



## **Connecting to Your Database**

**PowerBuilder®**

DOCUMENT ID: 37776-01-0900-01

LAST REVISED: March 2003

Copyright © 1989-2003 by Sybase, Inc. All rights reserved.

This publication pertains to Sybase software and to any subsequent release until otherwise indicated in new editions or technical notes. Information in this document is subject to change without notice. The software described herein is furnished under a license agreement, and it may be used or copied only in accordance with the terms of that agreement.

To order additional documents, U.S. and Canadian customers should call Customer Fulfillment at (800) 685-8225, fax (617) 229-9845.

Customers in other countries with a U.S. license agreement may contact Customer Fulfillment via the above fax number. All other international customers should contact their Sybase subsidiary or local distributor. Upgrades are provided only at regularly scheduled software release dates. No part of this publication may be reproduced, transmitted, or translated in any form or by any means, electronic, mechanical, manual, optical, or otherwise, without the prior written permission of Sybase, Inc.

Sybase, the Sybase logo, AccelaTrade, ADA Workbench, Adaptable Windowing Environment, Adaptive Component Architecture, Adaptive Server, Adaptive Server Anywhere, Adaptive Server Enterprise, Adaptive Server Enterprise Monitor, Adaptive Server Enterprise Replication, Adaptive Server Everywhere, Adaptive Server IQ, Adaptive Warehouse, Anywhere Studio, Application Manager, AppModeler, APT Workbench, APT-Build, APT-Edit, APT-Execute, APT-FORMS, APT-Translator, APT-Library, Backup Server, BizTracker, ClearConnect, Client-Library, Client Services, Convoy/DM, Copernicus, Data Pipeline, Data Workbench, DataArchitect, Database Analyzer, DataExpress, DataServer, DataWindow, DB-Library, dbQueue, Developers Workbench, Direct Connect Anywhere, DirectConnect, Distribution Director, e-ADK, E-Anywhere, e-Biz Integrator, E-Whatever, EC Gateway, ECMAP, ECRTP, eFulfillment Accelerator, Embedded SQL, EMS, Enterprise Application Studio, Enterprise Client/Server, Enterprise Connect, Enterprise Data Studio, Enterprise Manager, Enterprise SQL Server Manager, Enterprise Work Architecture, Enterprise Work Designer, Enterprise Work Modeler, eProcurement Accelerator, EWA, Financial Fusion, Financial Fusion Server, Gateway Manager, GlobalFIX, ImpactNow, Industry Warehouse Studio, InfoMaker, Information Anywhere, Information Everywhere, InformationConnect, InternetBuilder, iScript, Jaguar CTS, jConnect for JDBC, MainframeConnect, Maintenance Express, MDI Access Server, MDI Database Gateway, media.splash, MetaWorks, MySupport, Net-Gateway, Net-Library, New Era of Networks, ObjectConnect, ObjectCycle, OmniConnect, OmniSQL Access Module, OmniSQL Toolkit, Open Biz, Open Client, Open ClientConnect, Open Client/Server, Open Client/Server Interfaces, Open Gateway, Open Server, Open ServerConnect, Open Solutions, Optima++, PB-Gen, PC APT Execute, PC Net Library, Power++, power.stop, PowerAMC, PowerBuilder, PowerBuilder Foundation Class Library, PowerDesigner, PowerDimensions, PowerDynamo, PowerJ, PowerScript, PowerSite, PowerSocket, Powersoft, PowerStage, PowerStudio, PowerTips, Powersoft Portfolio, Powersoft Professional, PowerWare Desktop, PowerWare Enterprise, ProcessAnalyst, Rapport, Report Workbench, Report-Execute, Replication Agent, Replication Driver, Replication Server, Replication Server Manager, Replication Toolkit, Resource Manager, RW-DisplayLib, S-Designor, SDF, Secure SQL Server, Secure SQL Toolset, Security Guardian, SKILS, smart.partners, smart.parts, smart.script, SQL Advantage, SQL Anywhere, SQL Anywhere Studio, SQL Code Checker, SQL Debug, SQL Edit, SQL Edit/TPU, SQL Everywhere, SQL Modeler, SQL Remote, SQL Server, SQL Server Manager, SQL SMART, SQL Toolset, SQL Server/CFT, SQL Server/DBM, SQL Server SNMP SubAgent, SQL Station, SQLJ, STEP, SupportNow, S.W.I.F.T. Message Format Libraries, Sybase Central, Sybase Client/Server Interfaces, Sybase Financial Server, Sybase Gateways, Sybase MPP, Sybase SQL Desktop, Sybase SQL Lifecycle, Sybase SQL Workgroup, Sybase User Workbench, SybaseWare, Syber Financial, SyberAssist, SyBooks, System 10, System 11, System XI (logo), SystemTools, Tabular Data Stream, TradeForce, Transact-SQL, Translation Toolkit, UltraLite.NET, UNIBOM, Unilib, Uninull, Unisep, Unistring, URK Runtime Kit for UniCode, Viewer, Visual Components, VisualSpeller, VisualWriter, VQL, WarehouseArchitect, Warehouse Control Center, Warehouse Studio, Warehouse WORKS, Watcom, Watcom SQL, Watcom SQL Server, Web Deployment Kit, Web.PB, Web.SQL, WebSights, WebViewer, WorkGroup SQL Server, XA-Library, XA-Server and XP Server are trademarks of Sybase, Inc. 11/02

Unicode and the Unicode Logo are registered trademarks of Unicode, Inc.

All other company and product names used herein may be trademarks or registered trademarks of their respective companies.

Use, duplication, or disclosure by the government is subject to the restrictions set forth in subparagraph (c)(1)(ii) of DFARS 52.227-7013 for the DOD and as set forth in FAR 52.227-19(a)-(d) for civilian agencies.

Sybase, Inc., One Sybase Drive, Dublin, CA 94568.

# Contents

About This Book .....	ix
-----------------------	----

## **PART 1                      INTRODUCTION TO DATABASE CONNECTIONS**

<b>CHAPTER 1</b>	<b>Understanding Data Connections .....</b>	<b>3</b>
	How to find the information you need.....	3
	Accessing data in PowerBuilder.....	5
	Accessing Unicode data.....	7
	Accessing the EAS Demo DB .....	7
	Using database profiles.....	8
	About creating database profiles.....	8
	Creating a database profile .....	11
	What to do next .....	13

## **PART 2                      WORKING WITH STANDARD DATABASE INTERFACES**

<b>CHAPTER 2</b>	<b>Using the ODBC Interface .....</b>	<b>17</b>
	Using the ODBC interface.....	17
	What is ODBC? .....	18
	Using ODBC in PowerBuilder.....	19
	Components of an ODBC connection .....	19
	Types of ODBC drivers .....	21
	Ensuring the proper ODBC driver conformance levels .....	22
	Obtaining ODBC drivers.....	24
	Using ODBC drivers with PowerBuilder Desktop .....	24
	Getting help with ODBC drivers .....	25
	Preparing ODBC data sources.....	25
	Defining ODBC data sources .....	26
	How PowerBuilder accesses the data source .....	27
	Defining multiple data sources for the same data .....	29
	Displaying Help for ODBC drivers .....	30
	Selecting an ODBC translator .....	30

	Defining the ODBC interface.....	31
	Sybase Adaptive Server Anywhere.....	32
	Supported versions for Adaptive Server Anywhere.....	32
	Basic software components for Adaptive Server Anywhere....	33
	Preparing to use the Adaptive Server Anywhere data source.	34
	Defining the Adaptive Server Anywhere data source.....	34
	Support for Transact-SQL special timestamp columns.....	36
	What to do next.....	38
<b>CHAPTER 3</b>	<b>Using the JDBC Interface.....</b>	<b>39</b>
	About the JDBC interface.....	39
	What is JDBC?.....	39
	Using the JDBC interface.....	40
	Components of a JDBC connection.....	41
	JDBC registry entries.....	42
	Supported versions for JDBC.....	43
	Supported JDBC datatypes.....	43
	Preparing to use the JDBC interface.....	43
	Defining the JDBC interface.....	45
<b>CHAPTER 4</b>	<b>Using the OLE DB interface.....</b>	<b>47</b>
	About the OLE DB interface.....	47
	What is OLE DB?.....	48
	Components of an OLE DB connection.....	49
	Obtaining OLE DB data providers.....	50
	Supported versions for OLE DB.....	51
	Preparing to use the OLE DB interface.....	51
	Defining the OLE DB interface.....	53
<b>PART 3</b>	<b>WORKING WITH NATIVE DATABASE INTERFACES</b>	
<b>CHAPTER 5</b>	<b>Using Native Database Interfaces.....</b>	<b>57</b>
	About native database interfaces.....	58
	What is a native database interface?.....	58
	Components of a database interface connection.....	58
	Using a native database interface.....	60
	Informix.....	61
	Supported versions for Informix.....	61
	Supported Informix datatypes.....	62
	Basic software components for Informix.....	64
	Preparing to use the Informix database.....	64
	Defining the Informix database interface.....	66

Accessing serial values in a PowerBuilder script .....	67
What to do next .....	67
Microsoft SQL Server .....	68
Supported versions for SQL Server .....	68
Supported SQL Server datatypes .....	68
Basic software components for SQL Server .....	69
Preparing to use the SQL Server database .....	69
Defining the SQL Server database interface .....	71
What to do next .....	71
Oracle.....	72
Supported versions for Oracle.....	72
Supported Oracle datatypes.....	72
Basic software components for Oracle.....	74
Preparing to use the Oracle database.....	74
Defining the Oracle database interface .....	77
Using Oracle stored procedures as a data source .....	78
Using Oracle user-defined types .....	82
What to do next .....	84
Adaptive Server Enterprise .....	85
Supported versions for Adaptive Server.....	85
Supported Adaptive Server datatypes.....	86
Basic software components for Adaptive Server.....	88
Preparing to use the Adaptive Server database.....	88
Defining the Adaptive Server database interface .....	91
Using Open Client security services.....	92
Using Open Client directory services .....	94
Using PRINT statements in Adaptive Server stored procedures .....	99
Creating a DataWindow based on a heterogeneous cross-database join .....	99
What to do next .....	99
Installing PowerBuilder stored procedures in Adaptive Server databases .....	99
What are the PowerBuilder stored procedure scripts? .....	100
How to run the scripts.....	103
DirectConnect .....	105
Using the DirectConnect interface.....	105
Basic software components for the DirectConnect interface. ....	108
Supported versions for the DirectConnect interface.....	110
Supported DirectConnect interface datatypes.....	110
Preparing to use the database with DirectConnect .....	111
Defining the DirectConnect interface.....	114
Creating the extended attribute system tables in DB2 databases .....	114
Creating the extended attribute system tables .....	114
Using the DB2SYSPB.SQL script .....	115

## PART 4

## WORKING WITH DATABASE CONNECTIONS

### CHAPTER 6

<b>Managing Database Connections .....</b>	<b>119</b>
About database connections.....	120
When database connections occur .....	120
Using database profiles.....	120
Connecting to a database .....	121
Selecting a database profile .....	121
What happens when you connect .....	123
Specifying passwords in database profiles .....	124
Using the Preview tab to connect in a PowerBuilder application .	125
Maintaining database profiles .....	125
Sharing database profiles .....	126
About shared database profiles.....	126
Setting up shared database profiles.....	127
Using shared database profiles to connect .....	128
Making local changes to shared database profiles .....	129
Maintaining shared database profiles.....	129
Importing and exporting database profiles .....	130
About the PowerBuilder extended attribute system tables.....	131
Logging on to your database for the first time .....	132
Displaying the PowerBuilder extended attribute system tables....	132
Contents of the extended attribute system tables .....	134
Controlling system table access.....	135

### CHAPTER 7

<b>Setting Additional Connection Parameters.....</b>	<b>137</b>
Basic steps for setting connection parameters .....	137
About the Database Profile Setup dialog box .....	138
Setting DBParm parameters .....	139
Setting DBParm parameters in the development environment	139
Setting DBParm parameters in a PowerBuilder application script	140
Setting database preferences .....	144
Setting database preferences in the development environment ..	145
Setting AutoCommit and Lock in a PowerBuilder application script	150

---

<b>CHAPTER 8</b>	<b>Troubleshooting Your Connection.....</b>	<b>157</b>
	Overview of troubleshooting tools .....	157
	Using the Database Trace tool.....	158
	About the Database Trace tool.....	158
	Starting the Database Trace tool.....	160
	Stopping the Database Trace tool.....	166
	Specifying a nondefault Database Trace log.....	167
	Using the Database Trace log.....	168
	Sample Database Trace output.....	170
	Using the ODBC Driver Manager Trace.....	172
	About ODBC Driver Manager Trace.....	172
	Starting ODBC Driver Manager Trace.....	173
	Stopping ODBC Driver Manager Trace.....	178
	Viewing the ODBC Driver Manager Trace log.....	180
	Sample ODBC Driver Manager Trace output.....	180
	Using the JDBC Driver Manager Trace.....	184
	About JDBC Driver Manager Trace.....	184
	Starting JDBC Driver Manager Trace.....	185
	Stopping JDBC Driver Manager Trace.....	189
	Viewing the JDBC Driver Manager Trace log.....	191
 <b>PART 5</b>	 <b>WORKING WITH TRANSACTION SERVERS</b>	
 <b>CHAPTER 9</b>	 <b>Making Database Connections in PowerBuilder Components</b>	 <b>195</b>
	Deploying a component to EAServer .....	195
	Supported database connections when using Shared Connection	
	196	
	Supported database connections when using Microsoft DTC	197
	Supported database connections when using OTS/XA .....	197
	Using the SYJ database interface .....	197
	Using the JDB database interface.....	198
	Specifying AutoCommit mode .....	199
	Deploying a COM component to MTS .....	199
	Using the ODBC database interface .....	199
	Using the Oracle database interface .....	200
	DBParm support for PowerBuilder components .....	200

<b>PART 6</b>	<b>APPENDIX</b>	
APPENDIX A	<b>Adding Functions to the PBODB90 Initialization File .....</b>	<b>203</b>
	About the PBODB90 initialization file .....	203
	Adding functions to the PBODB90 initialization file .....	204
	Adding functions to an existing section in the file .....	204
	Adding functions to a new section in the file .....	208
	<b>Index .....</b>	<b>211</b>



# About This Book

## Audience

This book is for anyone who uses PowerBuilder® to connect to a database. It assumes that you are familiar with the database you are using and have installed the server and client software required to access the data.

## How to use this book

This book describes how to connect to a database in PowerBuilder by using a standard or native database interface. It gives procedures for preparing, defining, establishing, maintaining, and troubleshooting your database connections. For an overview of the steps you need to take, see Table 1-1 on page 3.

## Related documents

For detailed information about supported database interfaces, DBParm parameters, and database preferences, see the Database Connectivity section in the online Help. For a complete list of PowerBuilder documentation, see PowerBuilder *Getting Started*.

## Other sources of information

Use the Sybase Technical Library CD and the Technical Library Product Manuals Web site to learn more about your product:

- The Technical Library CD contains product manuals and is included with your software. The DynaText reader (included on the Technical Library CD) allows you to access technical information about your product in an easy-to-use format.

Refer to the *Technical Library Installation Guide* in your documentation package for instructions on installing and starting the Technical Library.

- The Technical Library Product Manuals Web site is an HTML version of the Technical Library CD that you can access using a standard Web browser. In addition to product manuals, you will find links to EBFs/Updates, Technical Documents, Case Management, Solved Cases, newsgroups, and the Sybase Developer Network.

To access the Technical Library Product Manuals Web site, go to Product Manuals at <http://www.sybase.com/support/manuals/>.

---

## Conventions

The formatting conventions used in this manual are:

Formatting example	To indicate
Retrieve and Update	When used in descriptive text, this font indicates: <ul style="list-style-type: none"><li>• Command, function, and method names</li><li>• Keywords such as true, false, and null</li><li>• Datatypes such as integer and char</li><li>• Database column names such as emp_id and f_name</li><li>• User-defined objects such as dw_emp or w_main</li></ul>
<i>variable or file name</i>	When used in descriptive text and syntax descriptions, oblique font indicates: <ul style="list-style-type: none"><li>• Variables, such as <i>myCounter</i></li><li>• Parts of input text that must be substituted, such as <i>pblname.pbd</i></li><li>• File and path names</li></ul>
File>Save	Menu names and menu items are displayed in plain text. The greater than symbol (>) shows you how to navigate menu selections. For example, File>Save indicates “select Save from the File menu.”
<code>dw_1.Update ( )</code>	Monospace font indicates: <ul style="list-style-type: none"><li>• Information that you enter in a dialog box or on a command line</li><li>• Sample script fragments</li><li>• Sample output fragments</li></ul>

## If you need help

Each Sybase installation that has purchased a support contract has one or more designated people who are authorized to contact Sybase Technical Support. If you cannot resolve a problem using the manuals or online help, please have the designated person contact Sybase Technical Support or the Sybase subsidiary in your area.

## PART 1

# Introduction to Database Connections

This part introduces data connections in PowerBuilder. It gives an overview of the concepts and procedures for connecting to a database in the PowerBuilder development environment.



About this chapter

This chapter gives an overview of the concepts and procedures for connecting to a database in the PowerBuilder development environment.

Contents

Topic	Page
How to find the information you need	3
Accessing data in PowerBuilder	5
Accessing Unicode data	7
Accessing the EAS Demo DB	7
Using database profiles	8
What to do next	13

## How to find the information you need

What's in this book

When you work with PowerBuilder, database connections can occur in the development environment or in an application script.

This book describes how to connect to your database *in the PowerBuilder development environment*.

For information about connecting to a database in a PowerBuilder application script, see *Application Techniques*.

Basic connection procedure

Table 1-1 gives an overview of the connection procedure and indicates where you can find detailed information about each step.

**Table 1-1: Basic connection procedure**

Step	Action	Details	See
1	(Optional) Get an introduction to database connections in PowerBuilder.	If necessary, learn more about how PowerBuilder connects to a database in the development environment.	Chapter 1 (this chapter)

Step	Action	Details	See
2	Prepare to use the data source or database before connecting to it for the first time in PowerBuilder.	Outside PowerBuilder, install the required network, database server, and database client software and verify that you can connect to the database.	<p><i>For ODBC data sources:</i> Chapter 2, "Using the ODBC Interface"</p> <p><i>For JDBC data sources:</i> Chapter 3, "Using the JDBC Interface"</p> <p><i>For OLE DB data sources:</i> Chapter 4, "Using the OLE DB interface"</p> <p><i>For native database interfaces:</i> Chapter 5, "Using Native Database Interfaces"</p>
3	Install the ODBC driver, OLE DB data provider, or native database interface.	Install the ODBC data source driver (if supported on your platform), OLE DB data provider, or native database interface required to access your data.	For a list of what is supported on your platform: "Supported Database Interfaces" in online Help
4	Define the data source (ODBC connections and some OLE DB drivers).	Create the required configuration for a data source accessed through ODBC.	<i>For ODBC data sources:</i> Chapter 2, "Using the ODBC Interface"
5	Define the database interface.	Create the database profile.	<p><i>For ODBC data sources:</i> Chapter 2, "Using the ODBC Interface"</p> <p><i>For JDBC data sources:</i> Chapter 3, "Using the JDBC Interface"</p> <p><i>For OLE DB data sources:</i> Chapter 4, "Using the OLE DB interface"</p> <p><i>For native database interfaces:</i> Chapter 5, "Using Native Database Interfaces"</p> <p><i>For PowerBuilder components:</i> Chapter 9, "Making Database Connections in PowerBuilder Components"</p>
6	Define the EAServer connection.	Create an EAServer profile.	Chapter 9, "Making Database Connections in PowerBuilder Components"
7	Connect to the data source or database.	Access the data in PowerBuilder.	Chapter 6, "Managing Database Connections"

Step	Action	Details	See
8	(Optional) Set additional connection parameters.	If necessary, set DBParm parameters and database preferences to fine-tune your database connection and take advantage of DBMS-specific features that your interface supports.	<i>For procedures:</i> Chapter 7, “Setting Additional Connection Parameters” <i>For DBParm descriptions:</i> online Help <i>For database preference descriptions:</i> online Help
9	(Optional) Troubleshoot the data connection.	If necessary, use the trace tools to troubleshoot problems with your connection.	Chapter 8, “Troubleshooting Your Connection”

## Accessing data in PowerBuilder

There are several ways to access data in the PowerBuilder development environment:

- Through one of the standard database interfaces such as ODBC, JDBC, or OLE DB
- Through one of the native database interfaces

### Standard database interfaces

A standard database interface communicates with a database through a standard-compliant driver (in the case of ODBC and JDBC) or data provider (in the case of OLE DB). The standard-compliant driver or data provider translates the abstract function calls defined by the standard’s API into calls that are understood by a specific database. To use a standard interface, you need to install the standard’s API and a suitable driver or data provider. Then, install the standard database interface you want to use to access your DBMS by selecting the interface in the PowerBuilder Setup program.

PowerBuilder currently supports the following standard interfaces:

- Open Database Connectivity (ODBC)
- Java Database Connectivity (JDBC)
- Microsoft’s Universal Data Access Component OLE DB

Native database interfaces

A native database interface communicates with a database through a direct connection. It communicates to a database using that database's native API.

To access data through one of the native database interfaces, you must first install the appropriate database software on the server and client workstations at your site. Then, install the native database interface that accesses your DBMS by selecting the interface in the PowerBuilder Setup program.

For example, if you have the appropriate Sybase Adaptive Server® Enterprise server and client software installed, you can access the database by installing the Adaptive Server Enterprise database interface.

Loading database interface libraries

PowerBuilder loads the libraries used by a database interface when it connects to the database. PowerBuilder does *not* automatically free the database interface libraries when it disconnects.

While memory use is somewhat increased by this technique (since the loaded database interface libraries continue to be held in memory), the technique improves performance and eliminates problems associated with the freeing and subsequent reloading of libraries experienced by some database connections.

If you want PowerBuilder to free database interface libraries on disconnecting from the database (as it did prior to PowerBuilder 8), you can change its default behavior:

To change the default behavior for	Do this
Connections in the development environment	Select the Free Database Driver Libraries On Disconnect check box on the General tab of the System Options dialog box.
Runtime connections	Set the FreeDBLibraries property of the Application object to TRUE on the General tab of the Properties view in the Application painter or in a script.

---

**EAServer components**

This behavior cannot be controlled when components are deployed to EAServer.

---



## Accessing Unicode data

PowerBuilder supports access to Unicode databases, including the conversion of data between double-byte character sets (DBCS) and Unicode, for Adaptive Server Enterprise 12.5 and Oracle9i. PowerBuilder also supports the use of Unicode datatypes for these databases.

For Adaptive Server® Anywhere 7.x and later releases, PowerBuilder supports access to Unicode databases, including conversion of data between DBCS and Unicode.

For other DBMSs, PowerBuilder can connect, save, and retrieve data in a Unicode database but does not convert data between DBCS and Unicode. When data is saved to the database, PowerBuilder sends ANSI or DBCS data to the database. The database itself must guarantee that the data is saved as Unicode data correctly. When PowerBuilder retrieves data, it assumes the data is ANSI or DBCS.

### Unicode databases

A Unicode database is a database whose character set is set to a Unicode format, such as UTF-8, UTF-16, UCS-2, or UCS-4. All data must be in Unicode format, and any data saved to the database must be converted to Unicode data implicitly or explicitly.

### Unicode columns

A database that uses ANSI (or DBCS) as its character set may use special datatypes to store Unicode data. These datatypes are NCHAR, NVARCHAR, and NVARCHAR2. Columns with this datatype can store *only* Unicode data. Any data saved into such a column must be converted to Unicode explicitly.

## Accessing the EAS Demo DB

PowerBuilder includes a standalone Adaptive Server Anywhere database called the EAS Demo DB. Unless you clear this option in the Setup program, the database is installed automatically. You access tables in the EAS Demo DB when you use the PowerBuilder tutorial.

An Adaptive Server Anywhere database is considered an ODBC data source, because you access it with the Adaptive Server Anywhere ODBC driver.

## Using database profiles

What is a database profile?

A **database profile** is a named set of parameters stored in your system registry that defines a connection to a particular database in the PowerBuilder development environment. You must create a database profile for each data connection.

What you can do

Using database profiles is the easiest way to manage data connections in the PowerBuilder development environment. For example, you can:

- Select a database profile to connect to or switch between databases
- Edit a database profile to customize a connection
- Delete a database profile if you no longer need to access that data
- Import and export database profiles to quickly share connection parameters

For more information

For instructions on using database profiles, see Chapter 6, “Managing Database Connections”.

## About creating database profiles

You work with two dialog boxes when you create a database profile in PowerBuilder: the Database Profiles dialog box and the interface-specific Database Profile Setup dialog box.

