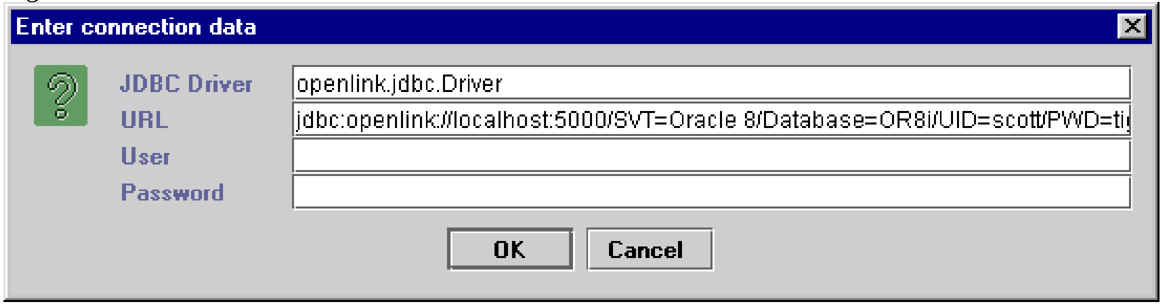
1. Go to the OpenLink Data Access Drivers "Start Menu" program group and then follow the JDBC Samples-->Jbench (for JDK1.1 1.2 or 1.3) menu path, depending on the JVM you have installed. This will execute a DOS batch program that initializes the JBench application.

Figure 15.23.



2. The follow the File-->Connect menu path to make your initial connection. You will need to identify your JDBC Driver (by providing appropriate Driver Name values in the JDBC Driver field) and then provide a valid JDBC URL for your specific JDBC Driver.

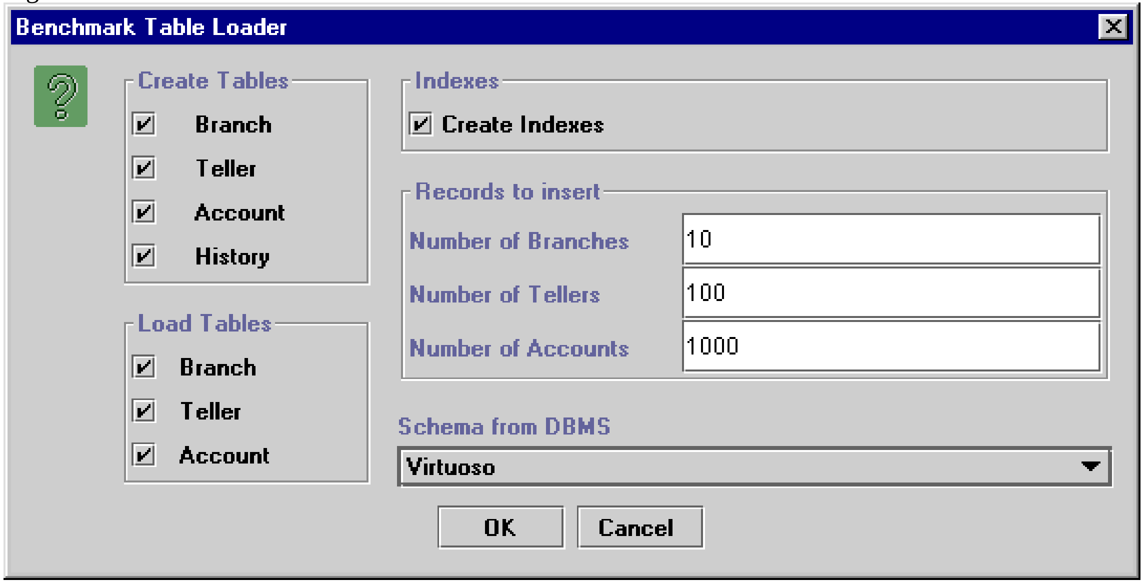
Figure 15.24.



JDBC Driver: openlink.jdbc.Driver
URL: jdbc:openlink://localhost:5000/SVT=Oracle 8/Database=OR8i/UID=scott/PWD=ti
User:
Password:

3. Select the open JDBC connection that you wish to test.
4. Follow the TPC-A-->Load Tables menu path to prepare your database for the TPC-A benchmark, select a database schema type that matches the database engine that you are benchmarking. If your database isn't listed ANSI should suffice (as long as this is an ANSI SQL compliant database).

Figure 15.25.



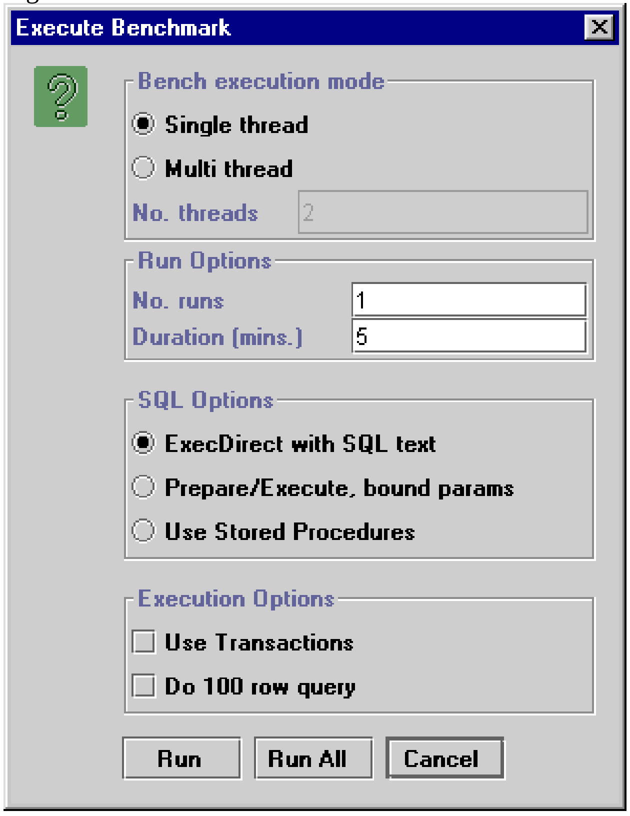
Create Tables: ✓ Branch, ✓ Teller, ✓ Account, ✓ History
Load Tables: ✓ Branch, ✓ Teller, ✓ Account
Indexes: ✓ Create Indexes
Records to insert:
Number of Branches: 10
Number of Tellers: 100
Number of Accounts: 1000
Schema from DBMS: Virtuoso