---

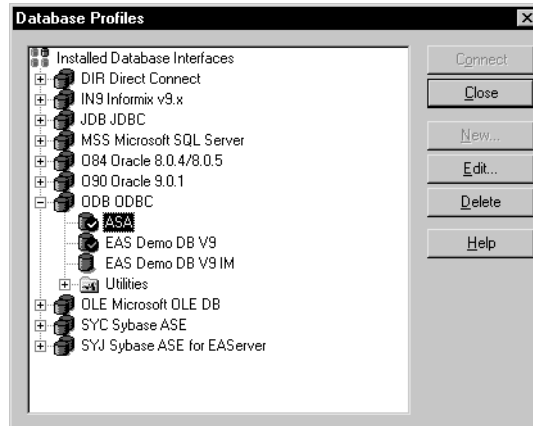
### Using the Database painter to create database profiles

You can also create database profiles from the Database painter’s Objects view.

---

### Database Profiles dialog box

The Database Profiles dialog box uses an easy-to-navigate tree control format to display your installed database interfaces and defined database profiles. You can create, edit, and delete database profiles from this dialog box.



When you run the PowerBuilder Setup program, it updates the Vendors list in the PowerBuilder section of the system registry with the interfaces you install. The Database Profiles dialog box displays the same interfaces that appear in the Vendors list.

---

#### Where the Vendors list is stored

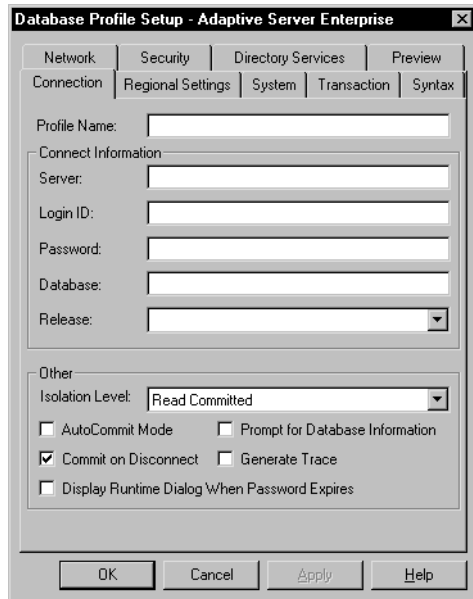
The *Sybase\PowerBuilder\9.0\Vendors* key in *HKEY\_LOCAL\_MACHINE\SOFTWARE* is used for InfoMaker as well as PowerBuilder.

---

For detailed instructions on using the Database Profiles dialog box to connect to a database and manage your profiles, see Chapter 6, “Managing Database Connections”.

Database Profile Setup dialog box

Each database interface has its own Database Profile Setup dialog box where you can set interface-specific connection parameters. For example, if you install the SYC interface (if available) and then select it and click New in the Database Profiles dialog box, the Database Profile Setup - Adaptive Server Enterprise dialog box displays, containing settings for those connection options that apply to this interface.



The Database Profile Setup dialog box groups similar connection parameters on the same tab page and lets you easily set their values by using check boxes, drop-down lists, and text boxes. Basic (required) connection parameters are on the Connection tab page, and additional connection options (DBParm parameters and SQLCA properties) are on the other tab pages.

As you complete the Database Profile Setup dialog box in PowerBuilder, the correct PowerScript connection syntax for each selected option is generated on the Preview tab. You can copy the syntax you want from the Preview tab into a PowerBuilder application script.

Supplying sufficient information in the Database Profile Setup dialog box

For some database interfaces, you may not need to supply values for all boxes in the Database Profile Setup dialog box. If you supply the profile name and click OK, PowerBuilder displays a series of dialog boxes to prompt you for additional information when you connect to the database.

This information can include:

- User ID or login ID
- Password or login password
- Database name
- Server name

For some databases, supplying only the profile name does not give PowerBuilder enough information to prompt you for additional connection values. For these interfaces, you must supply values for all applicable boxes in the Database Profile Setup dialog box.

For information about the values you should supply for your connection, click Help in the Database Profile Setup dialog box for your interface.

## Creating a database profile

To create a new database profile for a database interface, you must complete the Database Profile Setup dialog box for the interface you are using to access the database.

### ❖ To create a database profile for a database interface:

- 1 Click the Database Profile button in the PowerBar.

The Database Profiles dialog box displays, listing your installed database interfaces. To see a list of database profiles defined for a particular interface, click the plus sign to the left of the interface name or double-click the interface name to expand the list.

- 2 Highlight an interface name and click New.

The Database Profile Setup dialog box for the selected interface displays. For example, if you select the SYC interface, the Database Profile Setup - Adaptive Server Enterprise dialog box displays.

---

#### **Client software and interface must be installed**

To display the Database Profile Setup dialog box for your interface, the required client software and native database interface must be properly installed and configured. For specific instructions for your database interface, see the chapter on using the interface.

---

- 3 On the Connection tab page, type the profile name and supply values for any other basic parameters your interface requires to connect.

For information about the basic connection parameters for your interface and the values you should supply, click Help.

---

**About the DBMS identifier**

You do *not* need to specify the DBMS identifier in a database profile. When you create a new profile for any installed database interface, PowerBuilder generates the correct DBMS connection syntax for you.

---

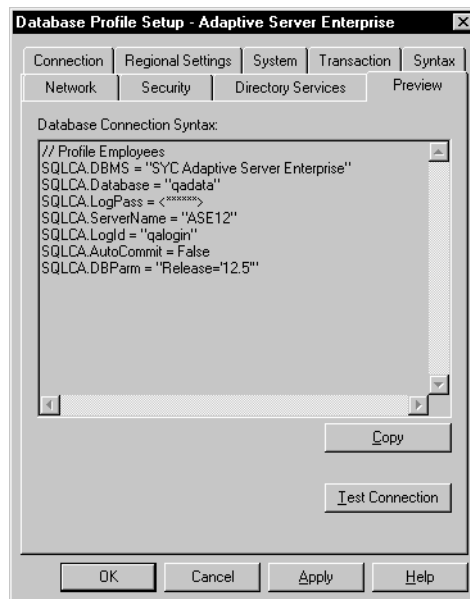
- 4 (Optional) On the other tab pages, supply values for any additional connection options (DBParm parameters and SQLCA properties) to take advantage of DBMS-specific features that your interface supports.

For information about the additional connection parameters for your interface and the values you should supply, click Help.

- 5 (Optional) Click the Preview tab if you want to see the PowerScript connection syntax that PowerBuilder generates for each selected option.

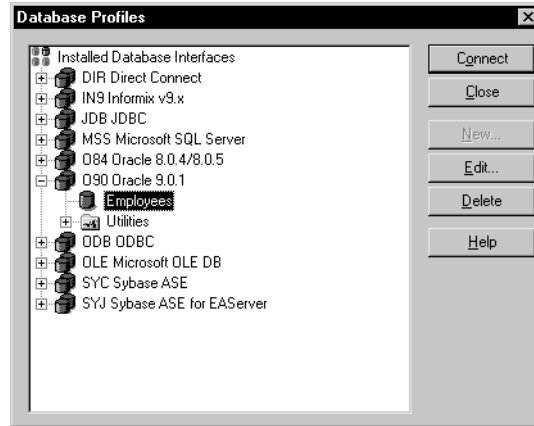
You can copy the PowerScript connection syntax from the Preview tab directly into a PowerBuilder application script.

For instructions on using the Preview tab to help you connect in a PowerBuilder application, see the section on using Transaction objects in *Application Techniques*.



- 6 Click OK to save your changes and close the Database Profile Setup dialog box. (To save your changes on a particular tab page *without* closing the dialog box, click Apply.)

The Database Profiles dialog box displays, with the new profile name highlighted under the appropriate interface. The database profile values are saved in the system registry.



## What to do next

For instructions on preparing to use and then defining an ODBC data source, see Chapter 2, “Using the ODBC Interface”.

For instructions on preparing to use and then defining a JDBC database interface, see Chapter 3, “Using the JDBC Interface”.

For instructions on preparing to use and then defining an OLE DB data provider, see Chapter 4, “Using the OLE DB interface”.

For instructions on preparing to use and then defining a native database interface, see Chapter 5, “Using Native Database Interfaces”.





## PART 2

# Working with Standard Database Interfaces

This part describes how to set up and define database connections accessed through one of the standard database interfaces.



# Using the ODBC Interface

## About this chapter

This chapter gives an introduction to the ODBC interface and then describes how to prepare to use the data source, how to define the data source, and how to define the ODBC database profile. It also describes how to use the Sybase Adaptive Server Anywhere ODBC driver.

## Contents

Topic	Page
Using the ODBC interface	17
Preparing ODBC data sources	25
Defining ODBC data sources	26
Defining the ODBC interface	31
Sybase Adaptive Server Anywhere	32

## For more information

This chapter gives general information about preparing to use and defining each ODBC data source. For more detailed information:

- Use the online Help provided by the driver vendor, as described in “Displaying Help for ODBC drivers” on page 30. This Help provides important details about using the data source.
- Check to see if there is a technical document that describes how to connect to your ODBC data source. Any updated information about connectivity issues is available from the Sybase Customer Service and Support Web site at <http://support.sybase.com>.

## Using the ODBC interface

You can access a wide variety of ODBC data sources in PowerBuilder. This section describes what you need to know to use ODBC connections to access your data in PowerBuilder.

---

### ODBC drivers and data sources

For a complete list of the ODBC drivers supplied with PowerBuilder and the data sources they access, see “Database Interfaces” in online Help.

---

## What is ODBC?

The ODBC API

**Open Database Connectivity (ODBC)** is a standard application programming interface (API) developed by Microsoft. It allows a single application to access a variety of data sources for which ODBC-compliant drivers exist. The application uses Structured Query Language (SQL) as the standard data access language.

The ODBC API defines the following:

- A library of ODBC function calls that connect to the data source, execute SQL statements, and retrieve results
- A standard way to connect and log in to a DBMS
- SQL syntax based on the X/Open and SQL Access Group (SAG) CAE specification (1992)
- A standard representation for datatypes
- A standard set of error codes

Accessing ODBC data sources

Applications that provide an ODBC interface, like PowerBuilder, can access data sources for which an ODBC driver exists. An **ODBC data source driver** is a dynamic link library (DLL) that implements ODBC function calls. The application invokes the ODBC driver to access a particular data source.

---

### For up-to-date information

As connectivity to additional ODBC data sources is supported in future releases of PowerBuilder, updated information will be available from the Sybase Customer Service and Support Web site at <http://support.sybase.com>. Check the Release Bulletin for the latest information about ODBC drivers provided with your edition of PowerBuilder.

---

## Using ODBC in PowerBuilder

What you can do

The following ODBC connectivity features are available in PowerBuilder:

- Connect to an Adaptive Server Anywhere standalone database (including the EAS Demo DB) using the Adaptive Server Anywhere ODBC driver and the ODBC interface.
- Create and delete local Adaptive Server Anywhere databases.

For instructions, see the *User's Guide*.

- Use Sybase-supplied DataDirect ODBC drivers to access your data.

For a list of the ODBC drivers supplied, see "Database Interfaces" in online Help.

- In all editions *except* PowerBuilder Desktop, use Level 1 or later ODBC-compliant drivers obtained from vendors other than Sybase to access your data.

See "Obtaining ODBC drivers" on page 24.

- Use Microsoft's ODBC Data Source Administrator to define ODBC data sources.

See "Defining ODBC data sources" on page 26.

## Components of an ODBC connection

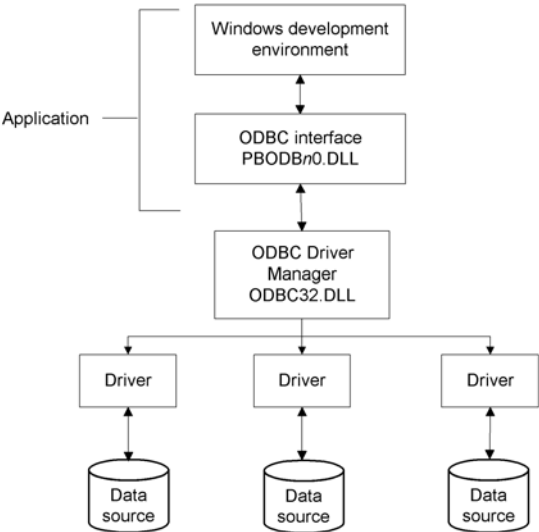
How an ODBC connection is made

When you access an ODBC data source in PowerBuilder, your connection goes through several layers before reaching the data source. It is important to understand that each layer represents a separate component of the connection, and that each component may come from a different vendor.

Because ODBC is a standard API, PowerBuilder uses the same interface to access every ODBC data source. As long as a driver is ODBC compliant, PowerBuilder can access it through the ODBC interface to the ODBC Driver Manager. The development environment and the ODBC interface work together as the application component.

Figure 2-1 shows the general components of an ODBC connection.

Figure 2-1: Components of an ODBC connection



Component descriptions

Table 2-1 gives the provider and a brief description of each ODBC component shown in the diagram:

Table 2-1: Provider and function of ODBC connection components

Component	Provider	What it does
Application	Sybase	<p>Calls ODBC functions to submit SQL statements, catalog requests, and retrieve results from a data source.</p> <p>PowerBuilder uses the same ODBC interface to access all ODBC data sources.</p>
ODBC Driver Manager	Microsoft	Installs, loads, and unloads drivers for an application.
Driver	Driver vendor	<p>Processes ODBC function calls, submits SQL requests to a particular data source, and returns results to an application.</p> <p>If necessary, translates an application's request so that it conforms to the SQL syntax supported by the back-end database. See "Types of ODBC drivers" next.</p>
Data source	DBMS or database vendor	<p>Stores and manages data for an application. Consists of the data to be accessed and its associated DBMS, operating system, and (if present) network software that accesses the DBMS.</p>

## Types of ODBC drivers

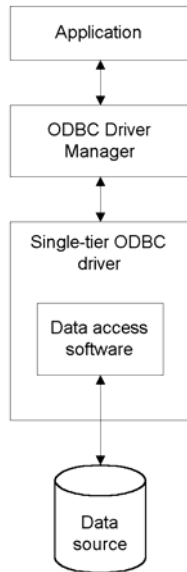
When PowerBuilder is connected to an ODBC data source, you may see messages from the ODBC driver that include the words *single-tier* or *multiple-tier*. These terms refer to the two types of drivers defined by the ODBC standard.

### Single-tier driver

A **single-tier ODBC driver** processes both ODBC functions and SQL statements. In other words, a single-tier driver includes the data access software required to manage the data source file and catalog tables.

An example of a single-tier ODBC driver is one that accesses Xbase files, such as the DataDirect dBASE ODBC driver.

**Figure 2-2: Single-tier ODBC driver**

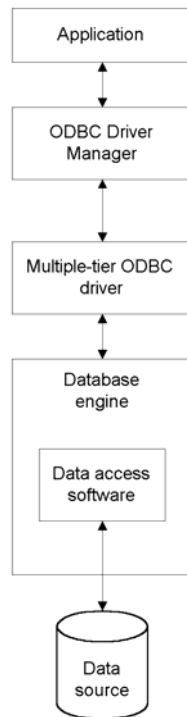


### Multiple-tier driver

A **multiple-tier ODBC driver** processes ODBC functions, but sends SQL statements to the database engine for processing. Unlike the single-tier driver, a multiple-tier driver does not include the data access software required to manage the data directly.

An example of a multiple-tier ODBC driver is the Sybase Adaptive Server Anywhere driver.

**Figure 2-3: Multi-tier ODBC driver**



## Ensuring the proper ODBC driver conformance levels

You can access data in PowerBuilder Enterprise or PowerBuilder Professional with ODBC drivers obtained from vendors *other* than Sybase, such as DBMS vendors.

An ODBC driver obtained from another vendor must meet certain conformance requirements to ensure that it works properly with PowerBuilder. This section describes how to make sure your driver meets these requirements.

### What are ODBC conformance levels?

PowerBuilder can access many data sources for which ODBC-compliant drivers exist. However, ODBC drivers manufactured by different vendors may vary widely in the functions they provide.



To ensure a standard level of compliance with the ODBC interface, and to provide a means by which application vendors can determine if a specific driver provides the functions they need, ODBC defines conformance levels for drivers in two areas:

- **API** Deals with supported ODBC function calls
- **SQL grammar** Deals with supported SQL statements and SQL datatypes

#### API conformance levels

ODBC defines three API conformance levels, in order of increasing functionality:

- **Core** A set of core API functions that corresponds to the functions in the ISO Call Level Interface (CLI) and X/Open CLI specification
- **Level 1** Includes all Core API functions and several extended functions usually available in an OLTP relational DBMS
- **Level 2** Includes all Core and Level 1 API functions and additional extended functions

#### ❖ To ensure the proper ODBC driver API conformance level:

- Sybase recommends that the ODBC drivers you use with PowerBuilder meet *Level 1 or higher* API conformance requirements. However, PowerBuilder may also work with drivers that meet Core level API conformance requirements.

#### SQL conformance levels

ODBC defines three SQL grammar conformance levels, in order of increasing functionality:

- **Minimum** A set of SQL statements and datatypes that meets a basic level of ODBC conformance
- **Core** Includes all Minimum SQL grammar and additional statements and datatypes that roughly correspond to the X/Open and SAG CAE specification (1992)
- **Extended** Includes all Minimum and Core SQL grammar and an extended set of statements and datatypes that support common DBMS extensions to SQL

#### ❖ To ensure the proper ODBC driver SQL conformance level:

- Sybase recommends that the ODBC drivers you use with PowerBuilder meet *Core or higher* SQL conformance requirements. However, PowerBuilder may also work with drivers that meet Minimum level SQL conformance requirements.

## Obtaining ODBC drivers

### Two sources

There are two ways that you can obtain ODBC drivers for use with PowerBuilder:

- **From Sybase (recommended)** Install one or more of the ODBC drivers shipped with PowerBuilder. You can do this when you first install PowerBuilder, or later.
- **From another vendor** PowerBuilder Enterprise and PowerBuilder Professional let you access data with *any* Level 1 or higher ODBC-compliant drivers obtained from a vendor other than Sybase. In most cases, these drivers will work with PowerBuilder.

## Using ODBC drivers with PowerBuilder Desktop

### Using ODBC drivers that come with Desktop

If you are using PowerBuilder Desktop, you can access data using only the ODBC drivers that are shipped with the product. For a list of these drivers, see ODBC drivers in the online Help. Unlike PowerBuilder Enterprise and PowerBuilder Professional, with PowerBuilder Desktop you *cannot* use an ODBC driver obtained from another vendor.

### Using existing Microsoft ODBC drivers

If you already have version 2.0 or later of any of the following Microsoft ODBC drivers installed and properly configured, you *can* use these drivers with PowerBuilder Desktop to connect to your data source:

Microsoft Access (\*.MDB)  
Microsoft Btrieve (\*.DDF)  
Microsoft dBASE (\*.DBF)  
Microsoft Excel (\*.XLS)  
Microsoft FoxPro (\*.DBF)  
Microsoft Paradox (\*.DB)  
Microsoft Text (\*.CSV, \*.TXT)

---

### Using DataDirect drivers is recommended

PowerBuilder Desktop comes with DataDirect ODBC drivers for several of these data sources. You should use the DataDirect drivers whenever possible to access these data sources.

---

## Getting help with ODBC drivers

To ensure that you have up-to-date and accurate information about using your ODBC driver with PowerBuilder, get help as needed by doing one or more of the following:

To get help on	Do this
Using the ODBC Data Source Administrator	Click the Help button on each tab.
Completing the ODBC setup dialog box for your driver	Click the Help button (if present) in the ODBC setup dialog box for your driver.
Using Adaptive Server Anywhere	See the Adaptive Server Anywhere documentation.
Using an ODBC driver obtained from a vendor other than Sybase	See the vendor's documentation for that driver.
Troubleshooting your ODBC connection	Check for a technical document that describes how to connect to your ODBC data source. Updated information about connectivity issues is available on the Sybase Customer Service and Support Web site at <a href="http://support.sybase.com">http://support.sybase.com</a> .

## Preparing ODBC data sources

The first step in connecting to an ODBC data source is preparing the data source. This ensures that you are able to connect to the data source and use your data in PowerBuilder.

You prepare to use a data source *outside* PowerBuilder *before* you start the product, define the data source, and connect to it. The requirements differ for each data source, but in general, preparing to use a data source involves the following steps.

❖ **To prepare to use an ODBC data source with PowerBuilder:**

- 1 If network software is required to access the data source, make sure it is properly installed and configured at your site and on the client workstation.
- 2 If database software is required, make sure it is properly installed and configured on your computer or network server.

- 3 Make sure the required data files are present on your computer or network server.
- 4 Make sure the names of tables and columns you want to access follow standard SQL naming conventions.

Avoid using blank spaces or database-specific reserved words in table and column names. Be aware of the case-sensitivity options of the DBMS. It is safest to use all uppercase characters when naming tables and columns that you want to access in PowerBuilder.

---

**Backquote character not allowed as a delimiter**

The online Help supplied for the DataDirect ODBC drivers indicates that you can use the backquote (‘) character, also known as the *grave* character, as a delimiter for table and column names that do not follow standard SQL naming conventions. However, PowerBuilder does *not* currently allow use of the backquote character as a delimiter for table and column names.

---

- 5 If your database requires it, make sure the tables you want to access have unique indexes.
- 6 Install both of the following using the PowerBuilder Setup program:
  - The ODBC driver that accesses your data source
  - The ODBC interface

## Defining ODBC data sources

Each ODBC data source requires a corresponding ODBC driver to access it. When you define an ODBC data source, you provide information about the data source that the driver requires in order to connect to it. Defining an ODBC data source is often called **configuring** the data source.

After you prepare to use the data source, you must define it using Microsoft's ODBC Data Source Administrator utility. This utility can be accessed from the Control Panel in Windows or PowerBuilder's Database painter.

The rest of this section describes what you need to know to define an ODBC data source in order to access it in the PowerBuilder development environment.

## How PowerBuilder accesses the data source

When you access an ODBC data source in PowerBuilder, there are several initialization files and registry entries on your computer that work with the ODBC interface and driver to make the connection.

### PBODB90 initialization file

Contents	The PBODB90 initialization file is located in the <i>Sybase\Shared\PowerBuilder</i> directory. PowerBuilder uses <i>PBODB90.INI</i> to maintain access to extended functionality in the back-end DBMS, for which ODBC does not provide an API call. Examples of extended functionality are SQL syntax or DBMS-specific function calls.
Editing	<p>In most cases, you do not need to edit the PBODB90 initialization file. In certain situations, however, you may need to add functions to the PBODB90 initialization file for your back-end DBMS.</p> <p>For instructions, see the Appendix, “Adding Functions to the PBODB90 Initialization File”.</p>

### ODBCINST registry entries

Contents	<p>The ODBCINST initialization information is located in the <i>HKEY_LOCAL_MACHINE\SOFTWARE\ODBC\ODBCINST.INI</i> registry key. When you install an ODBC-compliant driver supplied by Sybase or another vendor, <i>ODBCINST.INI</i> is automatically updated with a description of the driver.</p> <p>This description includes:</p> <ul style="list-style-type: none"><li>• The DBMS or data source associated with the driver</li><li>• The drive and directory of the driver and setup DLLs (for some data sources, the driver and setup DLLs are the same)</li><li>• Other driver-specific connection parameters</li></ul>
Editing	You do <i>not</i> need to edit the registry key directly to modify connection information. If your driver uses the information in the <i>ODBCINST.INI</i> registry key, the key is automatically updated when you install the driver. This is true whether the driver is supplied by Sybase or another vendor.

## ODBC registry entries

### Contents

ODBC initialization information is located in the *HKEY\_CURRENT\_USER\SOFTWARE\ODBC\ODBC.INI* registry key. When you define a data source for a particular ODBC driver, the driver writes the values you specify in the ODBC setup dialog box to the *ODBC.INI* registry key.

The *ODBC.INI* key contains a subkey named for each defined data source. Each subkey contains the values specified for that data source in the ODBC setup dialog box. The values may vary for each data source but generally include the following:

- Database
- Driver
- Optional description
- DBMS-specific connection parameters

### Editing

Do *not* edit the *ODBC* subkey directly to modify connection information. Instead, use a tool designed to define ODBC data sources and the ODBC configuration automatically, such as the ODBC Data Source Administrator.

## Database profiles registry entry

### Contents

Database profiles for all data sources are stored in the registry in *HKEY\_CURRENT\_USER\SOFTWARE\Sybase\PowerBuilder\9.0\DatabaseProfiles*.

### Editing

You should *not* need to edit the profiles directly to modify connection information. These files are updated automatically when PowerBuilder creates the database profile as part of the ODBC data source definition.

You can also edit the profile in the Database Profile Setup dialog box or complete the Database Preferences property sheet in PowerBuilder to specify other connection parameters stored in the registry. (For instructions, see Chapter 7, “Setting Additional Connection Parameters”.)

### Example

The following example shows a portion of the database profile for the EAS Demo DB data source:

```
DBMS=ODBC
Database=EAS Demo DB
UserId=dba
DatabasePassword=
LogPassword=
```

```
ServerName=  
LogId=  
Lock=  
DbParm=ConnectionString='DSN=EAS Demo DB;UID=dba;PWD=sql '  
Prompt=0
```

This registry entry example shows the two most important values in a database profile for an ODBC data source:

- **DBMS** The DBMS value (ODBC) indicates that you are using the ODBC interface to connect to the data source.
- **DBParm** The ConnectString DBParm parameter controls your ODBC data source connection. The connect string *must* specify the DSN (data source name) value, which tells ODBC which data source you want to access. When you select a database profile to connect to a data source, ODBC looks in the ODBC.INI registry key for a subkey that corresponds to the data source name in your profile. ODBC then uses the information in the subkey to load the required libraries to connect to the data source. The connect string can also contain the UID (user ID) and PWD (password) values needed to access the data source.

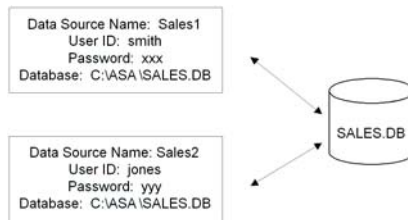
## Defining multiple data sources for the same data

When you define an ODBC data source in PowerBuilder, each data source name must be unique. You can, however, define multiple data sources that access the same data, as long as the data sources have unique names.

For example, assume that your data source is an Adaptive Server Anywhere database located in *C:\ASA\SALES.DB*. Depending on your application, you may want to specify different sets of connection parameters for accessing the database, such as different passwords and user IDs.

To do this, you can define two ODBC data sources named *Sales1* and *Sales2* that specify the same database (*C:\ASA\SALES.DB*) but use different user IDs and passwords. When you connect to the data source using a profile created for either of these data sources, you are using different connection parameters to access the same data.

**Figure 2-4: Using two data sources to access a database**



## Displaying Help for ODBC drivers

The online Help for ODBC drivers in PowerBuilder is provided by the driver vendors. It gives help on:

- Completing the ODBC setup dialog box to define the data source
- Using the ODBC driver to access the data source

### Help for any ODBC driver

Use the following procedure to display vendor-supplied Help when you are in the ODBC setup dialog box for ODBC drivers supplied with PowerBuilder.

❖ **To display Help for any ODBC driver:**

- 1 Click the Help button in the ODBC setup dialog box for your driver.  
A Help window displays, describing features in the setup dialog box.
- 2 Click the Contents button in the Help window to display additional Help topics for this driver.  
Another Help window displays, listing the topics you can view.
- 3 Click an underlined topic to display its Help window.

## Selecting an ODBC translator

What is an ODBC translator?

The ODBC drivers supplied with PowerBuilder allow you to specify a translator when you define the data source. An **ODBC translator** is a DLL that translates data passing between an application and a data source. Typically, translators are used to translate data from one character set to another.



What you do

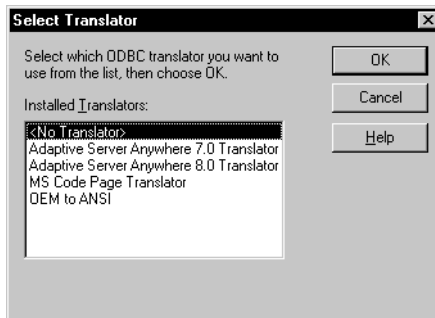
Follow these steps to select a translator for your ODBC driver.

❖ **To select a translator when using an ODBC driver:**

- 1 In the ODBC setup dialog box for your driver, display the Select Translator dialog box.

The way you display the Select Translator dialog box for Sybase-supplied ODBC drivers depends on the driver and Windows platform you are using. Click Help in your driver's setup dialog box for instructions on displaying the Select Translator dialog box.

In the Select Translator dialog box, the translators listed are determined by the values in your *ODBCINST.INI* registry key.



- 2 Select a translator to use from the Installed Translators list.

If you need help using the Select Translator dialog box, click Help.

- 3 Click OK.

The Select Translator dialog box closes and the driver performs the translation.

## Defining the ODBC interface

To define a connection through the ODBC interface, you must create a database profile by supplying values for at least the basic connection parameters in the Database Profile Setup - ODBC dialog box. You can then select this profile at any time to connect to your data source in the development environment.

For information on how to define a database profile, see “Using database profiles” on page 8.

## **Sybase Adaptive Server Anywhere**

This section describes how to prepare and define a Sybase Adaptive Server Anywhere data source in order to connect to it using the Adaptive Server Anywhere ODBC driver.

Adaptive Server Anywhere includes two database servers—a personal database server and a network database server. For information about using Sybase Adaptive Server Anywhere, see the Adaptive Server Anywhere documentation.

### **Supported versions for Adaptive Server Anywhere**

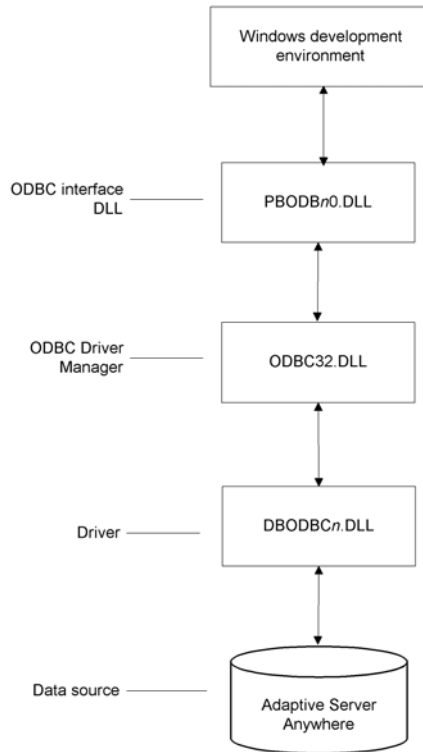
The Adaptive Server Anywhere ODBC driver supports connection to local and remote databases created with the following:

- PowerBuilder running on your computer
- Adaptive Server Anywhere version 8.x
- Adaptive Server Anywhere version 7.x
- Adaptive Server Anywhere version 6.x
- SQL Anywhere version 5.x

## Basic software components for Adaptive Server Anywhere

Figure 2-5 show the basic software components required to connect to an Adaptive Server Anywhere data source in PowerBuilder.

**Figure 2-5: Components of an Adaptive Server Anywhere connection**



## Preparing to use the Adaptive Server Anywhere data source

Before you define and connect to an Adaptive Server Anywhere data source in PowerBuilder, follow these steps to prepare the data source.

### ❖ To prepare an Adaptive Server Anywhere data source:

- 1 Make sure the database file for the Adaptive Server Anywhere data source already exists. You can create a new database by:

- Launching the Create ASA Database utility. This utility can be accessed from the Utilities folder for the ODBC interface in the Database profile or Database painter when PowerBuilder is running on your computer.

This method creates a local Adaptive Server Anywhere database on your computer, and also creates the data source definition and database profile for this connection. (For instructions, see the *User's Guide*.)

- Creating the database some other way, such as with PowerBuilder running on another user's computer or by using Adaptive Server Anywhere outside PowerBuilder. (For instructions, see the Adaptive Server Anywhere documentation.)

- 2 Make sure you have the log file associated with the Adaptive Server Anywhere database so that you can fully recover the database if it becomes corrupted.

If the log file for the Adaptive Server Anywhere database does not exist, the Adaptive Server Anywhere database engine creates it. However, if you are copying or moving a database from another computer or directory, you should copy or move the log file with it.

## Defining the Adaptive Server Anywhere data source

When you create a local Adaptive Server Anywhere database, PowerBuilder automatically creates the data source definition and database profile for you. Therefore, you need only use the following procedure to define an Adaptive Server Anywhere data source when you want to access an Adaptive Server Anywhere database not created using PowerBuilder on your computer.

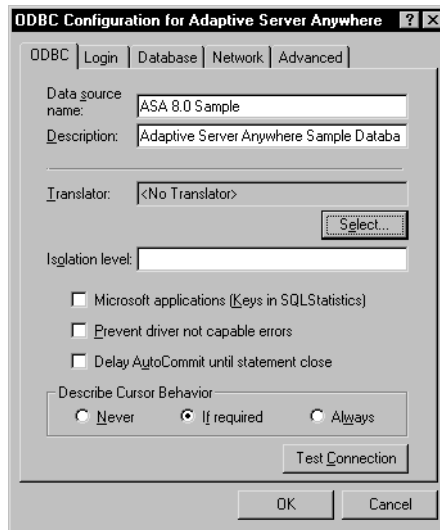
❖ **To define an Adaptive Server Anywhere data source for the Adaptive Server Anywhere driver:**

- 1 Launch the ODBC Data Source Administrator utility. From the User DSN tab, click the Add button.

The Create New Data Source dialog box displays.

- 2 Select the Adaptive Server Anywhere driver and click Finish.

The ODBC Configuration for Adaptive Server Anywhere dialog box displays:



- 3 You must supply the following values:
  - Data source name on the ODBC tab
  - User ID and password on the Login tab
  - Database file on the Database tab

Use the Help button to get information about boxes in the dialog box.

**Using the Browse button**

When you use the Browse button to supply the Database File name (for example, SALES.DB), this name also displays without the extension in both the Data Source Name and Database Name boxes. This may change values that you previously supplied in these boxes.

If you want to specify a different name for the data source or database, you can edit one or both of these boxes *after* using the Browse button.

- 4 (Optional) To select an ODBC translator to translate your data from one character set to another, click the Select button on the ODBC tab.  
See “Selecting an ODBC translator” on page 30.
- 5 Click OK to save the data source definition.

Specifying a Start Line value

When the Adaptive Server Anywhere ODBC driver cannot find a running personal or network database server using the PATH variable and Database Name setting, it uses the commands specified in the Start Line field to start the database servers.

Specify one of the following commands in the Start Line field on the Database tab, where *n* is the version of ASA you are using.

Specify this command	To
dbeng <i>n</i> .exe	Start the personal database server and the database specified in the Database File box
rteng <i>n</i> .exe	Start the runtime database server and the database specified in the Database File box

For information on completing the ODBC Configuration For Adaptive Server Anywhere dialog box, see the Adaptive Server Anywhere documentation.

**Support for Transact-SQL special timestamp columns**

When you work with an Adaptive Server Anywhere table in the Database, DataWindow, or Data Pipeline painter, the default behavior is to treat any column named timestamp as an Adaptive Server Anywhere Transact-SQL special timestamp column.

Creating special  
timestamp columns

You can create a Transact-SQL special timestamp column in an Adaptive Server Anywhere table.

❖ **To create a Transact-SQL special timestamp column in an Adaptive Server Anywhere table in PowerBuilder:**

- 1 Give the name *timestamp* to any column having a timestamp datatype that you want treated as a Transact-SQL special timestamp column. Do this in one of the following ways:
  - In the painter – Select *timestamp* as the column name. (For instructions, see the *User's Guide*.)
  - In a SQL CREATE TABLE statement – Follow the "CREATE TABLE example" next.
- 2 Specify *timestamp* as the default value for the column. Do this in one of the following ways:
  - In the painter – Select *timestamp* as the default value for the column. (For instructions, see the *User's Guide*.)
  - In a SQL CREATE TABLE statement – Follow the "CREATE TABLE example" next.
- 3 If you are working with the table in the Data Pipeline painter, select the initial value *exclude* for the special timestamp column from the drop-down list in the Initial Value column of the workspace.

You must select *exclude* as the initial value to exclude the special timestamp column from INSERT or UPDATE statements.

For instructions, see the *User's Guide*.

CREATE TABLE  
example

The following CREATE TABLE statement defines an Adaptive Server Anywhere table named *timesheet* containing three columns: *employee\_ID* (integer datatype), *hours* (decimal datatype), and *timestamp* (timestamp datatype and *timestamp* default value):

```
CREATE TABLE timesheet (
    employee_ID INTEGER,
    hours DECIMAL,
    timestamp TIMESTAMP default timestamp )
```

Not using special  
timestamp columns

If you want to change the default behavior, you can specify that PowerBuilder *not* treat Adaptive Server Anywhere columns named *timestamp* as Transact-SQL special timestamp columns.

❖ **To specify that PowerBuilder *not* treat columns named *timestamp* as a Transact-SQL special timestamp column:**

- Edit the Sybase Adaptive Server Anywhere section of the *PBODB90* initialization file to change the value of `SQLSrvrTSName` from `'Yes'` to `'No'`.

After making changes in the initialization file, you must reconnect to the database to have them take effect. See the Appendix, “Adding Functions to the PBODB90 Initialization File”.

## What to do next

For instructions on connecting to the ODBC data source, see “Connecting to a database” on page 121.



# Using the JDBC Interface

About this chapter

This chapter describes the JDBC interface and then explains how to prepare to use this interface and how to define the JDBC database profile.

Contents

Topic	Page
About the JDBC interface	39
Preparing to use the JDBC interface	43
Defining the JDBC interface	45

For more information

For more detailed information about JDBC, go to the Java Web site at <http://java.sun.com/products/jdbc>.

## About the JDBC interface

You can access a wide variety of databases through JDBC in PowerBuilder. This section describes what you need to know to use JDBC connections to access your data in PowerBuilder.

## What is JDBC?

The JDBC API

Java Database Connectivity (JDBC) is a standard application programming interface (API) that allows a Java application to access any database that supports Structured Query Language (SQL) as its standard data access language.

The JDBC API includes classes for common SQL database activities so that you can open connections to databases, execute SQL commands, and process results. Consequently, Java programs have the capability to use the familiar SQL programming model of issuing SQL statements and processing the resulting data. The JDBC classes are included in Java 1.1+ and Java 2 as the `java.sql` package.

The JDBC API defines the following:

- A library of JDBC function calls that connect to a database, execute SQL statements, and retrieve results
- A standard way to connect and log in to a DBMS
- SQL syntax based on the X/Open SQL Call Level Interface or X/Open and SQL Access Group (SAG) CAE specification (1992)
- A standard representation for datatypes
- A standard set of error codes

How JDBC APIs are implemented

JDBC API implementations fall into two broad categories: those that communicate with an existing ODBC driver (a JDBC-ODBC bridge) and those that communicate with a native database API (a JDBC driver that converts JDBC calls into the communications protocol used by the specific database vendor). The PowerBuilder implementation of the JDBC interface can be used to connect to any database for which a JDBC-compliant driver exists.

The PowerBuilder JDB interface

A Java Virtual Machine (JVM) is required to interpret and execute the bytecode of a Java program. The PowerBuilder JDB interface supports the Sun Java Runtime Environment (JRE) versions 1.2 and later.

---

#### **JDM and JDS support has been discontinued**

The JDS and JDM interfaces have been discontinued in this release.

---

## Using the JDBC interface

You can use the JDBC interface to develop several types of components and/or applications in PowerBuilder:

- **Thin client/server applications** If a client is already running a JVM (in a running Web browser or inside the operating system), the use of the JDBC interface to access a database does not require the client-side installation and administration of a database driver, which is required when using ODBC.
- **DataWindow objects to be used in a DataWindow Web control for ActiveX** Using the JDBC interface does not require the installation of a database driver on the client, since the JDBC driver can be downloaded with the Web ActiveX in a CAB file.

- **Transactional components to be deployed on EAServer that access a database through the EAServer JDBC interface** Using the JDBC interface allows a PowerBuilder transactional component to share the same transaction as another component.

## Components of a JDBC connection

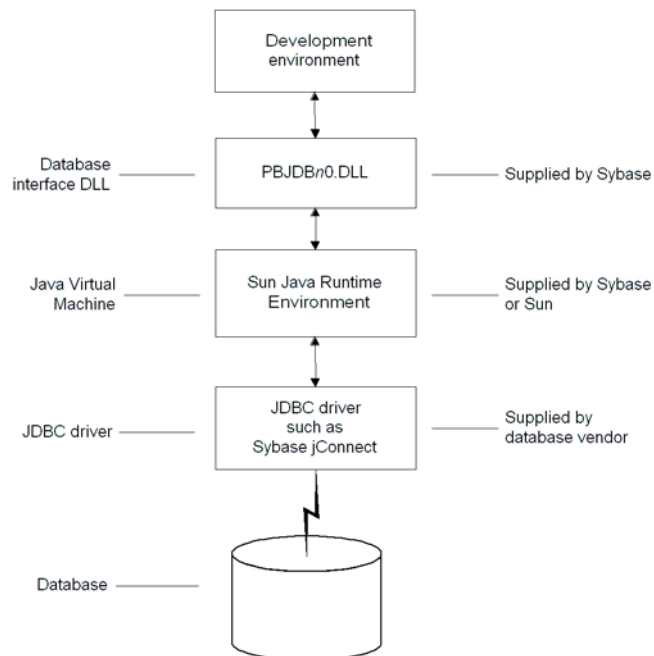
How a JDBC connection is made

In PowerBuilder when you access a database through the JDBC interface, your connection goes through several layers before reaching the database. It is important to understand that each layer represents a separate component of the connection, and that each component may come from a different vendor.

Because JDBC is a standard API, PowerBuilder uses the same interface to access every JDBC-compliant database driver.

Figure 3-1 shows the general components of a JDBC connection.

**Figure 3-1: Components of a JDBC connection**



The JDBC DLL

PowerBuilder provides the *pbjdb90.dll*. This DLL runs with the Sun Java Runtime Environment (JRE) versions 1.1 and later and the Microsoft Java virtual machine.

PowerBuilder Java package

PowerBuilder includes a small package of Java classes that gives the JDBC interface the level of error-checking and efficiency (SQLException catching) found in other PowerBuilder interfaces. The package is called *pbjdbc1290.jar* and is found in *Sybase\Shared\PowerBuilder*.

The Java Virtual Machine

The Java Virtual Machine (JVM) is a component of Java development software. When you install PowerBuilder, the Sun Java Development Kit (JDK), including the Java Runtime Environment (JRE), is installed on your system in *Sybase\Shared\PowerBuilder*. For PowerBuilder 9.0, JDK 1.4 is installed. This version of the JVM is started when you use a JDBC connection or any other process that requires a JVM and is used throughout the PowerBuilder session.

If you need to use a different JVM, see the instructions in “Preparing to use the JDBC interface” on page 43. For more information about how the JVM is started, see the chapter on deploying your application in *Application Techniques*.

The JDBC drivers

The JDBC interface can communicate with any JDBC-compliant driver including Sybase jConnect™ for JDBC and the Oracle and IBM Informix JDBC drivers. These drivers are native-protocol, all-Java drivers—that is, they convert JDBC calls into the SQL syntax supported by the databases.

## JDBC registry entries

When you access data through the PowerBuilder JDBC interface, PowerBuilder uses an internal registry to maintain definitions of SQL syntax, DBMS-specific function calls, and default DBParm parameter settings for the back-end DBMS. This internal registry currently includes subentries for Adaptive Server Anywhere, Adaptive Server Enterprise, and Oracle databases.

In most cases you do not need to modify the JDBC entries. However, if you do need to customize the existing entries or add new entries, you can make changes to the system registry by editing the registry directly or executing a registry file. Changes you introduce in the system registry override the PowerBuilder internal registry entries. See the *egreg.txt* file in *Sybase\Shared\PowerBuilder* for an example of a registry file you could execute to change entry settings.

## Supported versions for JDBC

The PowerBuilder JDBC interface uses the *pbjdb90.dll* to access a database through a JDBC driver.

To use the JDBC interface to access the jConnect driver, use jConnect Version 4.2 or higher or jConnect Version 5.2 or higher. For information on jConnect, see your Sybase documentation.

To use the JDBC interface to access the Oracle JDBC driver, use Oracle 8 JDBC driver Version 8.0.4 or higher. For information on the Oracle JDBC driver, see your Oracle documentation.

## Supported JDBC datatypes

Like ODBC, the JDBC interface compiles, sorts, presents, and uses a list of datatypes that are native to the back-end database to emulate as much as possible the behavior of a native interface.

## Preparing to use the JDBC interface

Before you define the interface and connect to a database through the JDBC interface, follow these steps to prepare the database for use:

- 1 Configure the database server for its JDBC connection and install its JDBC-compliant driver and network software.
- 2 Install the JDBC driver.
- 3 Set or verify the settings in the CLASSPATH environment variable and the Java tab of the System Options dialog box.

Step 1: Configure the database server

You must configure the database server to make JDBC connections as well as install the JDBC driver and network software.

❖ **To configure the database server for its JDBC connection:**

- 1 Make sure the database server is configured to make JDBC connections. For configuration instructions, see your database vendor's documentation.

- 2 Make sure the appropriate JDBC driver software is installed and running on the database server.

The driver vendor's documentation should provide the driver name, URL format, and any driver-specific properties you need to specify in the database profile. For notes about the jConnect driver, see "Configuring the jConnect driver" on page 44.

- 3 Make sure the required network software (such as TCP/IP) is installed and running on your computer and is properly configured so that you can connect to the database server at your site.

You must install the network communication driver that supports the network protocol and operating system platform you are using.

For installation and configuration instructions, see your network or database administrator.

#### Step 2: Install the JDBC driver

In the PowerBuilder Setup program, select the Typical install, or select the Custom install and select the JDBC driver.

#### Step 3: Set or verify the settings in the CLASSPATH variable and Java tab

Set or verify that the settings in the PATH and CLASSPATH environment variables or the Classpaths list on the Java tab of the PowerBuilder System Options dialog box point to the appropriate, fully-qualified file names.

If you are using the JDK installed with PowerBuilder, you do not need to make any changes to these environment variables.

If you are using JDK 1.2 or 1.3, you do not need to include any Sun Java VM packages in your CLASSPATH variable, but your PATH environment variable must include an entry for the Sun Java VM library, *jvm.dll* (for example, *path\JDK122\JRE\bin\classic*).

#### Configuring the jConnect driver

If you are using the Sybase jConnect driver, make sure to complete the required configuration steps such as installing the JDBC stored procedures in Adaptive Server databases. Also, verify that the CLASSPATH environment variable on your machine or the Classpaths list on the Java tab of the PowerBuilder System Options dialog box includes an entry pointing to the location of the jConnect driver. For example, if you are using jConnect 5.5, you should include an entry similar to the following:

```
C:\Program Files\Sybase\Shared\jConnect-5_5\classes\jconn2.jar
```

For more information about configuring jConnect, see the jConnect for JDBC documentation.

## Defining the JDBC interface

### Defining the profile

To define a connection through the JDBC interface, you must create a database profile by supplying values for at least the basic connection parameters in the Database Profile Setup - JDBC dialog box. You can then select this profile at any time to connect to your database in the development environment.

For information on how to define a database profile, see “Using database profiles” on page 8.

### Specifying connection parameters

To provide maximum flexibility (as provided in the JDBC API), the JDBC interface supports database connections made with different combinations of connection parameters:

- **Driver name, URL, and Properties** You should specify values for this combination of connection parameters if you need to define driver-specific properties. When properties are defined, you *must* also define the user ID and password in the properties field.

For example, when connecting to the jConnect driver, the following values should be entered in the Driver-Specific Properties field:

```
SQLINITSTRING=set TextSize 32000;
user=system;password=manager;
```

- **Driver name, URL, User ID, and Password** You should specify values for this combination of connection parameters if you do not need to define any driver-specific properties.

```
Driver Name: com.sybase.jdbc.SybDriver
URL:         jdbc:sybase:Tds:localhost:2638
Login ID:    dba
Password:    sql
```

- **Driver name and URL** You should specify values for this combination of connections parameters when the user ID and password are included as part of the URL.

For example, when connecting to the Oracle JDBC driver, the URL can include the user ID and password:

```
jdbc:oracle:thin:userid/password@host:port:dbname
```

Specifying the appropriate Java Virtual Machine (JVM)

---

**Specifying properties when connecting to jConnect**

If you plan to use the blob datatype in PowerBuilder, you should be aware that jConnect imposes a restriction on blob size. Consequently, before you make your database connection from PowerBuilder, you may want to reset the blob size to a value greater than the maximum size you plan to use.

To set blob size, define the jConnect property SQLINITSTRING in the Driver-Specific Properties box on the Connection page. The SQLINITSTRING property is used to define commands to be passed to the back-end database server:

```
SQLINITSTRING=set TextSize 32000;
```

Remember that if you define a property in the Driver-Specific Properties box, you must also define the user ID and password in this box.