5. Follow the TPC-A-->Load Procedures menu path to Load the TPC-A stored procedures.

6. Now that all the benchmark data and stored procedures have been loaded into your database, follow the TPC-A-->Run Benchmark menu path and then configure your actual benchmark session parameters:

   The benchmark parameters fall into 4 categories, Bench execution mode, Run Options, SQL Options, and Execution Options.

   Figure 15.26.



Execute Benchmark

Bench execution mode

(● Single thread) ○ Multi thread

No. threads: 2

Run Options

No. runs: 1

Duration (mins.): 5

SQL Options

(● ExecDirect with SQL text) ○ Prepare/Execute, bound params ○ Use Stored Procedures

Execution Options

☐ Use Transactions

☐ Do 100 row query

17.2.5.5.1 Bench execution mode:

These setting allow you to configure the threads used for the benchmark.

Decide on a single or multiple threads test.

No. Threads. this is the number of concurrent threads to be used during the benchmark.

17.2.5.5.2 Run Options:

These setting allow you to configure the duration related aspects of this benchmark program.

No. runs. this controls how many iterations of the benchmarks you actually run (the default is 100 benchmark iterations).

Duration (mins.). this is the duration in minutes of each benchmark run.

17.2.5.5.3 SQL Options:

These setting allow you to configure how your benchmark's SQL instructions are actually handled.

ExecDirect with SQL Text. this means that no form of repetitive SQL execution optimization is being applied (SQL statements are prepared and executed repetitively)

Prepare/Execute Bound Params. this means that the Parameter Binding SQL execution optimization is being applied (SQL is prepared once but executed many times without the overhead of re-preparing statements prior to execution)

Use Stored Procedures. this means that the Stored Procedure SQL optimization is being applied (benchmark instructions are stored within database being benchmarked)

Execution Options: These setting allow you to configure the tone of your benchmark, for instance it could have Transaction scoping and a mix of record retrieval queries, or it could simply be input and update intensive with a minimal amount of record retrieval queries (the case when the 100 row query checkbox is unchecked a typical OLTP scenario)

Use Transactions. make the benchmark use transaction control (instructions are scoped to transaction blocks)

Do 100 row Query. perform a simulation of a 100 record retrieval as part of the benchmark activity.

17.2.5.5.4

1. Press Run or Run All to start your TPC-A benchmark. The Run All will sequence through all SQL and Execution combinations, so it will take much longer.
2. Follow the TPC-A-->Cleanup menu path to clean up your database so that you can then run other benchmarks.

18 Chapter 16. Bugs Fixed

Abstract
An overview of issues fixed with the Multi-Tier drivers.

Table 16.1. Bugs Fixed

Bug	Description
506	Oracle 8.0 on AIX fails InterDev and VB6 tests
734	Mac OS X iODBC installer overwrites existing symbolic links.
950	Windows Installer error `Severe -- VJ CLASSPATH too long to update`
1047	Version String for testcono should read `ODBC` not `UDBC`
1227	Modify Tracing tab description on Classic, to match Mac OS X
1486	sql2k_mv needs support for `translate character data` setting
1833	ODBCTest documentation is incomplete
1841	user-authentication fails, though UID/PWD are known correct
1968	oplrvc scripts are not documented
1989	Installer needs to shut down Broker and www_sv
2074	Progress 9.1C SQL Lite [Release 4.2] does not connect!
2199	DB2 Lite screen shots don't match actual interface & help text is recursive
2220	Current 4.2 SQL 92 Lite installer contains non-working Driver DLL
2229	privileges must be set to permit ADMIN group write access...
2330	iODBC HTTP Admin needs correct version graphic on About tab
2444	`rejected shutdown request from tthibodeau@tthibodeau` in a Broker-host CMD session
2455	Admin Assistant wizard and Mozilla browser
2685	prs91_sv.exe as of 4.2.2002-03-27 installer fails on some Windows boxes
2762	LingerTimeOut option needs to be documented
2831	data type `bit` returns `0.0` for every record, regardless of actual content (all Unix; Windows SQL Lite unavailable)
2863	MS Access "#deleted" errors when primary key is SQL_BIGINT
2880	Unix Lite needs updated filenames
2903	Postgres agent updated fail using the OLEDB rowset viewer program
2971	Rel 4.2 installer fails to populate CLASSPATH, no informative error when no license
3097	SQLDataSources() does not follow API spec
3168	MT 4.2 sql2k agent doesn't bring back BIGINT data (sql6 agent can)
3251	Win32 Broker not responding after 60 sql2k_mv connections
3254	[SQL Server]Syntax error converting DATETIME from character string in Sco ODBC SDK
3265	remove notation of `beta release` from iODBC description
3266	Installer must not reboot Macintosh without user confirmation
3272	iODBC 3.5.3 SQLDriverConnect() fails with System DSNs, unless Driver is also specified
3325	client connections to Oracle 9.2 on Mac OS X require new libclntsh.dylib.9.0
3328	Retrieval of Progress Decimals results in DB Server Crash and Driver General Error
3330	SQLSetConnectAttr or SQLSetConnectOption fails to set transaction isolation level
3366	TDS Lite drivers are not "describing" columns as NULLABLE
3423	TDS Lite Driver does not recognize 'option + f' special character on Mac Classic
3464	4.2 Lite driver doesn't bring back BYTE 800 Ingres data (Multi-tier does)
3483	PostgreSQL Lite license for Linux does not work
3487	MIN_FIELD_LEN not available in Progress Lite
3516	SQLGetConnectAttr (SQL_ATTR_CURRENT_CATALOG = 109) fails with `Option type out of range`
3533	DATETIME and TIMESTAMP columns do not work as expected in TDS Driver