---

Since the JDB interface supports several JVMs, you must specify which version of the JVM you want to use. For consistent behavior, the same version of the JVM used during development should be used at runtime.

Set the JavaVM DBParm on the Options tab page to select the appropriate JVM. The default value is JRE 1.4. Table 3-1 lists the supported JVMs and their corresponding JavaVM DBParm value.

**Table 3-1: Available Java VMs and JavaVM DBParm values**

JVM	DBParm Value
Sun JRE 1.2	Sun1.2
Sun JRE 1.3	Sun1.3
Sun JRE 1.4	Sun1.4

You do not need to set this DBParm for a PowerBuilder component running in EAServer.

---

**Selecting the JVM for a component deployed to EAServer**

If a PowerBuilder component running in EAServer makes a database connection using JDBC, the JDB interface verifies that the JVM used by EAServer matches the JVM selected in the PowerBuilder database profile. If the versions do not match, the JDB interface overrides the profile setting and uses the EAServer JVM. It also enters a warning in the EAServer log file. (The EAServer log file records errors relating to component execution. You can view its contents using the Jaguar Manager File Viewer.)

---



# Using the OLE DB interface

About this chapter

This chapter describes the OLE DB interface and then explains how to prepare to use this interface and how to define the OLE DB database profile.

Contents

Topic	Page
About the OLE DB interface	47
Preparing to use the OLE DB interface	51
Defining the OLE DB interface	53

For more information

This chapter gives general information about using the OLE DB interface. For more detailed information:

- See Microsoft's Universal Data Access Web site at [www.microsoft.com/data](http://www.microsoft.com/data).
- Use the online Help provided by the data provider vendor.
- Check to see if there is a technical document that describes how to connect to your OLE DB data provider. Any updated information about connectivity issues is available from the Sybase Customer Service and Support Web site at <http://support.sybase.com>.

## About the OLE DB interface

You can access a wide variety of data through OLE DB data providers in PowerBuilder. This section describes what you need to know to use OLE DB connections to access your data in PowerBuilder.

---

### Supported OLE DB data providers

For a complete list of the OLE DB data providers supplied with PowerBuilder and the data they access, see "Supported Database Interfaces" in online Help.

---

## What is OLE DB?

### OLE DB API

OLE DB is a standard application programming interface (API) developed by Microsoft. It is a component of Microsoft's Data Access Components software. OLE DB allows an application to access a variety of data for which OLE DB data providers exist. It provides an application with uniform access to data stored in diverse formats, such as indexed-sequential files like Btrieve, personal databases like Corel Paradox, productivity tools such as spreadsheets and electronic mail, and SQL-based DBMSs.

The OLE DB interface supports direct connections to SQL-based databases.

### Accessing data through OLE DB

Applications like PowerBuilder that provide an OLE DB interface can access data for which an OLE DB data provider exists. An **OLE DB data provider** is a dynamic link library (DLL) that implements OLE DB function calls to access a particular data source.

The PowerBuilder OLE DB interface can connect to any OLE DB data provider that supports the OLE DB object interfaces listed in Table 4-1. An OLE DB data provider must support these interfaces in order to adhere to the Microsoft OLE DB 2.0 specification.

**Table 4-1: Required OLE DB interfaces**

IAccessor	IDBInitialize
IColumnsInfo	IDBProperties
ICommand	IOpenRowset
ICommandProperties	IRowset
ICommandText	IRowsetInfo
IDBCreateCommand	IDBSchemaRowset
IDBCreateSession	ISourcesRowset

In addition to the required OLE DB interfaces, PowerBuilder also uses the OLE DB interfaces listed in Table 4-2 to provide further functionality.

**Table 4-2: Additional OLE DB interfaces**

OLE DB interface	Use in PowerBuilder
ICommandPrepare	Preparing commands and retrieving column information.
IDBInfo	Querying the data provider for its properties. If this interface is not supported, database connections may fail.
IDBCommandWithParameters	Querying the data provider for parameters.
IErrorInfo	Providing error information.
IErrorRecords	Providing error information.

OLE DB interface	Use in PowerBuilder
IIndexDefinition	Creating indexes for the extended attribute system tables. Also creating indexes in the Database painter. If this interface is not supported, PowerBuilder looks for index definition syntax in the <i>pbodb90.ini</i> file.
IMultipleResults	Providing information.
IRowsetChange	Populating the extended attribute system tables when they are created. Also, for updating blobs.
IRowsetUpdate	Creating the extended attribute system tables.
ISQLErrorInfo	Providing error information.
ISupportErrorInfo	Providing error information.
ITableDefinition	Creating the extended attribute system tables and also for creating tables in the Database painter. If this interface is not supported, the following behavior results: <ul style="list-style-type: none"><li>• PowerBuilder looks for table definition syntax in the <i>pbodb90.ini</i> file</li><li>• PowerBuilder catalog tables cannot be used</li><li>• DDL and DML operations, like modifying columns or editing data in the database painter, do not function properly</li></ul>
ITransactionLocal	Supporting transactions. If this interface is not supported, PowerBuilder defaults to AutoCommit mode.

---

**For up-to-date information**

As connectivity to data through additional OLE DB data providers is supported in future releases of PowerBuilder, updated information will be available on the Sybase Web site. Check the Release Bulletin for the latest information about supported OLE DB data providers for your version of PowerBuilder.

---

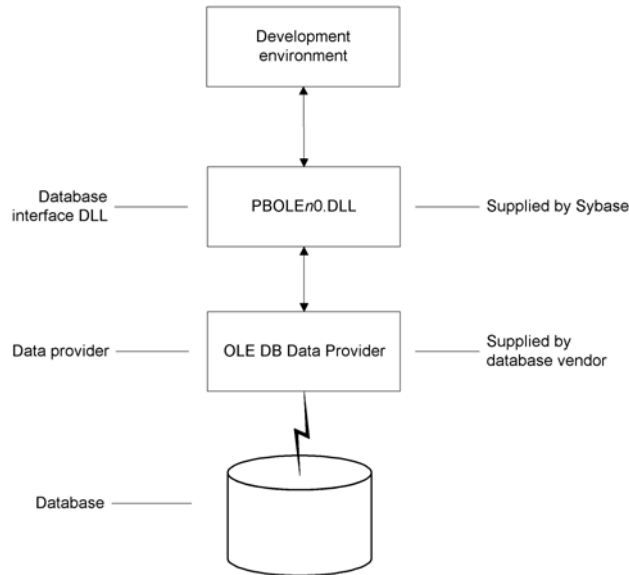
## Components of an OLE DB connection

When you access an OLE DB data provider in PowerBuilder, your connection goes through several layers before reaching the data provider. It is important to understand that each layer represents a separate component of the connection, and that each component may come from a different vendor.

Because OLE DB is a standard API, PowerBuilder uses the same interface to access every OLE DB data provider. As long as an OLE DB data provider supports the object interfaces required by PowerBuilder, PowerBuilder can access it through the OLE DB interface.

Figure 4-1 shows the general components of a OLE DB connection.

**Figure 4-1: Components of an OLE DB connection**



## Obtaining OLE DB data providers

There are two ways you can obtain OLE DB data providers for use with PowerBuilder:

- **From Sybase (recommended)** Install the OLE DB data providers shipped with PowerBuilder. You can do this either when you first install PowerBuilder or later.
- **From another vendor** PowerBuilder Enterprise lets you access data with *any* OLE DB data provider obtained from a vendor other than Sybase if that data provider supports the OLE DB object interfaces required by PowerBuilder. In most cases, these drivers work with PowerBuilder. However, Sybase may not have tested the drivers to verify this.

---

**PowerBuilder Professional and Desktop editions**

The PowerBuilder Professional and Desktop editions do *not* support the OLE DB interface.

---

## Supported versions for OLE DB

The OLE DB interface uses a DLL named *PBOLE90.DLL* to access a database through an OLE DB data provider.

---

**Required OLE DB version**

To use the OLE DB interface to access an OLE DB database, you must connect through an OLE DB data provider that supports OLE DB version 2.0 or later. For information on OLE DB specifications, see Microsoft's Universal Data Access Web site at [www.microsoft.com/data](http://www.microsoft.com/data).

---

## Preparing to use the OLE DB interface

Before you define the interface and connect to a data provider through the OLE DB:

- 1 Install and configure the database server, network, and client software.
- 2 Install the OLE DB interface and the OLE DB data provider that accesses your data source.
- 3 Install Microsoft's Data Access Components software on your machine.
- 4 If required, define the OLE DB data source.

Step 1: Install and  
configure the data  
server

You must install and configure the database server and install the network software and client software.

❖ **To install and configure the database server, network, and client software:**

- 1 Make sure the appropriate database software is installed and running on its server.

You must obtain the database server software from your database vendor. For installation instructions, see your database vendor's documentation.

- 2 Make sure the required network software (such as TCP/IP) is installed and running on your computer and is properly configured so that you can connect to the data server at your site. You must install the network communication driver that supports the network protocol and operating system platform you are using.

For installation and configuration instructions, see your network or data source administrator.

- 3 If required, install the appropriate client software on each client computer on which PowerBuilder is installed.

---

**Client software requirements**

To determine client software requirements, see your database vendor's documentation. To access supported remote Informix databases through the Informix data provider, you need Informix Connect for Windows platforms, version 2.x, or the Informix Client Software Development Kit for Windows platforms, version 2.x.

---

Step 2: Install the OLE DB interface and data provider

In the PowerBuilder Setup program, select the Custom install and select the OLE DB provider that accesses your database. You can install one or more of the OLE DB data providers shipped with PowerBuilder, or you can install data providers from another vendor later.

Step 3: Install the Microsoft Data Access Components software

The PowerBuilder OLE DB interface requires the functionality of the Microsoft Data Access Components software (MDAC). The installation of MDAC is not part of the PowerBuilder installation. You must perform this installation separately by executing the file *mdac\_typ.exe* found in the *Support* directory.

---

**OLE DB data providers installed with Microsoft Data Access Components**

When you execute the *mdac\_typ* file, several Microsoft OLE DB data providers are automatically installed, including the following:

Microsoft OLE DB Provider for ODBC (MSDASQL)  
Microsoft OLE DB Provider for SQL Server (SQLOLEDB)

---

Step 4: Define the OLE DB data source

Once the OLE DB data provider is installed, you may have to define the OLE DB data source the data provider will access. How you define the data source depends on the OLE DB data provider you are using and the vendor who provided it.

To define a data source for one of the OLE DB data providers shipped with PowerBuilder, use the DataDirect OLE DB Administrator. This utility is named PBadmin and can be found in *Sybase\Shared\DataDirect*.

If you are connecting to an ODBC data provider (such as Microsoft's OLE DB Provider for ODBC), you must define the ODBC data source as you would if you were using a direct ODBC connection. To define an ODBC data source, use Microsoft's ODBC Data Source Administrator. This utility can be accessed through the Control Panel in Windows or through the Database painter.

## Defining the OLE DB interface

### Using the OLE DB Database Profile Setup

To define a connection through the OLE DB interface, you must create a database profile by supplying values for at least the basic connection parameters in the Database Profile Setup – OLE DB dialog box. You can then select this profile anytime to connect to your data in the development environment.

For information on how to define a database profile, see “Using database profiles” on page 8.

### Specifying connection parameters

You must supply values for the Provider and Data Source connection parameters. Select a data provider from the list of installed data providers in the Provider drop-down list. The Data Source value varies depending on the type of data source connection you are making. For example:

- If you are using Microsoft's OLE DB Provider for ODBC to connect to the EAS Demo DB, you select MSDASQL as the Provider value and enter the actual ODBC data source name (in this case EAS Demo DB) as the Data Source value.
- If you are using Microsoft's OLE DB Provider for SQL Server, you select SQLOLEDB as the Provider value and enter the actual server name as the Data Source value. In the case of Microsoft SQL Server, you must also use the Extended Properties field to provide the database name (for example, Database=Pubs) since you can have multiple instances of a database.
- If you are using the PB OLE DB Provider to connect to an Oracle8i database, you select PB Oracle8 OLE DB Provider as the Provider value and enter the actual data source name (which you should have previously defined using the DataDirect OLE DB Administrator) as the Data Source value.

Using the Data Link API

The Data Link option allows you to access Microsoft's Data Link API, which allows you to define a file or use an existing file that contains your OLE DB connection information. A Data Link file is identified with the suffix *.udl*.

To launch this option, select the File Name check box on the Connection tab and double-click on the button next to the File Name box. (You can also launch the Data Link API in the Database painter by double-clicking on the Manage Data Links utility included with the OLE DB interface in the list of Installed Database Interfaces.)

For more information on using the Data Link API, see Microsoft's Universal Data Access Web site at [www.microsoft.com/data](http://www.microsoft.com/data).

---

**Using a Data Link file versus setting the DBParm parameters**

If you use a Data Link file to connect to your data source, all other settings you make in the OLE DB Database Profile Setup dialog box are ignored.

---



## PART 3

# Working with Native Database Interfaces

This part describes how to set up and define database connections accessed through one of the native database interfaces.



# Using Native Database Interfaces

## About this chapter

This chapter describes the native database interfaces. For each supported interface, it then explains how to prepare to use the database and define any unique database interface parameters so that you can access your data.

## Contents

Topic	Page
About native database interfaces	58
Informix	61
Microsoft SQL Server	68
Oracle	72
Adaptive Server Enterprise	85
Installing PowerBuilder stored procedures in Adaptive Server databases	99
DirectConnect	105
Creating the extended attribute system tables in DB2 databases	114

## For more information

This chapter gives general information about using each native database interface. For more detailed information:

- Check to see if there is a technical document that describes how to connect to your database. Any updated information about connectivity issues is available from the Sybase Customer Service and Support Web site at <http://support.sybase.com>.
- Ask your network or system administrator for assistance when installing and setting up the database server and client software at your site.

## About native database interfaces

The native database interfaces provide native connections to many databases and DBMSs. This section describes how the native database interfaces access these databases.

You can upgrade to PowerBuilder Enterprise to use the native database interfaces.

For a complete list of the supported native database interfaces, see “Supported Database Interfaces” in online Help.

## What is a native database interface?

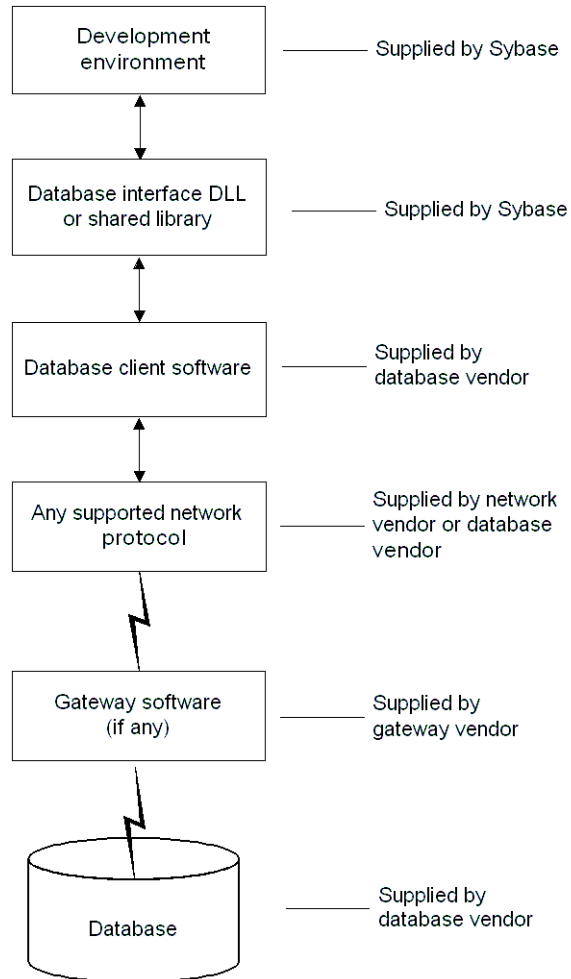
A native database interface is a direct connection to your data in PowerBuilder.

Each native database interface uses its own interface DLL to communicate with a specified database through a vendor-specific database API. For example, the Adaptive Server interface uses a DLL named `PBSYCn0.DLL` to access the database, while the Oracle 8.0.4 database interface accesses the database through `PBO84n0.DLL`.

In contrast, a standard database interface uses a standard API to communicate with the database. For example, PowerBuilder can use a single-interface DLL to communicate with the ODBC Driver Manager and corresponding driver to access any ODBC data source.

## Components of a database interface connection

When you use a native database interface to access a database, your connection goes through several layers before reaching the data. Each layer is a separate component of the connection and each component may come from a different vendor.

**Figure 5-1: Components of a database connection**

For diagrams showing the specific components of your connection, see “Basic software components” in the section in this chapter for your native database interface.

## Using a native database interface

About preparing to use the database

You perform several basic steps to use a native database interface to access a database.

The first step in connecting to a database through a native database interface is to prepare to use the database. Preparing the database ensures that you will be able to access and use your data in PowerBuilder.

You must prepare the database *outside* PowerBuilder *before* you start the product, then define the database interface and connect to it. The requirements differ for each database—but in general, preparing a database involves four basic steps.

### ❖ To prepare to use your database with PowerBuilder:

- 1 Make sure the required database server software is properly installed and configured at your site.
- 2 If network software is required, make sure it is properly installed and configured at your site and on the client computer so that you can connect to the database server.
- 3 Make sure the required database client software is properly installed and configured on the client computer. (Typically, the client computer is the one running PowerBuilder.)

You must obtain the client software from your database vendor and make sure that the version you install supports *all* of the following:

- The operating system running on the client computer
- The version of the database that you want to access
- The version of PowerBuilder that you are running

- 4 Verify that you can connect to the server and database you want to access outside PowerBuilder.

For specific instructions to use with your database, see "Preparing to use the database" in the section in this chapter for your native database interface.

About installing native database interfaces

After you prepare to use the database, you must install the native database interface that accesses the database. See the instructions for each interface for more information.

About defining native database interfaces

Once you are ready to access the database, you start PowerBuilder and define the database interface. To define a database interface, you must create a database profile by completing the Database Profile Setup dialog box for that interface.

For general instructions, see “About creating database profiles” on page 8. For instructions about defining database interface parameters unique to a particular database, see “Preparing to use the database” in the section in this chapter for your database interface.

What to do next

For instructions on how to prepare the database and define the database interface you are using, see the section in this chapter for your native database interface.

## Informix

This section describes how to use the native IBM Informix database interface in PowerBuilder.

### Supported versions for Informix

You can access the following Informix databases using the native Informix database interface:

- Informix Dynamic Server
- Informix-OnLine and Informix-SE version 9.x

PowerBuilder provides the IN9 interface in the *PBIN990.DLL* to connect through Informix-Connect version 9.x client software.

## Supported Informix datatypes

The Informix database interface supports the Informix datatypes listed in Table 5-1 in DataWindow objects and embedded SQL.

**Table 5-1: Supported datatypes for Informix**

Byte (a maximum of 231 bytes)	Integer (4 bytes)
Character (1 to 32,511 bytes)	Money
Date	Real
DateTime	Serial
Decimal	SmallInt (2 bytes)
Float	Text (a maximum of 231 bytes)
Interval	VarChar (1 to 255 bytes)

---

### Exceptions

Byte, text, and VarChar datatypes are not supported in Informix SE.

---

---

### Datatype conversion

When you retrieve or update columns, PowerBuilder converts data appropriately between the Informix datatype and the PowerScript datatype. Keep in mind, however, that similarly or identically named Informix and PowerScript datatypes do *not* necessarily have the same definitions.

For information about the definitions of PowerScript datatypes, see the *PowerScript Reference*.

---

## Informix DateTime datatype

The DateTime datatype is a contiguous sequence of boxes. Each box represents a component of time that you want to record. The syntax is:

**DATETIME** *largest\_qualifier* **TO** *smallest\_qualifier*

PowerBuilder defaults to Year TO Fraction(5).

For a list of qualifiers, see your Informix documentation.



❖ **To create your own variation of the DateTime datatype:**

- 1 In the Database painter, create a table with a DateTime column.  
For instructions on creating a table, see the *User's Guide*.
- 2 In the Columns view, select Pending Syntax from the Objects or pop-up menu.  
  
The Columns view displays the pending changes to the table definition. These changes execute only when you click the Save button to save the table definition.
- 3 Select Copy from the Edit or pop-up menu.  
*or*  
Click the Copy button.  
  
The SQL syntax (or the portion you selected) is copied to the clipboard.
- 4 In the ISQL view, modify the DateTime syntax and execute the CREATE TABLE statement.  
  
For instructions on using the ISQL view, see the *User's Guide*.

## Informix Time datatype

The Informix database interfaces also support a time datatype. The time datatype is a subset of the DateTime datatype. The time datatype uses only the time qualifier boxes.

## Informix Interval datatype

The interval datatype is one value or a sequence of values that represent a component of time. The syntax is:

**INTERVAL** *largest\_qualifier* **TO** *smallest\_qualifier*

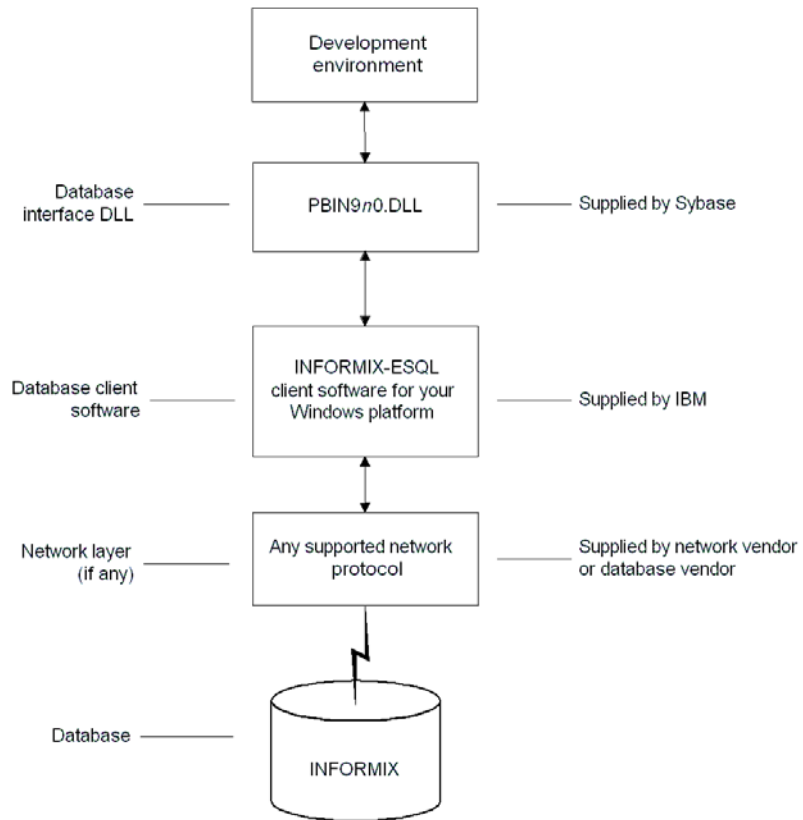
PowerBuilder defaults to Day(3) TO Day.

For more about interval datatypes, see your Informix documentation.

## Basic software components for Informix

Figure 5-2 shows the basic software components required to access an Informix database using the native Informix database interfaces.

**Figure 5-2: Components of an Informix connection**



## Preparing to use the Informix database

Before you define the database interface and connect to an Informix database in PowerBuilder, follow these steps to prepare the database for use:

- 1 Install and configure the required database server, network, and client software.
- 2 Install the native Informix IN9 database interface.

Step 1: Install and  
configure the  
database server

- 3 Verify that you can connect to the Informix server and database outside PowerBuilder.

You must install and configure the required database server, network, and client software for Informix.

❖ **To install and configure the required database server, network, and client software:**

- 1 Make sure the Informix database server software and database network software is installed and running on the server specified in your database profile.

You must obtain the database server and database network software from Informix.

For installation instructions, see your Informix documentation.

- 2 Install the required Informix client software on each client computer on which PowerBuilder is installed.

Install Informix Connect or the Informix Client SDK (which includes Informix Connect) and run the SetNet32 utility to configure the client registry settings.

You must obtain the Informix client software from IBM. Make sure the version of the client software you install supports *all* of the following:

- The operating system running on the client computer
- The version of the database that you want to access
- The version of PowerBuilder that you are running

For installation instructions, see your Informix documentation.

- 3 Make sure the Informix client software is properly configured so that you can connect to the Informix database server at your site.

For example, when you install Informix-Connect client software, it automatically creates the correct configuration file on your computer.

The configuration file contains default parameters that define your network configuration, network protocol, and environment variables. If you omit these values from the database profile when you define the native Informix database interface, they default to the values specified in your configuration file.

For instructions on setting up the Informix configuration file, see your Informix documentation.

- 4 If required by your operating system, make sure the directory containing the Informix client software is in your system path.

Step 2: Install the database interface

In the PowerBuilder Setup program, select the Typical install, or select the native Informix database interface in the Custom install.

Step 3: Verify the connection

Make sure you can connect to the Informix server and database you want to access from outside PowerBuilder.

To verify the connection, use any Windows-based utility (such as the Informix *ILOGIN.EXE* program) that connects to the database. When connecting, be sure to specify the same parameters you plan to use in your PowerBuilder database profile to access the database.

For instructions on using *ILOGIN.EXE*, see your Informix documentation.

## Defining the Informix database interface

To define a connection through an Informix database interface, you must create a database profile by supplying values for at least the basic connection parameters in the Database Profile Setup - Informix IN9 dialog box. You can then select this profile at any time to connect to your database in the development environment.

For information on how to define a database profile, see “Using database profiles” on page 8.

## Specifying the server name

When you specify the server name value, you *must* use the following format to connect to the database through the Informix interface:

*host\_name@server\_name*

Parameter	Description
<i>host_name</i>	The name of the host computer running the Informix database server. This corresponds to the Informix HOSTNAME environment variable.
<i>server_name</i>	The name of the server containing the Informix database. This corresponds to the Informix SERVER environment variable.

For example, to use the IN9 interface to connect to an Informix database server named `server01` running on a host machine named `sales`, do either of the following:

- **In a database profile** Type the host name (`sales`) in the Host Name box and the server name (`server01`) in the Server box on the Connection tab in the Database Profile Setup - Informix IN9 dialog box. PowerBuilder saves this server name as `sales@server01` in the database profile entry in the system registry.
- **In a PowerBuilder script** Type the following in your PowerBuilder application script:

```
SQLCA.ServerName = "sales@server01"
```

---

**Tip**

If you specify a value for Host Name and Server in your database profile, this syntax displays on the Preview tab in the Database Profile Setup - Informix IN9 dialog box. You can then copy the syntax from the Preview tab into your script.

---

## Accessing serial values in a PowerBuilder script

If you are connecting to an Informix database from a PowerBuilder script, you can obtain the serial number of the row inserted into an Informix table by checking the value of the `SQLReturnData` property of the `Transaction` object.

After an embedded SQL `INSERT` statement executes, `SQLReturnData` contains the serial number that uniquely identifies the row inserted into the table.

PowerBuilder updates `SQLReturnData` following an embedded SQL statement only; it does not update it following a `DataWindow` operation.

## What to do next

For instructions on connecting to the database, see “Connecting to a database” on page 121.

# Microsoft SQL Server

This section describes how to use the Microsoft SQL Server database interface in PowerBuilder.

## Supported versions for SQL Server

You can access Microsoft SQL Server version 6.0, 6.5, 7.0 and SQL Server 2000 databases using the Microsoft SQL Server interface. The SQL Server interface uses a DLL named *PBMSS90.DLL* to access the database.

For more information about supported interfaces for your product, see "Supported Database Interfaces" in online Help.

## Supported SQL Server datatypes

The Microsoft SQL Server database interface supports the datatypes listed in Table 5-2.

**Table 5-2: Supported datatypes for Microsoft SQL Server**

Binary	Numeric
Bit	Real
Character (fewer than 255 characters)	SmallDateTime
DateTime	SmallInt
Decimal	SmallMoney
Float	Text
Identity	Timestamp
Image	TinyInt
Int	VarBinary
Money	VarChar

---

### Datatype conversion

When you retrieve or update columns, PowerBuilder converts data appropriately between the Microsoft SQL Server datatype and the PowerScript datatype. Keep in mind, however, that similarly or identically named SQL Server and PowerScript datatypes do *not* necessarily have the same definitions.

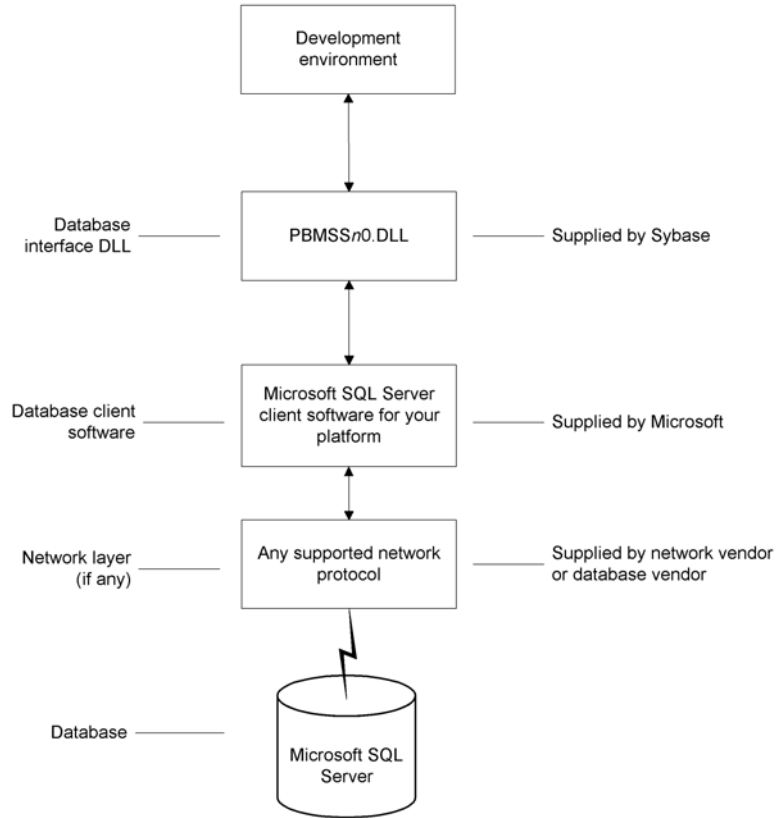
For information about the definitions of PowerScript datatypes, see the PowerScript Reference.

---

## Basic software components for SQL Server

You must install the software components in Figure 5-3 to access a database with the Microsoft SQL Server interface.

**Figure 5-3: Components of a Microsoft SQL Server connection**



## Preparing to use the SQL Server database

Before you define the database interface and connect to a Microsoft SQL Server database in PowerBuilder, follow these steps to prepare the database for use:

- 1 Install and configure the required database server, network, and client software.
- 2 Install the SQL Server database interface.

Step 1: Install and  
configure the  
database server

- 3 Verify that you can connect to the Microsoft SQL Server server and database outside PowerBuilder.

You must install and configure the database server, network, and client software for SQL Server.

❖ **To install and configure the database server, network, and client software:**

- 1 Make sure the Microsoft SQL Server database software is installed and running on the server specified in your database profile.

You must obtain the database server software and required licenses from Microsoft Corporation. For installation instructions, see your Microsoft SQL Server documentation.

---

**Upgrading from an earlier version of SQL Server**

For instructions on upgrading to a later version of SQL Server or installing it alongside an earlier version, see your Microsoft SQL Server documentation.

---

- 2 If you are accessing a remote SQL Server database, make sure the required network software (for example, TCP/IP) is installed and running on your computer and is properly configured so that you can connect to the SQL Server database server at your site.

For installation and configuration instructions, see your network or database administrator.

- 3 Install the required Microsoft SQL Server client software on each client computer on which PowerBuilder is installed.

You must obtain the SQL Server client software from Microsoft Corporation. Make sure the version of the client software you install supports *all* of the following:

- The operating system running on the client computer
- The version of the database that you want to access
- The version of PowerBuilder that you are running

For installation instructions, see your Microsoft SQL Server documentation.

- 4 Make sure the SQL Server client software is properly configured so that you can connect to the SQL Server database server at your site.



Once you install the SQL Server client software, you can configure optional client connection parameters by using the SQL Server Client Configuration Utility that comes with the client software.

For configuration instructions, see your Microsoft SQL Server documentation.

- 5 If required by your operating system, make sure the directory containing the SQL Server client software is in your system path.
- 6 Make sure only one copy of each of the following files is installed on your computer:
  - *DBMSSOCN.DLL*
  - *NTWDBLIB.DLL* (32-bit SQL Server DLL)

Step 2: Install the database interface

In the PowerBuilder Setup program, select the Typical install, or select the Custom install and select the native SQL Server database interface.

Step 3: Verify the connection

Make sure you can connect to the SQL Server server and database you want to access from outside PowerBuilder.

To verify the connection, use any Windows-based utility that connects to the database. When connecting, be sure to specify the same parameters you plan to use in your PowerBuilder database profile to access the database.

## Defining the SQL Server database interface

To define a connection through the Microsoft SQL Server interface, you must create a database profile by supplying values for at least the basic connection parameters in the Database Profile Setup - Microsoft SQL Server dialog box. You can then select this profile at any time to connect to your database in the development environment.

For information on how to define a database profile, see “Using database profiles” on page 8.

## What to do next

For instructions on connecting to the database, see “Connecting to a database” on page 121.

# Oracle

This section describes how to use the native Oracle database interfaces in PowerBuilder.

## Supported versions for Oracle

PowerBuilder provides three Oracle database interfaces. These interfaces use different DLLs and access different versions of Oracle.

**Table 5-3: Supported native database interfaces for Oracle**

Oracle interface	DLL
O73 Oracle 7.3	PBO7390.DLL
O84 Oracle 8.0.x and Oracle8i	PBO8490.DLL
O90 Oracle9i	PBO9090.DLL

---

**For more information**

Oracle has announced that it will no longer support Oracle 7.3 as of the end of 2002. Therefore, this will be the last version of PowerBuilder that will provide support for this interface. Updated information about supported versions of databases may be available electronically on the Sybase Customer Service and Support Web site at <http://support.sybase.com> or in the PowerBuilder Release Bulletin.

---

## Supported Oracle datatypes

The Oracle database interfaces support the Oracle datatypes listed in Table 5-4 in DataWindow objects and embedded SQL:

**Table 5-4: Supported datatypes for Oracle**

Bfile (Oracle 8 and later only)	NChar (Oracle9i only)
Blob (Oracle 8 and later only)	Number
Char	NVarChar2 (Oracle9i only)
Clob (Oracle 8 and later only)	Raw
Date	TimeStamp (Oracle9i only)
Float	VarChar
Long	VarChar2
LongRaw	

**NChar and  
NVarChar2 datatypes**

PowerBuilder supports the NChar and NVarChar2 datatypes when connecting to an Oracle9i server using the O90 interface. This support is not available for Oracle8i because the DataWindow object is unable to generate a prefix required when sending an ANSI string to an NChar or NVarChar2 column in a table on an Oracle8i server. The prefix instructs the server to convert data to a Unicode string. Oracle9i servers do not need to convert ANSI strings to Unicode strings.

For Oracle8i, a client statement might look like this:

```
Update table1 set C1 = N'NEW STRING' where C1 = N'OLD
STRING' ;
```

where the datatype of C1 is NChar or NVarChar2, and the prefix N instructs the Oracle8i server to convert the data to a Unicode string.

**TimeStamp datatype**

The Oracle9i TimeStamp datatype is an extension of the Date datatype. It stores the year, month, and day of the Date value plus hours, minutes, and seconds:

```
Timestamp[fractional_seconds_precision]
```

The *fractional\_seconds\_precision* value is optional and provides the number of digits for indicating seconds. The range of valid values for use with PowerBuilder is 0-6.

**Datatype conversion**

When you retrieve or update columns, in general PowerBuilder converts data appropriately between the Oracle datatype and the PowerScript datatype. Keep in mind, however, that similarly or identically named Oracle and PowerScript datatypes do *not* necessarily have the same definitions.

For information about the definitions of PowerScript datatypes, see the *PowerScript Reference*.

**Number datatype  
converted to decimal**

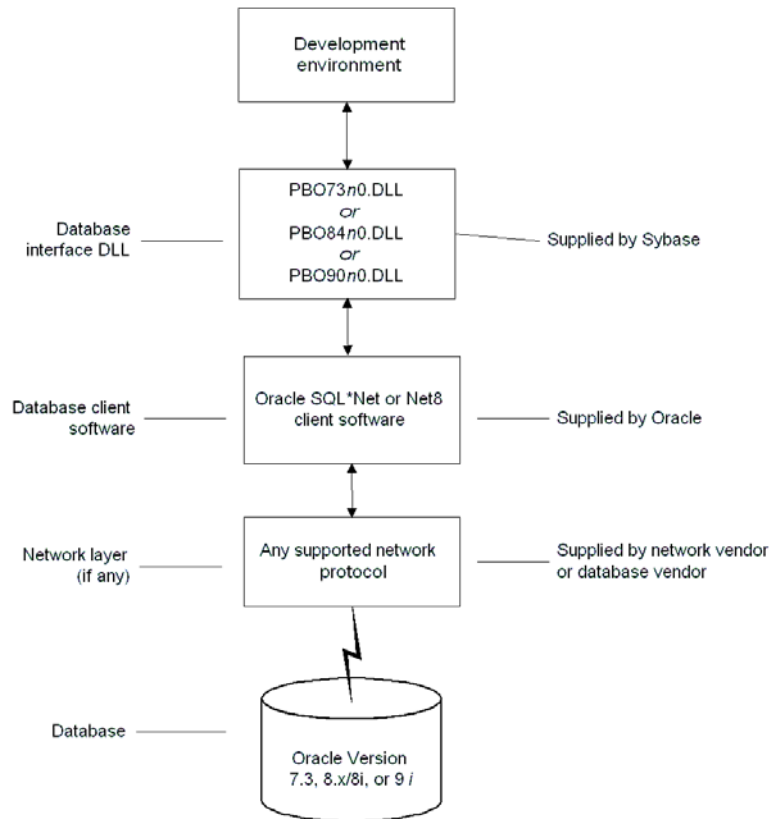
When a DataWindow object is defined in PowerBuilder, the Oracle datatype number(size,d) is mapped to a decimal datatype. In PowerBuilder, the precision of a decimal is 18 digits. If a column's datatype has a later precision, for example number(32,30), inserting a number with a precision greater than 18 digits produces an incorrect result when the number is retrieved in a DataWindow. For example, 1.8E-17 displays as 0.0000000000000000018, whereas 1.5E-25 displays as 0.

You may be able to avoid this problem by using a different datatype, such as float, for high precision number columns in the Oracle DBMS. The float datatype is mapped to the number datatype within the DataWindow's source.

## Basic software components for Oracle

You must install the software components in Figure 5-4 to access an Oracle database in PowerBuilder.

**Figure 5-4: Components of an Oracle connection**



## Preparing to use the Oracle database

Before you define the database interface and connect to an Oracle database in PowerBuilder, follow these steps to prepare the database for use:

- 1 Install and configure the required database server, network, and client software.
- 2 Install the native Oracle database interface for the version of Oracle you want to access.

**Step 1: Install and  
configure the  
database server**

- 3 Verify that you can connect to the Oracle server and database outside PowerBuilder.

Preparing an Oracle database for use with PowerBuilder involves these three basic tasks.

You must install and configure the database server, network, and client software for Oracle.

❖ **To install and configure the database server, network, and client software:**

- 1 Make sure the Oracle database software is installed on your computer or on the server specified in your database profile.

For example, with the Oracle O84 interface you can access an Oracle 8.0.x or Oracle8i database server.

You must obtain the database server software from Oracle Corporation.

For installation instructions, see your Oracle documentation.

- 2 Make sure the supported network software (such as TCP/IP) is installed and running on your computer and is properly configured so that you can connect to the Oracle database server at your site.

The Hosts and Services files must be present on your computer and properly configured for your environment.

You must obtain the network software from your network vendor or database vendor.

For installation and configuration instructions, see your network or database administrator.

- 3 Install the required Oracle client software on each client computer on which PowerBuilder is installed.

You must obtain the client software from Oracle Corporation. Make sure the client software version you install supports *all* of the following:

- The operating system running on the client computer
- The version of the database that you want to access
- The version of PowerBuilder that you are running

---

**Required client software versions**

To use the Oracle 7.3 (O73) interface, you must install Oracle SQL\*Net client software version 2.3 or later.

To use the Oracle 8.0.x and Oracle8i (O84) interface or the Oracle9i (O90) interface, you must install Oracle Net client software version 8.0.4 or later.

---

- 4 Make sure the Oracle SQL\*Net or Net client software is properly configured so that you can connect to the Oracle database server at your site.

Installing SQL\*Net software places the correct configuration file in the Oracle directory on your computer. For example, if you are using SQL\*Net version 2.x, the required configuration file is called *TNSNAMES.ORA*.

The configuration file provides information that Oracle needs to find and connect to the database server at your site. To modify and view the information in *TNSNAMES.ORA*, use an Oracle tool designed to edit the configuration file (such as Oracle Network Manager or the SQL\*Net Easy Configuration utility).

For information about setting up Oracle configuration files, see your SQL\*Net or Net documentation.

- 5 If required by your operating system, make sure the directory containing the Oracle client software is in your system path.

**Step 2: Install the database interface**

In the PowerBuilder Setup program, select the Typical install or select the Custom install and select the Oracle database interfaces you require. You must select the Custom install to install the Oracle 7.3 database interface.

For a list of the Oracle database interfaces available, see “Supported versions for Oracle” on page 72.

**Step 3: Verify the connection**

Make sure you can connect to the Oracle database server and log in to the database you want to access from outside PowerBuilder.

Some possible ways to verify the connection are by running the following Oracle tools:

- **Accessing the database server** Tools such as Oracle TNSPING (or any other ping utility) check whether you can reach the database server from your computer.

- **Accessing the database** Tools such as Oracle SQL\*Plus check whether you can log in to the Oracle database you want to access and perform database operations. It is a good idea to specify the same connection parameters you plan to use in your PowerBuilder database profile to access the database.

## What to do next

For instructions on defining the Oracle database interface in PowerBuilder, see “Defining the Oracle database interface” on page 77.

## Defining the Oracle database interface

To define a connection through an Oracle database interface, you must create a database profile by supplying values for at least the basic connection parameters in the Database Profile Setup dialog box for your Oracle interface. You can then select this profile at any time to connect to your database in the development environment.

For information on how to define a database profile, see “Using database profiles” on page 8.

## Specifying the Oracle server connect descriptor

To connect to an Oracle database server that resides on a network, you must specify the proper connect descriptor in the Server box on the Connection tab of the Database Profile Setup dialog box for your Oracle interface. The connect descriptor specifies the connection parameters that Oracle uses to access the database.

For help determining the proper connect descriptor for your environment, see your Oracle documentation or system administrator.

### Specifying a connect descriptor

The syntax of the connect descriptor depends on the Oracle client software you are using.

If you are using Net version 8.x or later, the syntax is:

*OracleServiceName*

If you are using SQL\*Net version 2.x, the syntax is:

@ **TNS:** *OracleServiceName*

Parameter	Description
@	The at ( @ ) sign is required
TNS	The identifier for the Oracle Transparent Network Substrate (TNS) technology
:	The colon ( : ) is required
<i>OracleServiceName</i>	The service name assigned to your server in the Oracle configuration file for your platform

**Net version 8.x example** To use Net version 8.x or later client software to connect to the service named ORA8, type the following connect descriptor in the Server box on the Connection tab of the Database Profile Setup dialog box for Oracle 8.x and later:ORA8.

## Using Oracle stored procedures as a data source

This section describes how you can use Oracle stored procedures.

### What is an Oracle stored procedure?

Oracle defines a **stored procedure** (or function) as a named PL/SQL program unit that logically groups a set of SQL and other PL/SQL programming language statements together to perform a specific task.

Stored procedures can take parameters and return one or more result sets (also called cursor variables). You create stored procedures in your schema and store them in the data dictionary for use by multiple users.

### What you can do with Oracle stored procedures

Ways to use Oracle stored procedures

You can use an Oracle stored procedure in the following ways in your PowerBuilder application:

- As a data source for DataWindow objects
- Called by an embedded SQL DECLARE PROCEDURE statement in a PowerBuilder application (includes support for fetching against stored procedures with result sets)
- Called as an external function or subroutine in a PowerBuilder application by using the RPCFUNC keyword when you declare the procedure



For information about the syntax for using the `DECLARE PROCEDURE` statement with the `RPCFUNC` keyword, see the *PowerScript Reference*.

**Procedures with a single result set** You can use stored procedures that return a single result set in DataWindow objects and embedded SQL, but *not* when using the `RPCFUNC` keyword to declare the stored procedure as an external function or subroutine.

**Procedures with multiple result sets** You can use procedures that return multiple result sets *only* in embedded SQL. Multiple result sets are *not supported* in DataWindows, reports, or with the `RPCFUNC` keyword.

## Using Oracle stored procedures with result sets

Overview of basic steps

The following procedure assumes you are creating the stored procedure in the ISQL view of the Database painter in PowerBuilder.

### ❖ To use an Oracle stored procedure with a result set:

- 1 Set up the ISQL view of the Database painter to create the stored procedure.
- 2 Create the stored procedure with a result set as an IN OUT (reference) parameter.
- 3 Create DataWindow objects that use the stored procedure as a data source.

Setting up the Database painter

When you create a stored procedure in the ISQL view of the Database painter, you must change the default SQL statement terminator character to one that you do not plan to use in your stored procedure syntax.

The default SQL terminator character for the Database painter is a semicolon (;). If you plan to use a semicolon in your Oracle stored procedure syntax, you must change the painter's terminator character to something other than a semicolon to avoid conflicts. A good choice is the backquote ( ` ) character.

### ❖ To change the default SQL terminator character in the Database painter:

- 1 Connect to your Oracle database in PowerBuilder as the System user.  
For instructions, see “Defining the Oracle database interface” on page 77.
- 2 Open the Database painter.
- 3 Select Design>Options from the menu bar.

The Database Preferences property sheet displays. If necessary, click the General tab to display the General property page.

## Creating the stored procedure

- 4 Type the character you want (for example, a backquote) in the SQL Terminator Character box.
- 5 Click Apply or OK.

The SQL Terminator Character setting is applied to the current connection and all future connections (until you change it).

After setting up the Database painter, you can create an Oracle stored procedure that has a result set as an IN OUT (reference) parameter. PowerBuilder retrieves the result set to populate a DataWindow object.

There are many ways to create stored procedures with result sets. The following procedure describes one possible method that you can use.

For information about when you can use stored procedures with single and multiple result sets, see “What you can do with Oracle stored procedures” on page 78.

❖ **To create Oracle stored procedures with result sets:**

- 1 Make sure your Oracle user account has the necessary database access and privileges to access Oracle objects (such as tables and procedures).

Without the appropriate access and privileges, you will be unable to create Oracle stored procedures.

- 2 Assume the following table named `tt` exists in your Oracle database:

a	b	c
1	Newman	sysdate
2	Everett	sysdate

- 3 Create an Oracle package that holds the result set type and stored procedure. The result type must match your table definition.

For example, the following statement creates an Oracle package named `spm` that holds a result set type named `rc1` and a stored procedure named `proc1`. The `tt%ROWTYPE` attribute defines `rc1` to contain all of the columns in table `tt`. The procedure `proc1` takes one parameter, a cursor variable named `rc1` that is an IN OUT parameter of type `rc1`.

```
CREATE OR REPLACE PACKAGE spm
IS TYPE rc1 IS REF CURSOR
RETURN tt%ROWTYPE;
PROCEDURE proc1(rc1 IN OUT rc1);END;`
```

- 4 Create the Oracle stored procedure separately from the package you defined.

The following examples show how to create two stored procedures: *spm\_proc 1* (returns a single result set) and *spm\_proc2* (returns multiple result sets).

The IN OUT specification means that PowerBuilder passes the cursor variable (*rc1* or *rc2*) by reference to the Oracle procedure and expects the procedure to open the cursor. After the procedure call, PowerBuilder fetches the result set from the cursor and then closes the cursor.

**spm\_proc1 example for DataWindow objects** The following statements create *spm\_proc1* that returns one result set. You can use this procedure as the data source for a DataWindow object in PowerBuilder.

```
CREATE OR REPLACE PROCEDURE spm_proc1(rc1 IN OUT
spm.rctl)
AS
BEGIN
    OPEN rc1 FOR SELECT * FROM tt;END;`
```

**spm\_proc2 example for embedded SQL** The following statements create *spm\_proc2* which returns two result sets. You can use this procedure only in embedded SQL.

```
CREATE OR REPLACE PROCEDURE spm_proc2 (rc1 IN OUT
spm.rctl, rc2 IN OUT spm.rctl)
AS
BEGIN
    OPEN rc1 FOR SELECT * FROM tt ORDER BY 1;
    OPEN rc2 FOR SELECT * FROM tt ORDER BY 2;END;`
```

---

### Error checking

If necessary, check the Oracle system table *public.user\_errors* for a list of errors.