19 Chapter 17. Technical Appendix

Abstract
Reference tables for the expert user.

Table of Contents

19.1 Rulebook Settings

19.1.1 [Request Broker]

Table 17.1.

Parameter	Default	Comment
BinaryDirectory		Absolute path to the bin directory which has the Broker program. Example: /home/openlink/openlink/bin
CommandLine		The command line parameters to use as a default. These parameters are also passed on to each program referenced in [Persistent Services]
Protocols	tcp	Name of protocol to use tcp/spx/DECnet
MaxAgents	0	if >0, maximum number of processes spawned by broker
MaxConnections	0	if >0, maximum number of connections from clients
HostNameResolver	Yes	Resolve IP addresses to hostnames Yes/No. Use Yes to give a more readable name in the logs. Set to No to avoid possible delays when frequently looking up unknown (new) IP numbers through a distant uncached DNS table. This parameter will have a direct affect on the type of match that is to be used for the StartupBy, ShutdownBy and ShutdownFrom parameters under the Security section.
LingerTimeout	0	if >0, time in seconds that a disconnected agent will linger ready for re-use before giving up, and terminating.

19.1.2 [Protocol TCP]

Table 17.2.

Parameter	Default	Comment
PingWatchdog	No	Send check-alive packets Yes/No
PingInterval	600	Number of seconds between each Ping
IPAddress	127.0.0.1	Overrules found interface address
PortLow	5000	Optional lower limit of port number to be used in allocation by the Broker for communication with an Agent.
PortHigh	60000	Optional upper limit of port range usage by Broker.
Listen	5000	Port that the client should contact the broker on.

19.1.3 [Protocol SPX]

This section is only relevant, and required when using the SPX protocol.

Table 17.3.

Parameter	Default	Comment
SAPBroadcast	Yes	Broadcast SAP registration packets
SAPInterval	100	Number of seconds between each SAP broadcast
SAPServiceType	0x321F	Type of the SAP service
SAPServiceName	OPENLINK	name of the SAP service
HostFile	HOSTS.IPX	For IPX/SPX name to address translation

19.1.4 [Protocol DECnet]

This section is only relevant, and required when using the DECnet protocol.

Table 17.4.

Parameter	Default	Comment
ObjectName	OPLRQB	Name of the DECnet server object

19.1.5 [Communications]

Table 17.5.

Parameter	Default	Comment
SendSize	16000	Send buffer size
ReceiveSize	4096	Receive buffer size
DataEncryption	No	Crypting of outgoing packets
BrokerTimeout	30	Timeout for utilities to contact the broker
ReceiveTimeout	120	Maximum time allowed for OpenLink Service Agents (JDBC_SV, ODBC_MV, PROXY_SV) to instigate a session with database agents (ora7_sv, inf7_mv etc.)
RetryTimeout	5	Initial retry interval in seconds. Doubles on failure to a maximum of 30 seconds

19.1.6 [ZeroConf]

Table 17.6.

Parameter	Default	Comment
<ServiceName>	<Connection String Attributes>	User defined ZeroConfig entry

19.1.7 See Also:

The Zero Configuration Chapter

19.1.8 [Security]

Table 17.7.

Parameter	Default	Comment
StartupBy	.*	Users who can startup oplrqb
ShutdownBy	.*	Users who can shutdown oplrqb
ShutdownFrom	localhost.*,127\.0\.0\.1	Hostnames that can shutdown
ValidUidRange	0, 50000	Valid range for OpSysLogin
TraceRulebook		Write merged rulebook (debug) Example: /tmp/debug.book
IncludeRulebook		Name of file in current directory holding additional rulebook sections. (Example: generic.book). There may be more than one occurance of IncludeRulebook so multiple books can be included. The sections specified in the included files cannot overrule the sections contained in the main broker rulebook. Only unique sections are included.

The StartupBy, ShutdownBy and ShutdownFrom parameters use full Regular Expressions. Separate multiple expressions with a comma.

Table 17.8. Regular Expression Examples

Expression	Interpretation	Comment
.*	Any character, any number of times	Any user or location.
localhost.*	localhost followed by zero or more characters	Will match: localhost, localhost2, localhost.company.com
127\.0\.0\.1	exact IP 127.0.0.1	Current machine when HostNameResolver=No
.*\.company\.com	Any text before .company.com	Used to match any sub domain of company.com. For example: sales.company.com, john.company.com

19.1.9 [generic_agentname]

Each Database Agent will have a section that is typically called "generic_" followed by a database label. For example: [generic_virt], [generic_odbc], [generic_db2].

Table 17.9.

Parameter	Default	Comment
Program		name of the executable to run
Environment		specify Environment section to use
CommandLine		commandline options (optional)
ServerOptions		force dbms client specific options (optional)
Directory		where to chdir before starting the server
Database		force name of the database (optional)
ConnectOptions		force dbms specific connect options (optional)
UserID		force userid (optional)
Password		force password (optional)
ReadOnly	No	force read-only mode (optional)
OpsysLogin	No	when set to Yes, the Request Broker will do an operating system login before spawning. This implies ReUse = ifsame user. (optional)
NeedsCLI		(VMS only) Use CLI to spawn the agent
ReUse	never	Defines how an agent will be re-used. See below for details of the ReUse Parameter.

For the CommandLine parameter, you can specify a selection of the following options:

+noautocommit
: This means that all connections routed (by the mapping rules) through this agent section will have autocommit behaviour turned off. This is useful if your client-side application relies on manual commits of its transactions; you can define a mapping rule to match that application and add +noautocommit while other applications use a different agent section.

+maxrows
: This defines the maximum number of rows to fetch from any query.

+initsql
: Specifies a file with a set of SQL statements to execute immediately each connection is established. For example, this might be useful to set transaction isolation levels, if your application assumes them to be set a specific way already.

+jetfix
: This enables various workarounds for operation with the Microsoft Jet Engine, e.g. through Access or MS Query. Particularly, the mapping of datatypes may be changed for greater accuracy using these applications.

+norowsetlimit
: This disables any rowset-size limit; it is useful in cursor operations on large tables.

19.1.9.1 Agent ReUse parameter

Table 17.10.

ReUse Value	When is agent re-used
never	never reuse this instance (default)
always	always try to reuse this instance
upto <n>	allow at most n connection to the same instance
ifreadonly	allow only the read-only requests
ifsame database	allow requests that alias to the same db.
ifsame process	same calling process id (pid)
ifsame options	same connect options specified
ifsame application	if originated from same appl. alias
ifsame user	if request for same username
ifsame machine	group connections from same machine
ifsame opsys	group connections with compatible operating sys.

19.1.10 [Domain Aliases]

This section is used to change a domain name specified in the connect string of a DNS with an internal alias. This alias is used in the first colon delimited field of a mapping rule.

This example will map two different Progess domains to one agent. An alternative is shown for mapping three different Oracle types to the one Oracle agent.

```
[Domain Aliases]
^Progress 90A$|^Progress 90B$ = pro90b
Oracle 9i                     = ora90
^Oracle 9.0$                  = ora90
^Oracle 9.0.x$                = ora90
```

19.1.11 [Database Aliases]

This section will replace a database name specified in the connect string of a DNS with an internal alias. This alias is used in the second colon delimited field of a mapping rule.

This example looks for a substring 'demo' and will replace with an alias of 'demo'. Thus anydemo, demo, demo123 are all matched, and converted to demo.

```
        [Database Aliases]
        demo = demo
```

19.1.12 [User Aliases]

This section will replace a user name specified in the connect string of a DNS with an internal alias. This alias is used in the third colon delimited field of a mapping rule.

The example below shows how certain users or an empty user are handled. In this case they are rejected.

```
        [User Aliases]
        scott|system = insecure
        ^$           = blank
        [Mapping Rules]
        *:*:blank:*:*:*:rw     = reject You should specify a username
        *:*:insecure:*:*:*:rw  = reject The user is not allowed
```

19.1.13 [Opsys Aliases]

This section will replace an operating system indentifier with an internal alias. This alias is used in the fourth colon delimited field of a mapping rule.

This example will map anything containing the substring 'java' to an alias of 'java'. Two variations of windows are given an alias of 'msdos'. Everything else will be matched to .* so it is mapped to the alias 'other'.

```
        [Opsys Aliases]
        java        = java
        win32|msdos = msdos
        .*          = other
```

19.1.14 [Machine Aliases]

This section will replace a machine name with an internal alias. This alias is used in the fifth colon delimited field of a mapping rule.

This example will map two different machine names to one of 'adminpc'. Also anything containing the word 'sales' such as mysales, sales, sales2 is then mapped to 'sales' alias.

```
        [Machine Aliases]
        fredspc|johnspc    = adminpc
        sales              = sales
```

19.1.15 [Application Aliases]

This section will replace the application name with an internal alias. This alias is used in the sixth colon delimited field of a mapping rule.

This example would match MSACCESS (a program requiring the Jet option), and map it to an alias of jet. The second alias mapping would match various Office applications and convert them to a single alias.

```
        [Application Aliases]
        MSACCESS           = jet
        MSQRY.*|EXCEL|WORD = msoffice
```

19.1.16 [Mapping Rules]

This section is used to determine which agent shall handle the incoming request. The mapping rules are checked once all the alias mappings have been performed. Each mapping rule is tried from top to bottom until a match with the current parameters has been found. There is no regular expression or glob handling in the mapping rules. The 7 colon delimited mapping parameters must each match up exactly. There is a special mapping rule of '*' that denotes a dont care parameter. Do not confuse this special '*' with the regular expression '*', or glob '*'.It is not possible to use the '*' with any other text such as 'demo*'.