---

#### Creating the DataWindow object

After you create the stored procedure, you can define the DataWindow object that uses the stored procedure as a data source.

You can use Oracle stored procedures that return a single result set in a DataWindow object. If your stored procedure returns multiple result sets, you must use embedded SQL commands to access it.

The following procedure assumes that your Oracle stored procedure returns only a single result set.

❖ **To create a DataWindow object using an Oracle stored procedure with a result set:**

1 Select a presentation style on the DataWindow page of the New dialog box and click OK.

2 Select the Stored Procedure icon and click OK.

The Select Stored Procedure wizard page displays, listing the stored procedures available in your database.

3 Select the stored procedure you want to use as a data source, and click Next.

4 Complete the wizard to define the DataWindow object.

When you preview the DataWindow object or call Retrieve, PowerBuilder fetches the result set from the cursor in order to populate the DataWindow object. If you selected Retrieve on Preview on the Choose Data Source page in the wizard, the result set displays in the Preview view when the DataWindow opens.

For more instructions on defining DataWindow objects, see the *User's Guide*.

## Using a large-object output parameter

You can define a large object (LOB) as an output parameter for an Oracle stored procedure or function to retrieve large-object data. There is no limit on the number of LOB output arguments that may be defined for each stored procedure or function.

## Using Oracle user-defined types

What PowerBuilder supports

When you use the O84 and O90 database interfaces, PowerBuilder supports SQL CREATE TYPE and CREATE TABLE statements for Oracle user-defined types (objects) in the ISQL view of the Database painter. It correctly handles SQL SELECT, INSERT, UPDATE, and DELETE statements for user-defined types in the Database and DataWindow painters.

What you can do

This means that using these database interfaces in PowerBuilder, you can:

Do this	In
Use Oracle syntax to create user-defined types	Database painter
Use Oracle syntax to create tables with columns that reference user-defined types	Database painter
View columns in Oracle tables that reference user-defined types	Database painter
Manipulate data in Oracle tables that have user-defined types	Database painter DataWindow painter DataWindow objects
Export Oracle table syntax containing use-defined types to a log file	Database painter
Invoke methods of objects columns	DataWindow painter (Compute tab in SQL Toolbox)

Example

Here is a simple example that shows how you might create and use Oracle 8 user-defined types in PowerBuilder.

For more information about Oracle user-defined types, see your Oracle 8 documentation.

❖ **To create and use Oracle 8 and later user-defined types:**

- 1 In the ISQL view of the Database painter, create two Oracle user-defined types: `ball_stats_type` and `player_type`.

Here is the Oracle syntax to create `ball_stats_type`. Notice that the `ball_stats` object of type `ball_stats_type` has a method associated with it called `get_avg`.

```
CREATE OR REPLACE TYPE ball_stats_type AS OBJECT
(bat_avg NUMBER(4,3), rbi NUMBER(3), MEMBER FUNCTION
get_avg RETURN NUMBER, PRAGMA RESTRICT_REFERENCES
(get_avg, WNDS, RNPS, WNPS));
CREATE OR REPLACE TYPE BODY ball_stats_type AS MEMBER
FUNCTION get_avg RETURN NUMBER IS BEGIN RETURN
SELF.bat_avg;
END;
END;
```

Here is the Oracle SQL syntax to create `player_type`. `Player_type` references the user-defined type `ball_stats_type`. PowerBuilder supports such nesting graphically in the Database, DataWindow, and Table painters (see step 3).

```
CREATE TYPE player_type AS OBJECT (player_no
NUMBER(2),player_name VARCHAR2(30),ball_stats
ball_stats_type);
```

- 2 In the Database painter, create an Oracle 8 table named `lineup` that references these user-defined types.

Here is the Oracle SQL syntax to create the `lineup` table and insert a row. `Lineup` references the `player_type` user-defined type.

```
CREATE TABLE lineup (position NUMBER(2) NOT NULL,
player player_type);
INSERT INTO lineup VALUES (1,player_type (5, 'Nomar
Garciparra',ball_stats_type (0.310, 120)));
```

- 3 Display the `lineup` table in the Database or DataWindow painter.

PowerBuilder uses the following structure->member notation to display the table:

```
lineup
=====
position
player->player_no
player->player_name
player->ball_stats->bat_avg
player->ball_stats->rbi
```

- 4 To access the `get_avg` method of the object `ball_stats` contained in the object column `player`, use the following structure->member notation when defining a computed column for the DataWindow object. For example, when working in the DataWindow painter, you could use this notation on the Compute tab in the SQL Toolbox:

```
player->ball_stats->get_avg()
```

## What to do next

For instructions on connecting to the database, see “Connecting to a database” on page 121.

## Adaptive Server Enterprise

This section describes how to use the Adaptive Server Enterprise database interface in PowerBuilder.

---

### Client Library API

The Adaptive Server database interface uses the Open Client™ CT-Library (CT-Lib) application programming interface (API) to access the database.

When you connect to an Adaptive Server database, PowerBuilder makes the required calls to the API. Therefore, you do not need to know anything about CT-Lib to use the database interface.

---

## Supported versions for Adaptive Server

You can access Adaptive Server version 11.x and 12.x using the Adaptive Server database interface. Use of this interface to access other Open Server™ programs is not supported.

The Adaptive Server database interface uses a DLL named *PBSYC90.DLL* to access the database through the Open Client CT-Lib API.

---

### When deploying a PowerBuilder custom class user object in EAServer

EAServer uses a slightly different version of the CT-Lib software. Therefore, at runtime you need to use the SYJ database interface rather than SYC to connect to an Adaptive Server database. The SYJ Database Profile Setup dialog box provides a convenient way to set the appropriate connection parameters and then copy the syntax from the Preview tab into the script for your Transaction object. The SYJ database interface uses a DLL named *PBSYJ90.DLL*.

You cannot use the SYJ interface, however, to connect to the database in the PowerBuilder development environment. Therefore, during the development phase (before the component has been deployed to EAServer), you must use SYC to connect to the database.

---

## Supported Adaptive Server datatypes

The Adaptive Server interface supports the Sybase datatypes listed in Table 5-5 in DataWindow objects and embedded SQL.

**Table 5-5: Supported datatypes for Adaptive Server Enterprise**

Binary	Numeric
Bit	NVarChar (ASE 12.5 only)
Char (see “Column-length limits” on page 86)	Real
DateTime	SmallDateTime
Decimal	SmallInt
Double precision	SmallMoney
Float	Text
Identity	Timestamp
Image	TinyInt
Int	VarBinary
Money	VarChar
NChar (ASE 12.5 only)	

Unicode data access      PowerBuilder can access Unicode data in an ASE 12.5 Unicode database or in Unicode columns in ASE 12.5.

*Unicode database* — PowerBuilder converts between double-byte character set (DBCS) and Unicode automatically, provided that the *lang* variable of Open Client has been set to a DBCS language, such as chs or cht.

*Unicode columns* — PowerBuilder supports the access of Unicode data stored in columns of type NChar and NVarChar. PowerBuilder converts data automatically.

Column-length limits      Adaptive Server 12.0 and earlier have a column-length limit of 255 bytes. Adaptive Server 12.5 supports wider columns for Char, VarChar, Binary, and VarBinary datatypes, depending on the logical page size and the locking scheme used by the server.

In PowerBuilder, you can use these wider columns for Char and VarChar datatypes with Adaptive Server 12.5 when the following conditions apply:

- The Release DBParm is set to 12.5.
- You are accessing the database using Open Client 12.5.



The database must be configured to use a larger page size to take full advantage of the widest limits. For detailed information about wide columns and configuration issues, see the Adaptive Server 12.5 documentation on the Sybase Product Manuals Web site at <http://manuals.sybase.com:80/onlinebooks/group-as/asg1250e/>.

---

**Datatype conversion**

When you retrieve or update columns, PowerBuilder converts data appropriately between the Adaptive Server datatype and the PowerScript datatype. Keep in mind, however, that similarly or identically named Adaptive Server and PowerScript datatypes do *not* necessarily have the same definitions.

For information about the definitions of PowerScript datatypes, see the *PowerScript Reference*.

---

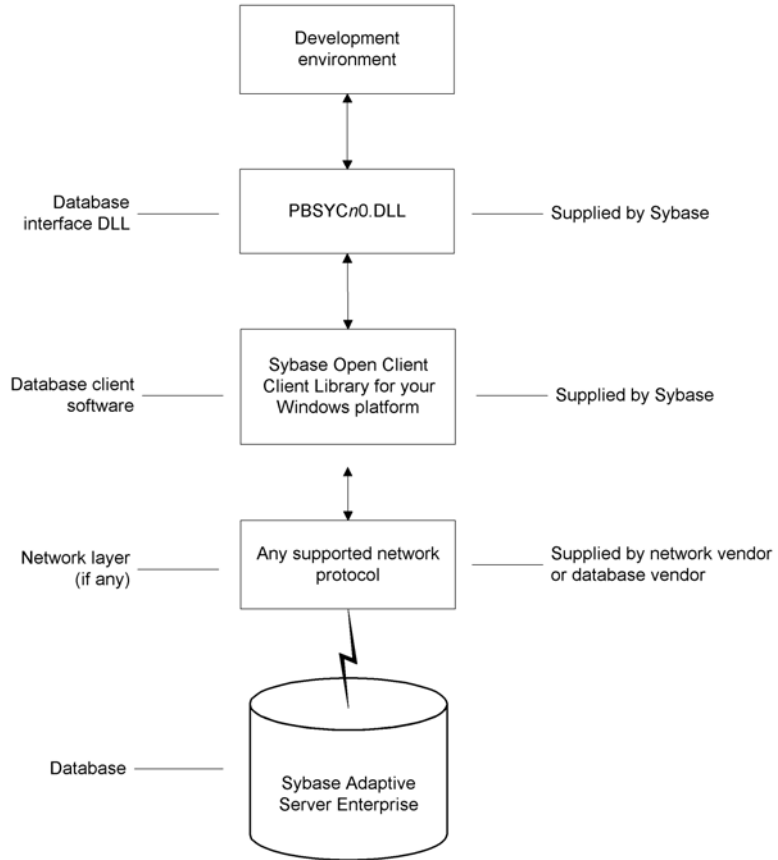
**Conversion in  
PowerBuilder scripts**

A double that has no fractional component is converted to a string with one decimal place if the converted string would cause Adaptive Server to have an overflow error when parsing the string. For example: the double value 12345678901234 would cause an overflow error, so PowerBuilder converts the double to the string value 12345678901234.0.

## Basic software components for Adaptive Server

You must install the software components in Figure 5-5 to access an Adaptive Server database in PowerBuilder.

**Figure 5-5: Components of an Adaptive Server Enterprise connection**



## Preparing to use the Adaptive Server database

Before you define the interface and connect to an Adaptive Server database in PowerBuilder, follow these steps to prepare the database for use:

- 1 Install and configure the required database server, network, and client software.

- 2 Install the Adaptive Server database interface.
- 3 Verify that you can connect to Adaptive Server outside PowerBuilder.
- 4 Install the required PowerBuilder stored procedures in the sybsystemprocs database.

Preparing an Adaptive Server database for use with PowerBuilder involves these four basic tasks.

Step 1: Install and  
configure the  
database server

You must install and configure the database server, network, and client software for Adaptive Server.

❖ **To install and configure the database server, network, and client software:**

- 1 Make sure the Adaptive Server database software is installed on the server specified in your database profile.

You must obtain the database server software from Sybase.

For installation instructions, see your Adaptive Server documentation.

- 2 Make sure the supported network software (for example, TCP/IP) is installed and running on your computer and is properly configured so that you can connect to the database server at your site.

You must install the network communication driver that supports the network protocol and operating system platform you are using. The driver is installed as part of the Net-Library client software.

For installation and configuration instructions, see your network or database administrator.

- 3 Install the required Open Client CT-Library (CT-Lib) software on each client computer on which PowerBuilder is installed.

You must obtain the Open Client software from Sybase. Make sure the version of Open Client you install supports *all* of the following:

- The operating system running on the client computer
- The version of Adaptive Server that you want to access
- The version of PowerBuilder that you are running

---

**Required client software versions**

To use the SYC Adaptive Server interface, you must install Open Client version 10.0.4 or later.

---

- 4 Make sure the Open Client software is properly configured so that you can connect to the database at your site.

Installing the Open Client software places the *SQL.INI* configuration file in the Adaptive Server directory on your computer.

*SQL.INI* provides information that Adaptive Server needs to find and connect to the database server at your site. You can enter and modify information in *SQL.INI* by using the configuration utility that comes with the Open Client software.

For information about setting up the *SQL.INI* or other required configuration file, see your Adaptive Server documentation.

- 5 If required by your operating system, make sure the directory containing the Open Client software is in your system path.
- 6 Make sure only one copy of each of the following files is installed on your client computer:
  - Adaptive Server interface DLL
  - Network communication DLL (for example, *NLWNSCK.DLL* for Windows Sockets-compliant TCP/IP)
  - Database vendor DLL (for example, *LIBCT.DLL*)

Step 2: Install the database interface

In the PowerBuilder Setup program, select the Typical install, or select the Custom install and select the Adaptive Server Enterprise (SYC) database interface.

If you work with PowerBuilder and EAServer, you should also select the Adaptive Server interface for EAServer (SYJ).

Step 3: Verify the connection

Make sure you can connect to the Adaptive Server database server and log in to the database you want to access from outside PowerBuilder.

Some possible ways to verify the connection are by running the following tools:

- **Accessing the database server** Tools such as the Open Client/Open Server Configuration utility (or any Ping utility) check whether you can reach the database server from your computer.
- **Accessing the database** Tools such as ISQL (interactive SQL utility) check whether you can log in to the database and perform database operations. It is a good idea to specify the same connection parameters you plan to use in your PowerBuilder database profile to access the database.

**Step 4: Install the PowerBuilder stored procedures**

PowerBuilder requires you to install certain stored procedures in the sybssystemprocs database *before* you connect to an Adaptive Server database for the first time. PowerBuilder uses these stored procedures to get information about tables and columns from the DBMS system catalog.

Run the SQL script or scripts required to install the PowerBuilder stored procedures in the sybssystemprocs database.

For instructions, see “Installing PowerBuilder stored procedures in Adaptive Server databases” on page 99.

**What to do next**

For instructions on defining the Adaptive Server database interface in PowerBuilder, see “Defining the Adaptive Server database interface” on page 91.

**Defining the Adaptive Server database interface**

To define a connection through the Adaptive Server interface, you must create a database profile by supplying values for at least the basic connection parameters in the Database Profile Setup - Adaptive Server Enterprise dialog box. You can then select this profile anytime to connect to your database in the development environment.

For information on how to define a database profile, see “Using database profiles” on page 8.

---

**Defining a connection for a PowerBuilder custom class user object deployed in EAServer**

You cannot use the SYJ interface to connect to the database in the PowerBuilder development environment. However, the SYJ Database Profile Setup dialog box provides a convenient way to set the appropriate connection parameters and then copy the syntax from the Preview tab into the script for your Transaction object.

---

## Using Open Client security services

The Adaptive Server interfaces provide several DBParm parameters that support Open Client 11.1.x or later network-based security services in your application. If you are using the required database, security, and PowerBuilder software, you can build applications that take advantage of Open Client security services.

### What are Open Client security services?

Open Client 11.1.x or later **security services** allow you to use a supported third-party security mechanism (such as CyberSafe Kerberos) to provide login authentication and per-packet security for your application. Login authentication establishes a secure connection, and per-packet security protects the data you transmit across the network.

---

#### Using third-party security mechanisms

For information about the third-party security mechanisms and operating system platforms that Sybase has tested with Open Client security services, see the Open Client documentation.

---

### Requirements for using Open Client security services

To use Open Client security services in your application, *all of the following must be true*:

- Accessing Adaptive Server through Open Client 11.1.1x or later  
You are accessing an Adaptive Server database server using Open Client Client-Library (CT-Lib) 11.x or later software
- Have network security mechanism and driver  
You have the required Sybase-supported network security mechanism and Sybase-supplied security driver properly installed and configured for your environment. Depending on your operating system platform, examples of supported security mechanisms include: Distributed Computing Environment (DCE) security servers and clients, CyberSafe Kerberos, and Windows NT LAN Manager Security Services Provider Interface (SSPI).  
For information about the third-party security mechanisms and operating system platforms that Sybase has tested with Open Client security services, see the Open Client documentation.

- Can access the secure server outside PowerBuilder

You must be able to access a secure Adaptive Server server using Open Client 11.1.x or later software from outside PowerBuilder.

To verify the connection, use a tool such as ISQL or SQL Advantage to make sure you can connect to the server and log in to the database with the same connection parameters and security options you plan to use in your PowerBuilder application.

- Using PowerBuilder database interface

You are using the SYC Adaptive Server interface to access the database.

- Release DBParm parameter set to the appropriate value for your database

You have set the Release DBParm parameter to 11, 11.5, 12, or 12.5 to specify that your application should use the appropriate version of the Open Client CT-Lib software.

For instructions, see Release (Adaptive Server Enterprise) in the online Help.

- Security mechanism and driver support requested service

The security mechanism and driver you are using must support the service requested by the DBParm parameter.

## Security services DBParm parameters

If you have met the requirements described in “Requirements for using Open Client security services” on page 92, you can set the security services DBParm parameters in the Database Profile Setup dialog box for your connection or in a PowerBuilder application script.

There are two types of DBParm parameters that you can set to support Open Client security services: login authentication and per-packet security.

### Login authentication DBParms

The following login authentication DBParm parameters correspond to Open Client 11.1.x or later connection properties that allow an application to establish a secure connection.

- Sec\_Channel\_Bind
- Sec\_Cred\_Timeout
- Sec\_Delegation
- Sec\_Keytab\_File
- Sec\_Mechanism
- Sec\_Mutual\_Auth

Sec\_Network\_Auth  
Sec\_Server\_Principal  
Sec\_Sess\_Timeout

For instructions on setting these DBParm parameters, see their descriptions in online Help.

#### Per-packet security DBParms

The following per-packet security DBParm parameters correspond to Open Client 11.1.x or later connection properties that protect each packet of data transmitted across a network. Using per-packet security services may create extra overhead for communications between the client and server.

Sec\_Confidential  
Sec\_Data\_Integrity  
Sec\_Data\_Origin  
Sec\_Replay\_Detection  
Sec\_Seq\_Detection

For instructions on setting these DBParm parameters, see their descriptions in online Help.

## Using Open Client directory services

The Adaptive Server interfaces provide several DBParm parameters that support Open Client 11.1.x or later network-based directory services in your application. If you are using the required database, directory services, and PowerBuilder software, you can build applications that take advantage of Open Client directory services.

### What are Open Client directory services?

Open Client 11.1.x or later **directory services** allow you to use a supported third-party directory services product (such as the Windows NT Registry) as your directory service provider. Directory services provide centralized control and administration of the network entities (such as users, servers, and printers) in your environment.

---

#### Using third-party directory service providers

For information about the third-party directory service providers and operating system platforms that Sybase has tested with Open Client directory services, see the Open Client documentation.

---



## Requirements for using Open Client directory services

To use Open Client directory services in your application, *all of the following must be true*:

- You are accessing an Adaptive Server database server using Open Client Client-Library (CT-Lib) 11.x or later software
- You have the required Sybase-supported directory service provider software and Sybase-supplied directory driver properly installed and configured for your environment. Depending on your operating system platform, examples of supported security mechanisms include: the Windows NT Registry, Distributed Computing Environment Cell Directory Services (DCE/CDS), Banyan StreetTalk Directory Assistance (STDA), and Novell NetWare Directory Services (NDS).

For information about the directory service providers and operating system platforms that Sybase has tested with Open Client directory services, see the documentation.

- You must be able to access a secure Adaptive Server server using Open Client 11.1.x or later software from outside PowerBuilder.

To verify the connection, use a tool such as ISQL or SQL Advantage to make sure you can connect to the server and log in to the database with the same connection parameters and directory service options you plan to use in your PowerBuilder application.

- You are using the SYC Adaptive Server interface to access the database.
- You must use the correct syntax as required by your directory service provider when specifying the server name in a database profile or PowerBuilder application script. Different providers require different syntax based on their format for specifying directory entry names.

For information and examples for different directory service providers, see "Specifying the server name with Open Client directory services" next.

- You have set the Release DBParm parameter to 11, 11.5, 12, or 12.5 to specify that your application should use the behavior of the appropriate version of the Open Client CT-Lib software.

For instructions, see Release (Adaptive Server Enterprise) in the online Help.

- The directory service provider and driver you are using must support the service requested by the DBParm parameter.

## Specifying the server name with Open Client directory services

When you are using Open Client directory services in a PowerBuilder application, you must use the syntax required by your directory service provider when specifying the server name in a database profile or PowerBuilder application script to access the database.

Different directory service providers require different syntax based on the format they use for specifying directory entry names. Directory entry names can be fully qualified or relative to the default (active) Directory Information Tree base (DIT base) specified in the Open Client/Server™ configuration utility.

The **DIT base** is the starting node for directory searches. Specifying a DIT base is analogous to setting a current working directory for UNIX or MS-DOS file systems. (You can specify a nondefault DIT base with the DS\_DitBase DBParm parameter. For information, see DS\_DitBase in the online Help.)

Windows NT Registry  
server name example

This example shows typical server name syntax if your directory service provider is the Windows NT Registry.

```
Node name: SALES:software\sybase\server\SYS12NT
DIT base: SALES:software\sybase\server
Server name: SYS12NT
```

### ❖ To specify the server name in a database profile:

Type the following in the Server box on the Connection tab in the Database Profile Setup dialog box. Do *not* start the server name with a backslash (\).

```
SYS12NT
```

### ❖ To specify the server name in a PowerBuilder application script:

- Type the following. Do *not* start the server name with a backslash (\).

```
SQLCA.ServerName = "SYS12NT"
```

If you specify a value in the Server box in your database profile, this syntax displays on the Preview tab in the Database Profile Setup dialog box. You can copy the syntax from the Preview tab into your script.

DCE/DCS server  
name example

This example shows typical server name syntax if your directory service provider is Distributed Computing Environment Cell Directory Services (DCE/CDS).

```
Node name: ../../boston.sales/dataservers/sybase/SYS12
DIT base: ../../boston.sales/dataservers
Server name: sybase/SYS12
```

❖ **To specify the server name in a database profile:**

Type the following in the Server box on the Connection tab in the Database Profile Setup dialog box. Do *not* start the server name with a slash (/).

```
sybase/SYS12
```

❖ **To specify the server name in a PowerBuilder application script:**

- Type the following. Do *not* start the server name with a slash (/).

```
SQLCA.ServerName = "sybase/SYS12"
```

If you specify a value in the Server box in your database profile, this syntax displays on the Preview tab in the Database Profile Setup dialog box. You can copy the syntax from the Preview tab into your script.

Banyan STDA server  
name example

This example shows typical server name syntax if your directory service provider is Banyan StreetTalk Directory Assistance (STDA).

```
Node name: SYS12@sales@chicago  
DIT base: chicago  
Server name: SYS12@sales
```

❖ **To specify the server name in a database profile:**

Type the following in the Server box on the Connection tab in the Database Profile Setup dialog box. Do *not* end the server name with @.

```
SYS12@sales
```

❖ **To specify the server name in a PowerBuilder application script:**

- Type the following. Do *not* end the server name with @.

```
SQLCA.ServerName = "SYS12@sales"
```

If you specify a value in the Server box in your database profile, this syntax displays on the Preview tab in the Database Profile Setup dialog box. You can copy the syntax from the Preview tab into your script.

Novell NDS server  
name example

This example shows typical server name syntax if your directory service provider is Novell NetWare Directory Services (NDS).

```
Node name: CN=SYS12.OU=miami.OU=sales.O=sybase  
DIT base: OU=miami.OU=sales.O=sybase  
Server name: SYS12
```

❖ **To specify the server name in a database profile:**

- Type the following in the Server box on the Connection tab in the Database Profile Setup dialog box. Do *not* start the server name with CN=.

SYS12

❖ **To specify the server name in a PowerBuilder application script:**

- Type the following. Do *not* start the server name with CN=.

```
SQLCA.ServerName = "SYS12"
```

If you specify a value in the Server box in your database profile, this syntax displays on the Preview tab in the Database Profile Setup dialog box. You can copy the syntax from the Preview tab into your script.

## Directory services DBParm parameters

If you have met the requirements described in “Requirements for using Open Client directory services” on page 95, you can set the directory services DBParm parameters in a database profile for your connection or in a PowerBuilder application script.

The following DBParm parameters correspond to Open Client 11.1.x or later directory services connection parameters:

DS\_Alias  
DS\_Copy  
DS\_DitBase  
DS\_Failover  
DS\_Password (Open Client 12.5 or later)  
DS\_Principal  
DS\_Provider  
DS\_TimeLimit

For instructions on setting these DBParm parameters, see their descriptions in the online Help.

## Using PRINT statements in Adaptive Server stored procedures

The SYC Adaptive Server database interface allows you to use PRINT statements in your stored procedures for debugging purposes.

This means, for example, that if you turn on Database Trace when accessing the database through the SYC interface, PRINT messages appear in the trace log but they do not return errors or cancel the rest of the stored procedure.

## Creating a DataWindow based on a heterogeneous cross-database join

This functionality is available through the use of Adaptive Server's Component Integration Services. Component Integration Services allows you to connect to multiple remote heterogeneous database servers and define multiple proxy tables that reference the tables residing on those servers.

For information on how to create proxy tables, see the Adaptive Server documentation.

## What to do next

For instructions on connecting to the database, see “Connecting to a database” on page 121.

## Installing PowerBuilder stored procedures in Adaptive Server databases

This section describes how to install PowerBuilder stored procedures in an Adaptive Server database by running SQL scripts provided for this purpose.

Sybase recommends that you run these scripts outside PowerBuilder *before* connecting to an Adaptive Server database for the first time through the Adaptive Server (SYC DBMS identifier) native database interface. Although the PBSYC development environment will run without the PowerBuilder stored procedures created by these scripts, the stored procedures are required for full functionality.

## What are the PowerBuilder stored procedure scripts?

What you do	<p>In order to work with an Adaptive Server database in PowerBuilder, you or your system administrator should install certain stored procedures in the database <i>before</i> you connect to Adaptive Server from PowerBuilder <i>for the first time</i>.</p> <p>You must run the PowerBuilder stored procedure scripts only once per database server, and not before each PowerBuilder session. If you have already installed the PowerBuilder stored procedures in your Adaptive Server database before connecting in PowerBuilder on any supported platform, you need <i>not</i> install the stored procedures again before connecting in PowerBuilder on a different platform.</p>						
PowerBuilder stored procedures	<p>A <b>stored procedure</b> is a group of precompiled and preoptimized SQL statements that performs some database operation. Stored procedures reside on the database server where they can be accessed as needed.</p> <p>PowerBuilder uses these stored procedures to get information about tables and columns from the Adaptive Server system catalog. (The PowerBuilder stored procedures are different from the stored procedures you may create in your database.)</p>						
SQL scripts	<p>PowerBuilder provides SQL script files for installing the required stored procedures in sybsystemprocs database:</p> <table><tr><th>Script</th><th>Use for</th></tr><tr><td><i>PBSYC.SQL</i></td><td>Adaptive Server databases</td></tr><tr><td><i>PBSYC2.SQL</i></td><td>Adaptive Server databases to restrict the Select Tables list</td></tr></table>	Script	Use for	<i>PBSYC.SQL</i>	Adaptive Server databases	<i>PBSYC2.SQL</i>	Adaptive Server databases to restrict the Select Tables list
Script	Use for						
<i>PBSYC.SQL</i>	Adaptive Server databases						
<i>PBSYC2.SQL</i>	Adaptive Server databases to restrict the Select Tables list						
Where to find the scripts	<p>The stored procedure scripts are located in the <i>Server</i> directory on the PowerBuilder CD-ROM. The <i>Server</i> directory contains server-side installation components that are <i>not</i> installed with PowerBuilder on your computer.</p>						

## PBSYC.SQL script

What it does	<p>The <i>PBSYC.SQL</i> script contains SQL code that overwrites stored procedures that correspond to the same version of PowerBuilder in the Adaptive Server sybsystemprocs database and then re-creates them.</p> <p>The <i>PBSYC.SQL</i> script uses the sybsystemprocs database to hold the PowerBuilder stored procedures. This database is created when you install Adaptive Server.</p>
--------------	--

**When to run it**

Before you connect to an Adaptive Server database in PowerBuilder *for the first time* using the SYC DBMS identifier, you or your database administrator *must run the PBSYC.SQL script once per database server into the sybsystemprocs database.*

Run *PBSYC.SQL* if the server at your site will be *accessed by anyone using the PowerBuilder development environment or by deployment machines.*

If you or your database administrator have already run the current version of *PBSYC.SQL* to install PowerBuilder stored procedures in the sybsystemprocs database on your server, you need not rerun the script to install the stored procedures again.

For instructions on running *PBSYC.SQL*, see “How to run the scripts” on page 103.

**Stored procedures it creates**

The *PBSYC.SQL* script creates the following PowerBuilder stored procedures in the Adaptive Server sybsystemprocs database. The procedures are listed in the order in which the script creates them.

<b>PBSYC.SQL stored procedure</b>	<b>What it does</b>
sp_pb90column	Lists the columns in a table.
sp_pb90primarykey	Lists the columns in a table's primary key.
sp_pb90pkcheck	Determines whether a table has a primary key.
sp_pb90fktable	Lists the tables that reference the current table.
sp_pb90foreignkey	Lists all foreign keys associated with a specified table.
sp_pb90extcat	Checks the status of the extended attributes system tables.
sp_pb90procdesc	Retrieves a description of the argument list for a specified stored procedure.
sp_pb90proclist	Lists available stored procedures and extended stored procedures.  If the SystemProcs DBParm parameter is set to 1 or Yes (the default), sp_pb90proclist displays both system stored procedures and user-defined stored procedures. If SystemProcs is set to 0 or No, sp_pb90proclist displays only user-defined stored procedures.
sp_pb90text	Retrieves the text of a stored procedure from the SYSCOMMENTS table.
sp_pb90db	Retrieves the names of all databases available for this server.

PBSYC.SQL stored procedure	What it does
sp_pb90table	Retrieves information about <i>all</i> tables in a database, including those for which the current user has no permissions.  PBSYC.SQL contains the default version of sp_pb90table. If you want to replace the default version of sp_pb90table with a version that restricts the table list to those tables for which the user has SELECT permission, you can run the <i>PBSYC2.SQL</i> script, described in "PBSYC2.SQL script" next.
sp_pb90index	Retrieves information about all indexes for a specified table.

## **PBSYC2.SQL script**

### What it does

The *PBSYC2.SQL* script contains SQL code that drops and re-creates one PowerBuilder stored procedure in the Adaptive Server sybsystemprocs database: a replacement version of sp\_pb90table.

The default version of sp\_pb90table is installed by the *PBSYC.SQL* script. PowerBuilder uses the sp\_pb90table procedure to build a list of *all* tables in the database, including those for which the current user has no permissions. This list displays in the Select Tables dialog box in PowerBuilder.

For security reasons, you or your database administrator may want to restrict the table list to display only those tables for which a user has permissions. To do this, you can run the *PBSYC2.SQL* script *after you run PBSYC.SQL*. *PBSYC2.SQL* replaces the default version of sp\_pb90table with a new version that displays a restricted table list including only tables and views:

- Owned by the current user
- For which the current user has SELECT authority
- For which the current user's group has SELECT authority
- For which SELECT authority was granted to PUBLIC

### When to run it

If you are accessing an Adaptive Server database using the SYC DBMS identifier in PowerBuilder, *you must first run PBSYC.SQL* once per database server to install the required PowerBuilder stored procedures in the sybsystemprocs database.



After you run *PBSYC.SQL*, you can optionally run *PBSYC2.SQL* if you want to replace `sp_pb90table` with a version that restricts the table list to those tables for which the user has `SELECT` permission.

If you do not want to restrict the table list, there is no need to run *PBSYC2.SQL*.

For instructions on running *PBSYC2.SQL*, see “How to run the scripts” on page 103.

Stored procedure it creates

The *PBSYC2.SQL* script creates the following PowerBuilder stored procedure in the Adaptive Server sybsystemprocs database:

<b>PBSYC2.SQL stored procedure</b>	<b>What it does</b>
<code>sp_pb90table</code>	Retrieves information about those tables in the database for which the current user has <code>SELECT</code> permission.  This version of <code>sp_pb90table</code> replaces the default version of <code>sp_pb90table</code> installed by the <i>PBSYC.SQL</i> script.

## How to run the scripts

You can use the ISQL or SQL Advantage tools to run the stored procedure scripts outside PowerBuilder.

### Using ISQL to run the stored procedure scripts

ISQL is an interactive SQL utility that comes with the Open Client software on the Windows platforms. If you have ISQL installed, use the following procedure to run the PowerBuilder stored procedure scripts.

For complete instructions on using ISQL, see your Open Client documentation.

#### ❖ To use ISQL to run the PowerBuilder stored procedure scripts:

- 1 Connect to the sybsystemprocs Adaptive Server database as the system administrator.
- 2 Open one of the following files containing the PowerBuilder stored procedure script you want to run:

*PBSYC.SQL*  
*PBSYC2.SQL*

- 3 Issue the appropriate ISQL command to run the SQL script with the user ID, server name, and (optionally) password you specify. Make sure you specify uppercase and lowercase exactly as shown:

**isql /U sa /S SERVERNAME /i pathname /P { password }**

Parameter	Description
sa	The user ID for the system administrator. Do <i>not</i> change this user ID.
SERVERNAME	The name of the computer running the Adaptive Server database.
pathname	The drive and directory containing the SQL script you want to run.
password	(Optional) The password for the sa (system administrator) user ID. The default Adaptive Server installation creates the sa user ID without a password. If you changed the password for sa during the installation, replace <i>password</i> with your new password.

For example, if you are using PowerBuilder and are accessing the stored procedure scripts from the product CD-ROM, type either of the following (assuming D is your CD-ROM drive):

```
isql /U sa /S TESTDB /i d:\server\pbsyb.sql /P
isql /U sa /S SALES /i d:\server\pbsyc.sql /P
adminpwd
```

## Using SQL Advantage to run the stored procedure scripts

SQL Advantage is an interactive SQL utility that comes with the Open Client software on the Windows platform. If you have SQL Advantage installed, use the following procedure to run the PowerBuilder stored procedure scripts.

For complete instructions on using SQL Advantage, see your Open Client documentation.

### ❖ To use SQL Advantage to run the PowerBuilder stored procedure scripts:

- 1 Start the SQL Advantage utility.
- 2 Open a connection to the sybsystemprocs Adaptive Server database as the system administrator.

- 3 Open one of the following files containing the PowerBuilder stored procedure script you want to run:

*PBSYC.SQL*  
*PBSYC2.SQL*

- 4 Delete the use sybssystemprocs command and the go command at the beginning of each script.

SQL Advantage requires that you issue the use sybssystemprocs command by itself, with no other SQL commands following it. When you open a connection to the sybssystemprocs database in step 2, you are in effect issuing the use sybssystemprocs command. This command should not be issued again as part of the stored procedure script.

Therefore, to successfully install the stored procedures, you *must* delete the lines shown in the following table from the beginning of the PowerBuilder stored procedure script *before* executing the script.

Before executing this script	Delete these lines
<i>PBSYC.SQL</i>	use sybssystemprocs go
<i>PBSYC2.SQL</i>	use sybssystemprocs go

- 5 Execute all of the statements in the SQL script.
- 6 Exit the SQL Advantage session.

## DirectConnect

This section describes how to use the DirectConnect™ interface in PowerBuilder.

### Using the DirectConnect interface

The DirectConnect interface uses Sybase's Open Client CT-Library (CT-Lib) API to access a database through Sybase middleware data access products such as the DirectConnect for OS/390 component of MainFrame Connect and Open ServerConnect™.

## Connecting through the DirectConnect middleware product

Sybase DirectConnect is a data access server that provides a standardized middleware interface between your applications and your enterprise data sources. Data access services to a particular database are defined in a DirectConnect server. Since a DirectConnect server can support multiple access services, you can access multiple databases through a single server. When you use the DirectConnect interface to connect to a particular database, your connection is routed through the access service for that database. An access service consists of a named set of configuration properties and a specific access service library.

To access DB2 data on an IBM mainframe through a DirectConnect server, you can use the DirectConnect interface to connect through either a DirectConnect for MVS access service or a DirectConnect Transaction Router Service (TRS).

TRS provides fast access to a DB2/MVS database by using remote stored procedures. The DirectConnect interface supports both versions of the TRS library: TRSLU62 and TRSTCP.

The DirectConnect server operates in two modes: SQL transformation and passthrough. The DirectConnect interface for DB2/MVS uses passthrough mode, which allows your PowerBuilder application to have direct access to the capabilities of the DB2/MVS data source.

## Connecting through the Open ServerConnect middleware product

Sybase's Open ServerConnect supports mainframe applications that retrieve and update data stored on the mainframe that Sybase client applications can execute. Client applications can connect directly to a DB2/MVS database through an Open ServerConnect application residing on the mainframe, eliminating the need for an intermediate gateway like DirectConnect. (This type of connection is also known as a *gateway-less* connection.) In addition, an Open ServerConnect application presents mainframe Remote Procedure Calls (RPCs) as database stored procedures to the client application.

To access DB2 data on an IBM mainframe through Open ServerConnect, you can use the DirectConnect interface to connect through Open ServerConnect for IMS and MVS.

## Selecting the type of connection

To select how PowerBuilder accesses the database, use the Choose Gateway drop-down list on the Connection tab of the DirectConnect Database Profile Setup dialog box and select one of the following:

- Access Service
- Gatewayless
- TRS

All the DBParm parameters defined for the DirectConnect interface are applicable to all three connections except the following:

- HostReqOwner applies to Access Service and Gatewayless only
- Request, ShowWarnings, and SystemOwner apply to Access Service only
- UseProcSyntax applies to Gatewayless only

See the online help for the complete list of DBParm parameters applicable to the DirectConnect interface.

## Basic software components for the DirectConnect interface

Figure 5-6 shows the basic software components required to access a database using the DirectConnect interface and the DirectConnect middleware data access product.

**Figure 5-6: Components of a DirectConnect connection using DirectConnect middleware**

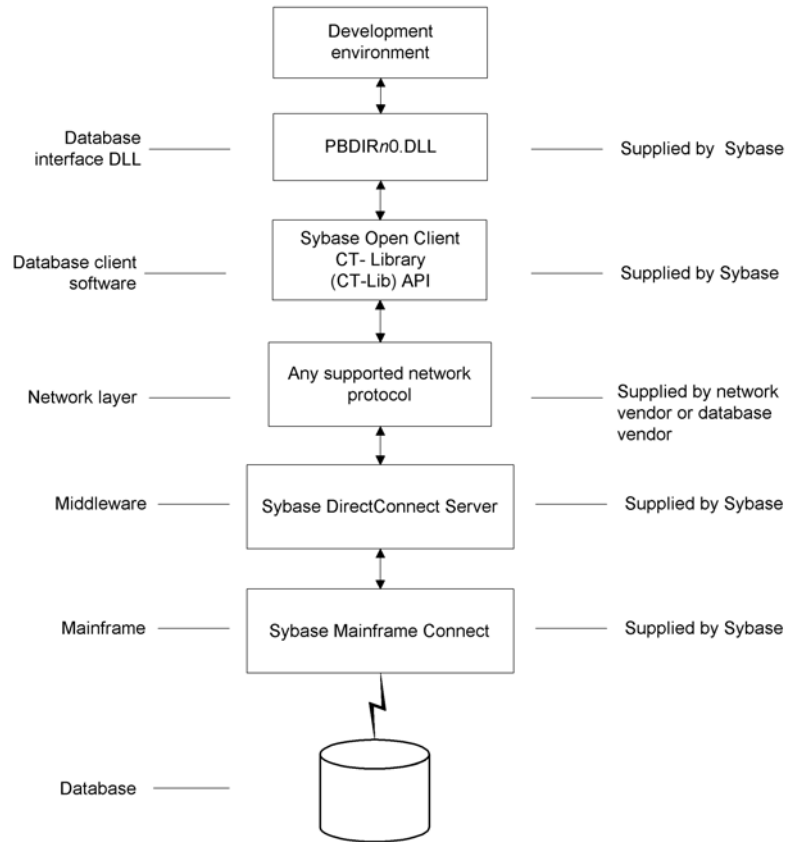
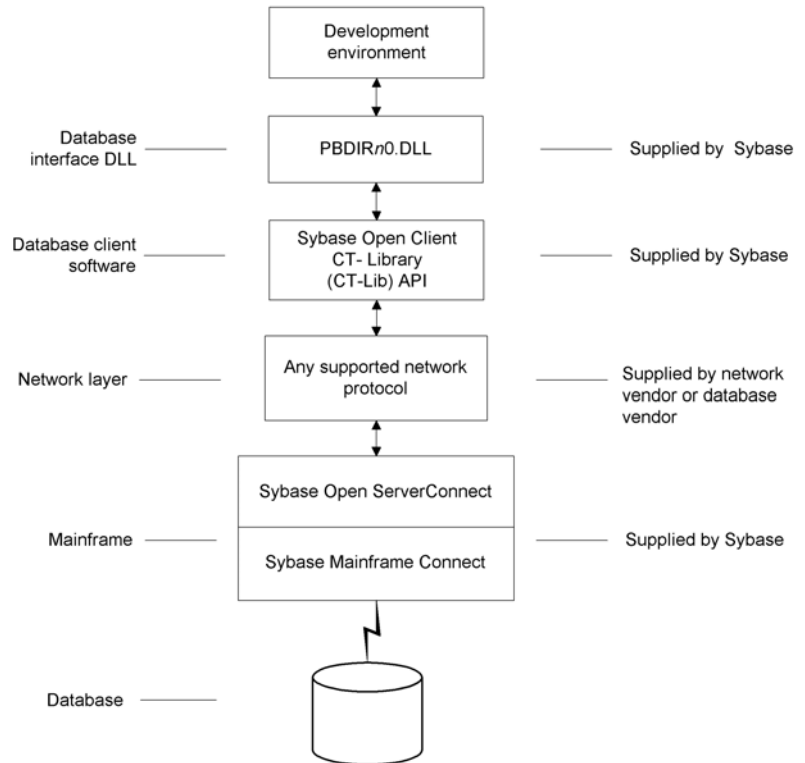


Figure 5-7 shows the basic software components required to access a database using the DirectConnect interface and the Open ServerConnect middleware data access product.

**Figure 5-7: Components of a DirectConnect connection using Open ServerConnect middleware**



## Supported versions for the DirectConnect interface

The DirectConnect interface uses a DLL named *PBDIR90.DLL* to access a database through either DirectConnect or Open ServerConnect.

---

### Required DirectConnect versions

To access a DB2/MVS database through the access service, it is strongly recommended that you use DirectConnect for MVS access service version 11.1.1p4 or later.

To access a DB2/MVS database through TRS, it is strongly recommended that you use DirectConnect TRS version 11.1.1p4 or later.

For information on DirectConnect for MVS and TRS, see your DirectConnect documentation.

---

---

### Required Open ServerConnect versions

To access a DB2/MVS database through Open ServerConnect, it is strongly recommended that you use Open ServerConnect IMS and MVS version 4.0 or later.

For information on Open ServerConnect for MVS, see your Open ServerConnect documentation.

---

## Supported DirectConnect interface datatypes

The DirectConnect interface supports the PowerBuilder datatypes listed in Table 5-6 in DataWindow objects. and embedded SQL.

**Table 5-6: Supported datatypes for DirectConnect**

Char (fewer than 255 characters)	Long VarChar
Char for Bit Data	Real
Date	SmallInt
Decimal	Time
Double Precision	Timestamp (DateTime)
Float	VarChar
Integer	VarChar for Bit Data



## Preparing to use the database with DirectConnect

Before you define the interface and connect to a database through the DirectConnect interface, follow these steps to prepare the database for use:

- 1 Install and configure the Sybase middleware data access products, network, and client software.
- 2 Install the DirectConnect interface.
- 3 Verify that you can connect to your middleware product and your database outside PowerBuilder.
- 4 Create the extended attribute system tables outside PowerBuilder.

Step 1: Install and configure the Sybase middleware product

You must install and configure the Sybase middleware data access product, network, and client software.

❖ **To install and configure the Sybase middleware data access product, network, and client software:**

- 1 Make sure the appropriate database software is installed and running on its server.

You must obtain the database server software from your database vendor.

For installation instructions, see your database vendor's documentation.

- 2 Make sure the appropriate DirectConnect access service software is installed and running on the DirectConnect server specified in your database profile.

*or*

Make sure the appropriate Open ServerConnect software is installed and running on the mainframe specified in your database profile.

- 3 Make sure the required network software (such as TCP/IP) is installed and running on your computer and is properly configured so you that can connect to the DirectConnect server or mainframe at your site.

You must install the network communication driver that supports the network protocol and operating system platform you are using.

For installation and configuration instructions, see your network or database administrator.

- 4 Install the required Open Client CT-Library (CT-Lib) software on each client computer on which PowerBuilder is installed.

You must obtain the Open Client software from Sybase. Make sure the version of Open Client you install supports *both* of the following:

- The operating system running on the client computer
- The version of PowerBuilder that you are running

---

#### **Required Open Client versions**

To use the DirectConnect interface, you must install Open Client.

---

For information about Open Client, see your Open Client documentation.

- 5 Make sure the Open Client software is properly configured so you can connect to the middleware data access product at your site.

Installing the Open Client software places the *SQL.INI* configuration file in the SQL Server directory on your computer. *SQL.INI* provides information that SQL Server uses to find and connect to the middleware product at your site. You can enter and modify information in *SQL.INI* with the configuration utility or editor that comes with the Open Client software.

For information about editing the *SQL.INI* file, see “Editing the SQL.INI file” on page 113. For more information about setting up *SQL.INI* or any other required configuration file, see your SQL Server documentation.

- 6 If required by your operating system, make sure the directory containing the Open Client software is in your system path.
- 7 Make sure only one copy of each of the following files is installed on your client computer:
  - DirectConnect interface DLL
  - Network communication DLL (such as *NLWNSCK.DLL* for Windows Sockets-compliant TCP/IP)
  - Open Client DLLs (such as *LIBCT.DLL* and *LIBCS.DLL*)

Step 2: Install the interface

In the PowerBuilder Setup program, select the Typical install, or select the Custom install and select the Direct Connect Interface (DIR).

**Step 3: Verify the connection**

Make sure you can connect to the your middleware product and your database and log in to the database you want to access from outside PowerBuilder.

Some possible ways to verify the connection are by running the following tools:

- **Accessing the database server** Tools such as the Open Client/Open Server Configuration utility (or any Ping utility) check whether you can reach the database server from your computer.
- **Accessing the database** Tools such as ISQL or SQL Advantage (interactive SQL utilities) check whether you can log in to the database and perform database operations. It is a good idea to specify the same connection parameters you plan to use in your PowerBuilder database profile to access the database.

**Step 4: Create the extended attribute system tables**

PowerBuilder uses a collection of five system tables to store extended attribute information. When using the DirectConnect interface, you *must* create the extended attribute system tables outside PowerBuilder to control the access rights and location of these tables.

Run the *DB2SYSPB.SQL* script outside PowerBuilder using the SQL tool of your choice.

For instructions, see “Creating the extended attribute system tables in DB2 databases” on page 114.

**Editing the SQL.INI file**

Make sure the *SQL.INI* file provides an entry about either the access service being used and the DirectConnect server on which it resides or the Open ServerConnect program being used and the mainframe on which it resides.

For the server object name, you need to provide the exact access service name as it is defined in the access service library configuration file on the DirectConnect server. You must also specify the network communication DLL being used, the TCP/IP address or alias used for the DirectConnect server on which the access service resides, and the port on which the DirectConnect server listens for requests:

```
[access_service_name]  
query=network_dll,server_alias,server_port_no
```

PowerBuilder users must also specify the access service name in the *SQLCA.ServerName* property of the Transaction object.

## Defining the DirectConnect interface

To define a connection through the DirectConnect interface, you must create a database profile by supplying values for at least the basic connection parameters in the Database Profile Setup - DirectConnect dialog box. You can then select this profile anytime to connect to your database in the development environment.

For information on how to define a database profile, see “Using database profiles” on page 8.

## Creating the extended attribute system tables in DB2 databases

This section describes how PowerBuilder creates the extended attribute system tables in your DB2 database to store extended attribute information. It then explains how to use the *DB2SYSPB.SQL* script to create the extended attribute system tables outside PowerBuilder.

You can use the *DB2SYSPB.SQL* script if you are connecting to the IBM DB2 family of databases through any of the following database interfaces:

- ODBC interface
- Sybase DirectConnect interface

## Creating the extended attribute system tables

When you create or modify a table in PowerBuilder, the information you provide is stored in five system tables in your database. These system tables contain extended attribute information such as the text to use for labels and column headings, validation rules, display formats, and edit styles. (These system tables are different from the system tables provided by your DB2 database.)

By default, the extended attribute system tables are created automatically the first time a user connects to the database using PowerBuilder.