On the right side of the '=' is either an accept, or reject statement. The accept statment has the word 'accept' followed by the section name that identifies the agent. A reject statement has the word 'reject' followed by a text string that is the error message reported to the client.

This is an example mapping section.

```
[Mapping Rules]
;*:*:blank:*:*:*:rw     = reject You should specify a username and password
*:*:Admin:msdos:*:jet:* = reject Admin user account is not registered
sql2000:*:*:*:*:*:*     = accept generic_sql2000
ora81:*:*:*:*:jet:*     = reject The Oracle 8 Database Agent is not configured for jet
*:*:*:java:*:*:*        = accept jodbc_client
```

Here is a snippet of the debug output showing how a request is shown to be matched.

```
request: domain=Oracle 8.1 database=db serveropts=
connectopts= user=scott opsys=win32 readonly=0
application=ODBCAD32 processid=520
solve mapping: ora8sv:db:scott:win32:MASTERSRVR:ODBCAD32:rw
using mapping: ora8sv:*:*:*:*:*:*
```

19.2 Error Codes

When a Broker or agent is terminated, the exit code can be seen in the log file (for example oplrqb.log) or in the foreground window.

The exits codes are shown below:

Table 17.11. Log Levels

Exit Code	Category	Comment
0	NORMAL	Normal exit with success
1	WARNING	some kind of failure at startup/shutdown, typically shown with a reason for this error
>1	UNKNOWN	When the exit code is not in the range of 0-15 then it is typically an uninitialised exit value taken from the stack. This can occur when an error in some database code is returned to the OpenLink code (Broker).

The agent or broker may be terminated with a signal:

Table 17.12. Reported Signals

Signal	Name	Category	Comment
2	SIGINT	NORMAL	user pressed CTRL-C on oplrqb -fd
3	SIGQUIT	NORMAL	user pressed CTRL-\ on oplrqb -fd
4	SIGILL	FAULT	illegal instruction resulting in a program crash
7	SIGBUS	FAULT	misaligned mem read/write resulting in a program crash
8	SIGFPE	FAULT	divide by 0 or other arithmetical error
9	SIGKILL	NORMAL	sent by the broker to kill an agent which has ignored multiple SIGTERMs
11	SIGSEGV	FAULT	buffer/stack overrun resulting in a program crash
13	SIGPIPE	NORMAL	other side of a pipe was killed while this side tried to write data to it
15	SIGTERM	NORMAL	sent by the broker to an agent to ask it to terminate

On the Windows platform, severe errors are reported in the following way:

```
pid ... died with status code 0x....
```

The hex number may be looked up in the winnt.h file. All errors reported in this way are considered severe, and should be reported to OpenLink Support.

19.3 Broker Log Levels

The Request Broker reports various information and messages to the foreground window, or the log file if selected. The amount of detail that is shown is controlled by the loglevel parameter.

Define the required loglevel on the command line call. For example to get loglevel 7, use:

```
oplrqb +loglevel 7
```

or

```
oplrqb -l 7
```

Alternatively set the loglevel default in the oplrqb.ini file.

```
[Request Broker]
CommandLine = +loglevel 7
```

When a loglevel is specified, the messages from priority 0 up to that level are shown. Thus using +loglevel 5 will produce messages from levels 0, 1, 2, 3, 4 and 5. Specifying +loglevel 9 is the same as +loglevel 7 as levels 8 and 9 are not defined.

Typical loglevels are 7 for troubleshooting, 3,4 or 5 for normal use, and 0 or 1 for a limited report. The if a loglevel is not specified, then a default of 5 is used.

The level of detailed for each log level is as follows:

Table 17.13. Log Levels

Level	Category	Comment
0	EMERG	System is unusable
1	ALERT	Action must be taken immediately
2	CRIT	Critical conditions
3	ERR	Error conditions
4	WARNING	Warning conditions
5	NOTICE	Normal but signification condition
6	INFO	Informational
7	DEBUG	Debug-level messages

19.4 ODBC to Jet Data Type Mapping

19.4.1 Overview

When linking to a remote table, Microsoft Jet calls the ODBC SQLColumns function to gather information on the fields in the table. The information returned by SQLColumns determines how the data type of each field will be mapped to a Microsoft Jet data type.

Jet uses the *fSqlType*, *lPrecision*, and *wScale* values to determine an appropriate local data type, on a field-by-field basis, and uses that data type to represent the remote data in a linked table. Each time Microsoft Jet executes an action or parameter query against the remote data source, the information stored in the linked table is used to ensure that ODBC calls are made with a valid ODBC data type.

The Jet Engine's mapping of the ODBC SQL_DECIMAL and SQL_NUMERIC types is detailed below. The mapping of these column types is problematic when the precision of a column is greater than 15. Under these circumstances Jet maps the column to the Jet Text type. This is done so as not to lose precision, since the maximum precision of the Jet numeric types is 15. An unfortunate consequence of mapping to the Text type is that the data is no longer recognised as being numeric.

To overcome this limitation, the Jetfix switch (+jetfix) changes the ODBC type returned by SQLColumns for native database column types which would normally be reported as SQL_DECIMAL or SQL_NUMERIC. These might, for example, be MONEY, DECIMAL or NUMERIC columns in the remote table. With Jetfix enabled, these columns are typically reported as SQL_DOUBLE; which Jet always maps to Number (Double). As a result, Jet recognises the data as numeric, albeit with a possible loss of precision, depending on the actual data values.

19.4.2 ODBC to Jet Data Type Mapping

Table 17.14. ODBC to Jet Data Type Mapping performed by Jet Engine

ODBC Data Type			Jet Data Type	Notes
Type	**Precision p**	**Scale s**		
	p <= 4	s = 0	Number (Integer)	
	p <= 9	s = 0	Number (Long Integer)	
SQL_DECIMAL	p <= 15	0 <= s <=4	Number (Double)	
SQL_NUMERIC	p <= 15?	s > 4	Number (Double)	"ODBC to Jet Data Type Mapping" document unclear here.
	p = 19	s = 4	Currency	MS SQL Server only
	p = 10	s = 4	Currency	MS SQL Server only
	p > 15	s irrelevant	Text	Not documented. Based on testing.

The Jet Data Type is a property setting available in the table Design view.

19.4.3 Jet Data Type Ranges

Table 17.15. Jet Data Type Ranges

Jet Data Type	Range
Integer	32,768 32,767
Long Integer	2,147,483,648 2,147,483,647
Double	-1.8E+308 -1.8E-308, 1.8E-308 1.8E+308
Currency	922,337,203,685,477.5808 922,337,203,685,477.5807

19.4.4 References

See the following articles from Microsoft:

- Knowledge base article KB214854, "Improved ODBC DataType Mappings with Jet 4.0" on the Microsoft Support Website .
- See Chapter 9 "Developing Client/Server Application" of Microsoft Jet Database Engine Programmer's Guide, Second Edition. Redmond, WA: Microsoft Press, 1997.

19.4.5 Informix Jet Support

19.4.5.1 Informix Jetfix Testing

To verify the effect of the +jetfix switch, two tables were created and then linked to from Access, once with +jetfix enabled, and once with +jetfix disabled. The first table contained five MONEY columns of various precisions and scales. The second table was identical except that it used DECIMAL instead of MONEY columns. The resulting Jet data type mappings as shown in the 'Table Design' view are detailed below:

Table 17.16. Informix ODBC to Jet Data Type Mapping

Informix table definition:	**Jet Type Mappings**	
create table CBTEST (<column defs>)	**Jetfix off**	**Jetfix on**
col1 {MONEY| DECIMAL} (14, 4)	Number (Double)	Number (Double)
col2 {MONEY| DECIMAL} (14, 5)	Number (Double)	Number (Double)

Informix table definition:	**Jet Type Mappings**	
create table CBTEST (<column defs>) col3 {MONEY| DECIMAL} (16, 0)	Text	Number (Double)
col4 {MONEY| DECIMAL} (16, 4)	Text	Number (Double)
col5 {MONEY| DECIMAL} (16, 5)	Text	Number (Double)

19.4.5.2 OpenLink Informix Agent Numeric Data type Mappings

The following table details how the Informix agent fetches numeric data from Informix. The SQL type reported for a SELECT column varies depending on the native Informix type of the column, and its precision and scale. The OpenLink C type used to fetch the data is also shown, together with the effect of the JetFix command line switch.

Changes to the type mappings required to implement the +jetfix option are shown in `italics`.

Table 17.17. Informix Agent Numeric Data type Mappings

Informix Type	**Scale (s)**	Decimal Precision (p) Range (r)	**JetFix**	**SQL Type**	OpenLink Fetch Type CTYPE_xxx
FLOAT [(n)] {1} Synonyms: DOUBLE PRECISION	NA	p <= 16 r = 1.7E308 (15 digits precision)	NA	SQL_DOUBLE	FLT64
DECIMAL DECIMAL (p) DECIMAL (p,s) Synonyms: NUMERIC	 NA NA	p = 16 p <= 32 p <= 32 r = 1E+124 1E-130 ..	OFF `ON`	SQL_DECIMAL `SQL_DOUBLE`	VCHR `FLT64` {2}
INTEGER Synonymns: INT	s = 0	p <= 9 r = -2,147,483,648 to +2,147,483,647	NA	SQL_INTEGER	SGN32
MONEY MONEY(p) MONEY (p,s)	 s = 2 s = 2	p = 16 p <= 32 p <=32 r = 1E(p-s) - 1E(-s)	OFF `ON`	SQL_DECIMAL `SQL_DOUBLE`	VCHR `FLT64` {2}
SMALLFLOAT Synonyms: REAL	NA	r =3.4E38	NA	SQL_REAL	FLT32
SMALLINT	s = 0	p = 5 r = -32,768 to +32,767	NA	SQL_SMALLINT	SGN16

- {1} Informix SQL ignores the specified float precision.
- {2} Possible loss of precision.

19.4.5.3 Precision of OpenLink C Types

The choice of OpenLink C type used to fetch a numeric value is determined by the maximum precision supported by the type. If a numeric type exists which can represent the data to be fetched, without loss of precision, then this is used as the preferred fetch type. If a numeric type capable of supporting the required precision does not exist, then the data is fetched in character form, unless the JetFix switch is on. In this case, the data is fetched using the CTYPE_FLT64 fetch type. This type has sufficient range to store the maximum numeric values supported by Informix, but its use may result in a loss of precision, if the precision of the data to be fetched exceeds 15 decimal digits.

Table 17.18. Precision of OpenLink C Types

OpenLink C Type CTYPE_xxx	Equivalent built in C type (32bit processor)	Range	Can hold all numbers of precision <= p decimal digits
SGN8	signed char	-128 127	p = 2
SGN16	short	-32768 32767	p = 4
FLT32	float	3.4E38 (6 digits precision)	p = 6
SGN32	long	-2,147,483,648 2,147,483,647	p = 9
FLT64	double	1.7 E308 (15 digits precision)	p = 15

19.4.6 Oracle Jet Support

19.4.6.1 OpenLink Oracle Agent Numeric Data type Mappings

The following table details how the Oracle agent fetches numeric data from Oracle. The SQL type reported for a SELECT column varies depending on the native Oracle type of the column, and its precision and scale. The OpenLink C type used to fetch the data is also shown, together with the effect of the JetFix command line switch.

Changes to the type mappings are shown in the table as `italics`.

Table 17.19. Informix ODBC to Jet Data Type Mapping

Oracle Type	Scale (s)	Decimal Precision (p)	JetFix	SQL Type	OpenLink Fetch Type CTYPE_xxx
FLOAT (b)	NA	p <= 15 (b <= 50) {1}	NA	SQL_DOUBLE	FLT64
		15 < p < 38 (50 < b < 126)	ON	SQL_DOUBLE	FLT64 {2}
			OFF	SQL_DECIMAL	VCHR16
FLOAT {3} , NUMBER {4}	NA	p = 38	ON	SQL_DOUBLE	FLT64 {2}
			OFF	SQL_DECIMAL	VCHR16
NUMBER (p,s)	s <> 0	p <= 15	ON	SQL_DOUBLE	FLT64
			OFF	SQL_DECIMAL	VCHR16
		p > 15	ON	SQL_DECIMAL `SQL_DOUBLE`	VCHR16 `FLT64` {2}?
			OFF	SQL_DECIMAL	VCHR16
Calculated field	s = 0	p = 0 {5}	ON	SQL_DOUBLE	FLT64
			OFF	SQL_DECIMAL	VCHR16
NUMBER (p)	s = 0	0 < p <= 2	NA	SQL_TINYINT	SGN8
		2 < p <= 4	NA	SQL_SMALLINT	SGN16
		4 < p <= 9	NA	SQL_INTEGER	SGN32
		9 < p <= 15	ON	SQL_DOUBLE	VCHR16 `FLT64`
			OFF	SQL_DOUBLE `SQL_DECIMAL`	VCHR16

Decimal Precision (p)	JetFix	SQL Type	OpenLink Fetch Type CTYPE_xxx
15 < p <= 38 {6}	ON	SQL_DOUBLE	FLT64 {2}
	OFF	SQL_DECIMAL	VCHR16

Oracle Type	Scale (s)

{Notes}:

- FLOAT(b) specifies a floating point number with binary precision b. The precision can range from 1 to 126. To convert from binary to decimal precision, multiply b by 0.30103.
- Possible loss of precision
- FLOAT specifies a floating point number with decimal precision 38, or a binary precision of 126.
- NUMBER specifies a floating point number with decimal precision = 38.
- Oracle appears to return s=0, p=0 for a calculated field
- Oracle permits a maximum precision of 38.

19.4.6.2 Precision of OpenLink C Types

The choice of OpenLink C type used to fetch a numeric value is determined by the maximum precision supported by the type. If a numeric type exists which can represent the data to be fetched, without loss of precision, then this is used as the preferred fetch type. If a numeric type capable of supporting the required precision does not exist, then the data is fetched in character form, unless the JetFix switch is on. In this case, the data is fetched using the CTYPE_FLT64 fetch type. This type has sufficient range to store the maximum numeric values supported by Oracle, but its use may result in a loss of precision, if the precision of the data to be fetched exceeds 15 decimal digits.

Table 17.20. Precision of OpenLink Oracle C Types

OpenLink C Type CTYPE_xxx	Equivalent built in C type (32 bit processor)	Range	Can hold all numbers of precision <= p decimal digits
SGN8	signed char	-128 127	p = 2
SGN16	short	-32768 32767	p = 4
SGN32	long	-2,147,483,648 2,147,483,647	p = 9
FLT64	double	1.7 E308 (15 digits precision)	p = 15

19.5 SQL Server 2000 Connection Options

19.5.1 Connection Option Parameters

When configuring an OpenLink SQL Server 2000 multi-tier Agent (sql2k_[sm]v[.exe] only and not sql_[sm]v[.exe]), the following keywords and values may be entered in the *Connection Options* section of the entry for the agent in the Request Broker configuration file (oplrqb.ini). All are optional.

Table 17.21.

Key	Keyword	Value(s)	Description
-A	ADDRESS	Network address of the server which is running an instance of SQL Server.	Address is usually the network name of the server, but can be other names such as a pipe, or a TCP/IP port and socket address.
-B	FALLBACK	Yes (default) No Applies to SQL Server 6.5 only.	When yes, instructs the driver to attempt connection to a fallback server if connection to a primary server fails. When no, no attempt at a fallback connection is made. This option applies only to standby servers. It does not apply to a virtual server in a cluster / failover configuration.
-C	AUTOTRANSLATE	Yes (default) No	When yes, ANSI character strings sent between the client and server are translated by converting through Unicode to minimize problems in matching extended characters

Key	Keyword	Value(s)	Description
			between the code pages on the client and the server.
-D	DATABASE	Database Name	The name of the database which will be the default database for the connection. Note however, that for OpenLink Agents and Single-tier Drivers, this value is normally entered into a separate control on the Configuration module. If this control contains a name, entering this option as a connection option also could lead to the attempt to establish a connection being rejected.
-E	QUERYLOGFILE	Full path, name and extension of the log file.	This option applies only when Option O (QUERYLOG_ON) is set to *yes*. This is the file to which data for long-running queries is saved.
-F	ATTACHDBFILENAME	Name of the primary file of an attachable database. Include the full path and escape any \ characters if using a C character string variable. Example: c:\\MyFolder\\MyDB.mdf	This database is attached and becomes the default database for the connection. You must also specify the database name in the parameter, DATABASE. However, see above for the Database option, -D. If the database was previously attached, SQL Server will not reattach it; it will use the attached database as the default for the connection.
-G	QUERYLOGTIME	Numeric character string	Applies only when Option O (QUERYLOG_ON) is set to *yes*. This specifies the threshold (in milliseconds) for logging long-running queries. Any query that does not get a response within this time is written to the long-running query log file (which is defined by the Option, -E, QUERYLOGFILE).
-H	SERVER	Server Name	Name of a server which is running an instance of SQL Server on the network. By default, this is *local*.
-L	LANGUAGE	SQL Server language name.	SQL Server language name. SQL Server can store messages for multiple languages in sysmessages. If connecting to a SQL Server with multiple languages, Language specifies which set of messages are used for the connection.
-N	NETWORK	Name of a network library dynamic-link library. E.g. dbnmpntw	The name need not include the path and must not include the .dll file name extension.
-O	QUERYLOG_ON	Yes No (default)	Enables or disables logging long-running query data on the connection.
-Q	QUOTEDID	Yes (default) No	When yes, ,the SQL-92 rules regarding the use of quotation marks in SQL statements are applied. When no, the legacy Transact-SQL rules apply.
-R	REGIONAL	Yes No (default)	When yes, client settings are used when converting currency, date, and time data to character data. When no, ODBC standard strings are used to represent currency, date, and time data that is converted to string data.
-S	STATSLOG_ON	Yes No (default)	When set to *yes*, SQL Server ODBC driver performance data is captured.

Key	Keyword	Value(s)	Description
-T	TRUSTEDCONNECTION	Yes No (default)	When yes, Windows Authentication Mode is used for login validation. I.e. no User ID or Password need be supplied. When no, a User ID and password must be supplied.
-U	USEPROCFORPREPARE	0 (default) 1 2 Applies to SQL Server 6.5 only.	When 0, the SQL Server ODBC driver does not create temporary stored procedures for SQLPrepare. When 1, instructs the SQL Server ODBC driver to create temporary stored procedures when statements are prepared with SQLPrepare. The temporary stored procedures are not dropped until the connection is broken. When 2, the SQL Server ODBC driver creates temporary stored procedures for SQLPrepare, but only one procedure is created per statement handle and the procedure is dropped when the statement handle becomes invalid or a new SQL statement is prepared.
-W	ANSINPW	Yes (default) No	When yes, the driver uses ANSI-defined behaviors for handling NULL comparisons, character data padding, warnings, and NULL concatenation. When no, ANSI defined behaviors are not exposed.

19.5.2 Entry Format

19.5.2.1 ODBC Administrator

Options should be entered into the ConnectOptions control in the format:

```
-<OptionLetter1> value1 -<OptionLetter2> value2 etc
e.g.
-C no  L English
```

See the table above for a list of available option letters.

19.5.2.2 Oplrqb.ini Settings

Connection Options are entered following the *ConnectOptions* keyword. Options may be entered in the same format as in the ODBC Configuration dialog, using key letters as above for example:

```
ConnectOptions = {-c no  l English}
```

Surrounding braces ({ and }) may be used but are not necessary.

Alternately, the full keywords may be used. In this case, they should be entered in the format:

```
ConnectOptions = {Option1=value1;Option2=value2;etc}
```

Semicolons should be used to separate items in the list. Surrounding braces ({ and }) may be used but are not necessary. If surrounding braces are used, any text outside the braces will be discarded. A trailing semicolon is not necessary and ought not to be used, but should not cause fatal errors if it is used.

19.5.2.3 General

Items may be entered into the list in any order. They are not case-sensitive, so upper case or lower case or a mixture of both can be used (though this may depend on the SQL Server itself).

If no option key letters or keys are used, the Agent will assume that the complete ConnectOptions entry is the name of the server running the SQL Server database i.e. it will assume it to be a value for the keyword *SERVER*. See the notes for this keyword in the table above.

19.5.3 Disallowed Keywords

The following Keywords should not be added to the ConnectOptions entry, as they are set by the Agent Configuration Setup dialog, or are applied automatically by the Request Broker:

APP
DATABASE (see below)
DRIVER
DSN
FILEDSN
PWD
UID (see below)

If any of the above Keywords are used, the agent will probably fail to log on to the Database server, as duplicate Connect Options may have been supplied.

Note: It is possible to add *DATABASE=* and *UID=* entries, provided that the Database:Name and Database:Username edit controls in the Agent Configuration dialog are left blank. Note the comments for the keyword, *TrustedConnection* above, which affects the handling of the UID keyword.

The OpenLink SQL2000 agent does not perform any consistency or validity checking on the ConnectOptions entry, and passes the Connection options as entered to the SQL Server. If invalid keywords or values, or unreadable entries are supplied, the SQL Server may reject the login.

The exception is the case where the ConnectOptions entry contains no recognizable entries, in which case it is assumed to be the value for the keyword, *SERVER*. A completely corrupted ConnectOptions entry will therefore probably try to connect to an invalid server.