---

**When you use the DirectConnect interface**

When you use the DirectConnect interface, the extended attribute system tables are *not* created automatically. You must run the *DB2SYSPB.SQL* script to create the system tables as described in “Using the DB2SYSPB.SQL script” on page 115.

---

- ❖ **To ensure that the extended attribute system tables are created with the proper access rights:**
  - Make sure the first person to connect to the database with PowerBuilder has sufficient authority to create tables and grant permissions to PUBLIC.  
  
This means that the first person to connect to the database should log in as the database owner, database administrator, system user, system administrator, or system owner, as specified by your DBMS.

## Using the *DB2SYSPB.SQL* script

Why do this

If you are a system administrator at a DB2 site, you may prefer to create the extended attribute system tables outside PowerBuilder for two reasons:

- The first user to connect to the DB2 database using PowerBuilder may not have the proper authority to create tables.
- When PowerBuilder creates the extended attribute system tables, it places them in the default tablespace. This may not be appropriate for your needs.

---

**When using the DirectConnect interface**

You *must* create the extended attribute system tables outside PowerBuilder if you are using the DirectConnect interface. You need to decide which database and tablespace should store the system tables. You may also want to grant update privileges only to specific developers or groups.

---

What you do

To create the extended attribute system tables, you run the *DB2SYSPB.SQL* script outside PowerBuilder. This script contains SQL commands that create and initialize the system tables with the table owner and tablespace you specify.

Where to find  
*DB2SYSPB.SQL*

The *DB2SYSPB.SQL* script is in the *Server* directory on the PowerBuilder CD-ROM. This directory contains server-side installation components and is *not installed* with PowerBuilder on your computer.

You can access the *DB2SYSPB.SQL* script directly from your computer's CD-ROM drive or you can copy it to your computer.

Use the following procedure *from the database server* to create the extended attribute system tables in a DB2 database outside PowerBuilder. This procedure assumes you are accessing the *DB2SYSPB.SQL* script from the product CD in your computer's CD-ROM drive and the drive letter is Z.

❖ **To create the extended attribute system tables in a DB2 database outside PowerBuilder:**

- 1 Log in to the database server or gateway as the system administrator.
- 2 Insert the PowerBuilder CD-ROM into the computer's CD-ROM drive.
- 3 Use any text editor to modify *Z:\Server\DB2SYSPB.SQL* for your environment. You can do any of the following:
  - Change all instances of PBOwner to another name.

---

**Specifying SYSIBM is prohibited**

You cannot specify SYSIBM as the table owner. This is prohibited by DB2.

---

- Change all instances of database.tablespace to the appropriate value.
- Add appropriate SQL statement delimiters for the tool you are using to run the script.
- Remove comments and blank lines if necessary.

---

**PBCatalogOwner**

If you changed PBOwner to another name in the *DB2SYSPB.SQL* script, you must specify the new owner name as the value for the PBCatalogOwner DBParm parameter in your database profile. For instructions, see PBCatalogOwner in the online Help.

---

- 4 Save any changes you made to the *DB2SYSPB.SQL* script.
- 5 Execute the *DB2SYSPB.SQL* script from the database server or gateway using the SQL tool of your choice.

## PART 4

# Working with Database Connections

This part describes how to establish, manage, and troubleshoot database connections.





# Managing Database Connections

## About this chapter

After you install the necessary database software and define the database interface, you can connect to the database from PowerBuilder. Once you connect to the database, you can work with the tables and views stored in that database.

This chapter describes how to connect to a database in PowerBuilder, maintain database profiles, and share database profiles.

## Contents

Topic	Page
About database connections	120
Connecting to a database	121
Maintaining database profiles	125
Sharing database profiles	126
Importing and exporting database profiles	130
About the PowerBuilder extended attribute system tables	131

## Terminology

In this chapter, the term **database** refers to *both* of the following unless otherwise specified:

- A database or DBMS that you access with a standard database interface and appropriate driver
- A database or DBMS that you access with the appropriate native database interface

## About database connections

This section gives an overview of when database connections occur in PowerBuilder. It also tells why you should use database profiles to manage your database connections.

### When database connections occur

Connections in PowerBuilder

PowerBuilder connects to your database when you:

- Open a painter that accesses the database
- Compile or save a PowerBuilder script containing embedded SQL statements (such as a `CONNECT` statement)
- Execute an application that accesses the database
- Invoke a DataWindow control function that accesses the database while executing an application

How PowerBuilder determines the database to access

PowerBuilder *connects to the database you used last* when you open a painter that accesses the database. PowerBuilder determines which database you used last by reading a setting in the registry.

What's in this book

This book describes how to connect to your database when you are working in the PowerBuilder development environment.

For instructions on connecting to a database in a PowerBuilder application, see *Application Techniques*.

### Using database profiles

What is a database profile?

A **database profile** is a named set of parameters stored in the registry that defines a connection to a particular database in the PowerBuilder development environment.

Why use database profiles?

Creating and using database profiles is the easiest way to manage your database connections in PowerBuilder because you can:

- Select a database profile to establish or change database connections. You can easily connect to another database anytime during a PowerBuilder session. This is particularly useful if you often switch between different database connections.

- Edit a database profile to modify or supply additional connection parameters.
- Delete a database profile if you no longer need to access that data.
- Import and export profiles.

Because database profiles are created when you define your data and are stored in the registry, they have the following benefits:

- They are always available to you.
- Connection parameters supplied in a database profile are saved until you edit or delete the database profile.

## Connecting to a database

To establish or change a database connection in PowerBuilder, use a database profile. You can select the database profile for the database you want to access in the Database Profiles dialog box.

---

### **Using the Database painter to select a database profile**

You can also select the database profile for the database you want to access from the Database painter's Objects view. However, this method uses more system resources than using the Database Profile dialog box.

---

## Selecting a database profile

You can select a database profile from the Database Profiles dialog box.

❖ **To connect to a database using the Database Profiles dialog box:**

- 1 Click the Database Profile button in the PowerBar.  
*or*  
Select Tools>Database Profile from the PowerBar.

---

### Database Profile button

If your PowerBar does not include the Database Profile button, use the customize feature to add the button to the PowerBar. Having the Database Profile button on your PowerBar is useful if you frequently switch connections between different databases. For instructions on customizing toolbars, see the *User's Guide*.

---

The Database Profiles dialog box displays, listing your installed database interfaces.

---

---

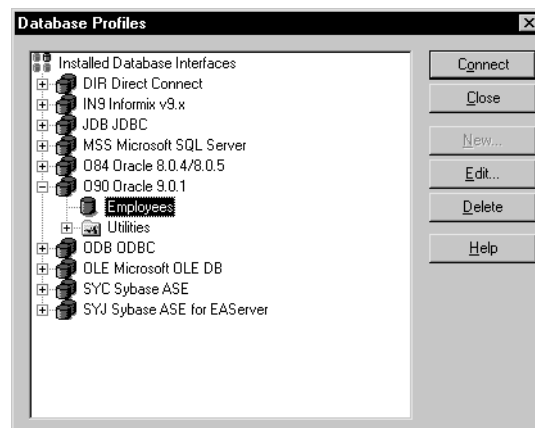
### Where the interface list comes from

When you run the Setup program, it updates the Vendors list in the registry with the interfaces you install. The Database Profiles dialog box displays the same interfaces that appear in the Vendors list.

---

- 2 Click the plus sign (+) to the left of the interface you are using.  
*or*  
Double-click the name.

The list expands to display the database profiles defined for your interface.



- 3 Select the name of the database profile you want to access and click Connect.  
*or*  
Display the pop-up menu for a database profile and select Connect.

PowerBuilder connects to the specified database and returns you to the painter workspace.

Database painter  
Objects view

You can select a database profile from the Database painter Objects view.

❖ **To connect to a database using the Database painter:**

- 1 Click the Database painter button in the PowerBar.

The Database painter displays. The Objects view lists your installed database interfaces.

---

**Where the interface list comes from**

When you run the Setup program, it updates the Vendors list in the registry with the interfaces you install. The Database painter Objects view displays the same interfaces that appear in the Vendors list.

---

- 2 Click the plus sign (+) to the left of the interface you are using.

*or*

Double-click the name.

The list expands to display the database profiles defined for your interface.

- 3 Select the name of the database profile you want to access and click the Connect button.

*or*

Display the pop-up menu for a database profile and select Connect.

## What happens when you connect

Connection  
parameters

When you connect to a database by selecting its database profile, PowerBuilder writes the profile name and its connection parameters to the registry key *HKEY\_CURRENT\_USER\Software\Sybase\PowerBuilder\9.0\DatabaseProfiles\PowerBuilder*.

Each time you connect to a different database, PowerBuilder overwrites the “most-recently used” profile name in the registry with the name for the new database connection.

What you get  
connected to

When you open a painter that accesses the database, you are connected to the database you used last. PowerBuilder determines which database this is by reading the registry.

## Specifying passwords in database profiles

As shown in the completed Database Profile Setup dialog box for Employees, your password does *not* display when you specify it in the Database Profile Setup dialog box.

However, when PowerBuilder stores the values for this profile in the registry, the actual password *does* display, albeit in encrypted form, in the DatabasePassword or LogPassword field.

Suppressing display in the profile registry entry

To suppress password display in the profile registry entry, do the following when you create a database profile.

❖ **To suppress password display in the profile registry entry:**

- 1 Select the Prompt For Database Information check box on the Connection tab in the Database Profile Setup dialog box.

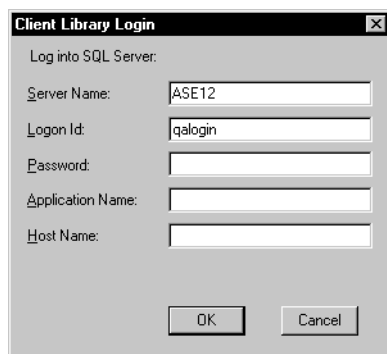
This tells PowerBuilder to prompt for any missing information when you select this profile to connect to the database.

- 2 Leave the Password box blank. Instead, specify the password in the dialog box that displays to prompt you for additional information when you connect to the database.

What happens

When you specify the password in response to a prompt instead of in the Database Profile Setup dialog box, the password does not display in the registry entry for this profile.

For example, if you do not supply a password in the Database Profile Setup - Adaptive Server Enterprise dialog box when creating a database profile, the Client Library Login dialog box displays to prompt you for the missing information:



## Using the Preview tab to connect in a PowerBuilder application

To access a database in a PowerBuilder application, you must specify the required connection parameters as properties of the Transaction object (SQLCA by default) in the appropriate script. For example, you might specify the connection parameters in the script that opens the application.

In PowerBuilder, the Preview tab in the Database Profile Setup dialog box makes it easy to generate accurate PowerScript connection syntax in the development environment for use in your PowerBuilder application script.

For instructions on using the Preview tab to help you connect in a PowerBuilder application, see the section on using Transaction objects in *Application Techniques*.

## Maintaining database profiles

You can easily edit or delete an existing database profile in PowerBuilder.

You can edit a database profile to change one or more of its connection parameters. You can delete a database profile when you no longer need to access its data. You can also change a profile using either the Database Profile dialog box or the Database painter.

What happens

When you edit or delete a database profile, PowerBuilder either updates the database profile entry in the registry or removes it.

---

### Deleting a profile for an ODBC data source

If you delete a database profile that connects to an ODBC data source, PowerBuilder does *not* delete the corresponding data source definition from the ODBC initialization file. This lets you re-create the database profile later if necessary without having to redefine the data source.

---

## Sharing database profiles

When you work in PowerBuilder, you can share database profiles among users.

---

### Sharing database profiles between PowerBuilder and InfoMaker

Since the database profiles used by both PowerBuilder and InfoMaker are stored in a common registry location, database profiles you create in either product are automatically available for use by the other, if both products are running on the same computer.

---

This section describes what you need to know to set up, use, and maintain shared database profiles in PowerBuilder.

## About shared database profiles

What you can do	You can share database profiles in the PowerBuilder development environment by specifying the location of a file containing the profiles you want to share. You specify this location in the Database Preferences property sheet in the Database painter.
Where to store a shared profile file	To share database profiles among all PowerBuilder users at your site, store a profile file on a network file server accessible to all users.
What happens	<p>When you share database profiles, PowerBuilder displays shared database profiles from the file you specify as well as those from your registry.</p> <p>Shared database profiles are read-only. You can select a shared profile to connect to a database—but you <i>cannot</i> edit, save, or delete profiles that are shared. (You can, however, make changes to a shared profile and save it on your computer, as described in “Making local changes to shared database profiles” on page 129.)</p>
How to do it	<p>To set up shared database profiles in PowerBuilder, you specify the location of the file containing shared profiles in the Database painter’s Database Preferences property sheet.</p> <p>For instructions, see “Setting up shared database profiles” next.</p>



## Setting up shared database profiles

What you do

You set up shared database profiles in the Database Preferences property sheet.

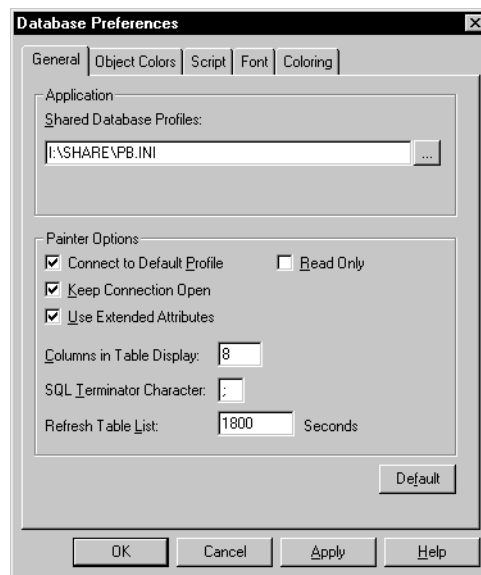
### ❖ To set up shared database profiles:

- 1 In the Database painter, select Design>Options from the menu bar.

The Database Preferences property sheet displays. If necessary, click the General tab to display the General property page.

- 2 In the Shared Database Profiles box, specify the location of the file containing the database profiles you want to share. Do this in either of the following ways:
  - Type the location (pathname) in the Shared Database Profiles box.
  - Click the Browse button to navigate to the file location and display it in the Shared Database Profiles box.

In the following example, *I:\SHARE\PB.INI* is the location of the file containing the database profiles to be shared:



- 3 Do one of the following:
  - Click Apply to apply the Shared Database Profiles setting to the current connection and all future connections without closing the Database Preferences property sheet.

- Click OK to apply the Shared Database Profiles setting to the current connection and all future connections and close the Database Preferences property sheet.

PowerBuilder saves your Shared Database Profiles setting in the registry.

## Using shared database profiles to connect

You select a shared database profile to connect to a database the same way you select a profile stored in your registry. You can select the shared profile in the Database Profiles dialog box or from the File>Connect menu.

Database Profiles dialog box

You can select and connect to a shared database profile in the Database Profiles dialog box.

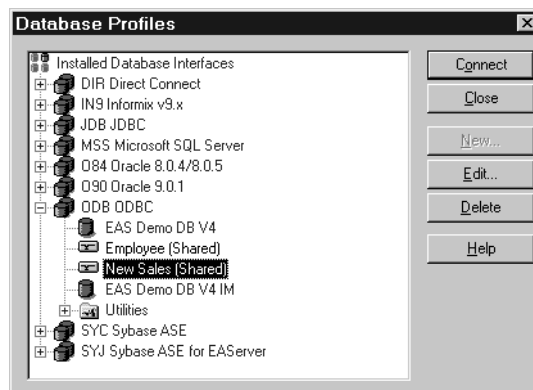
❖ **To select a shared database profile in the Database Profiles dialog box:**

- 1 Click the Database Profile button in the PowerBar.

*or*

Select Tools>Database Profile from the PowerBar.

The Database Profiles dialog box displays, listing both shared and local profiles. Shared profiles are denoted by a network icon and the word *(Shared)*.



- 2 Select the name of the shared profile you want to access and click Connect.

PowerBuilder connects to the selected database and returns you to the painter workspace.

## Making local changes to shared database profiles

Because shared database profiles can be accessed by multiple users running PowerBuilder, you should not make changes to these profiles. However, if you want to modify and save a copy of a shared database profile *for your own use*, you can edit the profile and save the modified copy in your computer's registry.

❖ **To save changes to a shared database profile in your registry:**

- 1 In the Database Profiles dialog box, select the shared profile you want to edit and click the Edit button.

Shared profiles are denoted by a network icon and the word (*Shared*).

- 2 In the Database Profile Setup dialog box that displays, edit the profile values as needed and click OK.

A message box displays, asking if you want to save a copy of the modified profile to your computer.

- 3 Click Yes.

PowerBuilder saves the modified profile in your computer's registry:

## Maintaining shared database profiles

If you maintain the database profiles for PowerBuilder at your site, you may need to update shared database profiles from time to time and make these changes available to your users.

Because shared database profiles can be accessed by multiple users running PowerBuilder, it is *not* a good idea to make changes to the profiles over a network. Instead, you should make any changes locally and then provide the updated profiles to your users.

❖ **To maintain shared database profiles at your site:**

- 1 Make and save required changes to the shared profiles on your own computer. These changes are saved in your registry.

For instructions, see “Making local changes to shared database profiles” on page 129.

- 2 Export the updated profile entries from your registry to the existing file containing shared profiles.

For instructions, see “Importing and exporting database profiles” on page 130.

- 3 If they have not already done so, have each user specify the location of the new profiles file in the Database Preferences property sheet so that they can access the updated shared profiles on their computer.

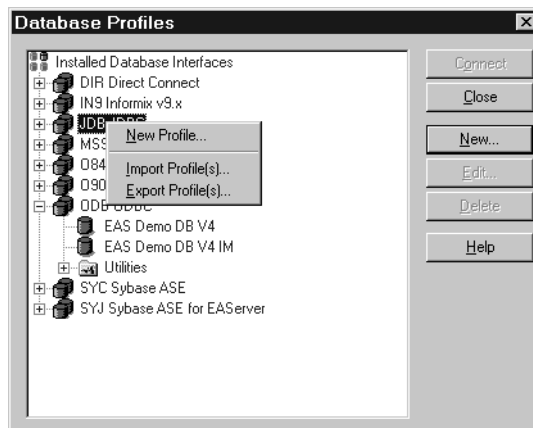
For instructions, see “Setting up shared database profiles” on page 127.

## Importing and exporting database profiles

Why do this

Each database interface provides an Import Profile(s) and an Export Profile(s) option. You can use the Import option to import a previously defined profile for use with an installed database interface. Conversely, you can use the Export option to export a defined profile for use by another user.

The ability to import and export profiles provides a way to easily move profiles between developers. It also means you no longer have to maintain a shared file to maintain profiles. It is ideal for mobile development when you cannot rely on connecting to a network to share a file.



What you do

This section describes how to import and export a profile.

❖ **To import a profile:**

- 1 Highlight a database interface and select Import Profile(s) from the pop-up menu. (In the Database painter, select Import Profile(s) from the File or pop-up menu.)
- 2 From the Select Profile File dialog box, select the file whose profiles you want to import and click Save.

- 3 Select the profile(s) you want to import from the Import Profile(s) dialog box and click OK.

The profiles are copied into your registry. If a profile with the same name already exists, you are asked if you want to overwrite it.

❖ **To export a profile:**

- 1 Highlight a database interface and select Export Profile(s) from the pop-up menu. (In the Database painter, select Export Profile(s) from the File or pop-up menu.)
- 2 Select the profile(s) you want to export from the Export Profile(s) dialog box and click OK.

The Export Profile(s) dialog box lists all profiles defined in your registry regardless of the database interface for which they were defined. By default, the profiles defined for the selected database interface are marked for export.

- 3 From the Select Profile File dialog box, select a directory and a file in which to save the exported profile(s) and click Save.

The exported profiles can be saved to a new or existing file. If saved to an existing file, the profile(s) are added to the existing profiles. If a profile with the same name already exists, you are asked if you want to overwrite it.

## About the PowerBuilder extended attribute system tables

PowerBuilder uses a collection of five system tables (formerly known as the Powersoft repository) to store extended attribute information (such as display formats, validation rules, and font information) about tables and columns in your database. You can also define extended attributes when you create or modify a table in PowerBuilder.

This section tells you how to:

- Make sure the PowerBuilder extended attribute system tables are created with the proper access rights when you log in to your database for the first time
- Display and open a PowerBuilder extended attribute system table

- Understand the kind of information stored in the PowerBuilder extended attribute system tables
- Control extended attribute system table access

## Logging on to your database for the first time

By default, PowerBuilder creates the extended attribute system tables the first time you connect to a database.

To ensure that PowerBuilder creates the extended attribute system tables with the proper access rights to make them available to all users, the first person to connect to the database with PowerBuilder must log in with the proper authority.

❖ **To ensure proper creation of the PowerBuilder extended attribute system tables:**

- Make sure the first person to connect to the database with PowerBuilder has sufficient authority to create tables and grant permissions to PUBLIC.

This means that the first person to connect to the database should log in as the database owner, database administrator, system user, system administrator, or system owner, as specified by your DBMS.

---

### **Creating the extended attribute system tables when using the DirectConnect interface**

When you are using the DirectConnect interface, the PowerBuilder extended attribute system tables are *not* created automatically the first time you connect to a database. You must run the *DB2SYSPB.SQL* script to create the system tables, as described in “Using the DB2SYSPB.SQL script” on page 115.

---

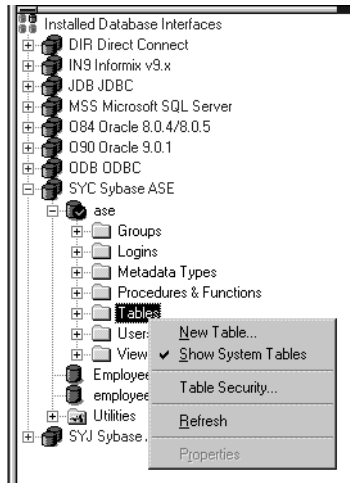
## Displaying the PowerBuilder extended attribute system tables

PowerBuilder updates the extended attribute system tables automatically whenever you change the information for a table or column. The PowerBuilder extended attribute system tables are different from the system tables provided by your DBMS.

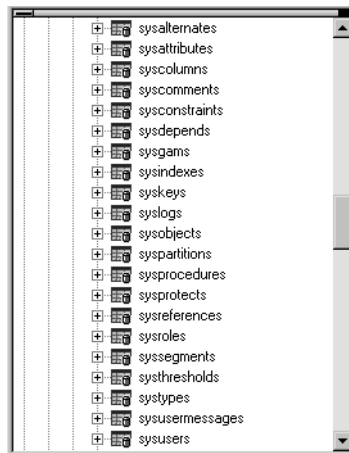
You can display and open PowerBuilder extended attribute system tables in the Database painter just like other tables.

## ❖ To display the PowerBuilder extended attribute system tables:

- 1 In the Database painter, highlight Tables in the list of database objects for the active connection and select Show System Tables from the pop-up menu.



- 2 The PowerBuilder extended attribute system tables and DBMS system tables display in the tables list, as follows:
  - **PowerBuilder system tables** The five system tables are: pbcatalog, pbcatedt, pbcatfmt, pbcattbl, and pbcattvld.
  - **DBMS system tables** The system tables supplied by the DBMS usually have a DBMS-specific prefix (such as *sys* or *dbo*).



- 3 Display the contents of a PowerBuilder system table in the Object Layout, Object Details, and/or Columns views.

For instructions, see the *User's Guide*.

---

**Do not edit the extended attribute system tables**

Do not change the values in the PowerBuilder extended attribute system tables.

---

## Contents of the extended attribute system tables

PowerBuilder stores five types of extended attribute information in the system tables as described in Table 6-1.

**Table 6-1: Extended attribute system tables**

System table	Information about	Attributes
pbcatcol	Columns	Names, comments, headers, labels, case, initial value, and justification
pbcatedt	Edit styles	Edit style names and definitions
pbcatfmt	Display formats	Display format names and definitions
pbcattbl	Tables	Name, owner, default fonts (for data, headings and labels), and comments
pbcatvld	Validation rules	Validation rule names and definitions

For more about the PowerBuilder system tables, see the Appendix in the *User's Guide*.

---

**Prefixes in system table names**

For some databases, PowerBuilder precedes the name of the system table with a default DBMS-specific prefix. For example, the names of PowerBuilder system tables have the prefix DBO in a SQL Server database (such as DBO.pbcatcol), or SYSTEM in an ORACLE database (such as SYSTEM.pbcatfmt).

The preceding table gives the base name of each system table without the DBMS-specific prefix.

---



## Controlling system table access

To control access to the PowerBuilder system tables at your site, you can specify that PowerBuilder not create or update the system tables or that the system tables be accessible only to certain users or groups.

You can control system table access by doing any of the following:

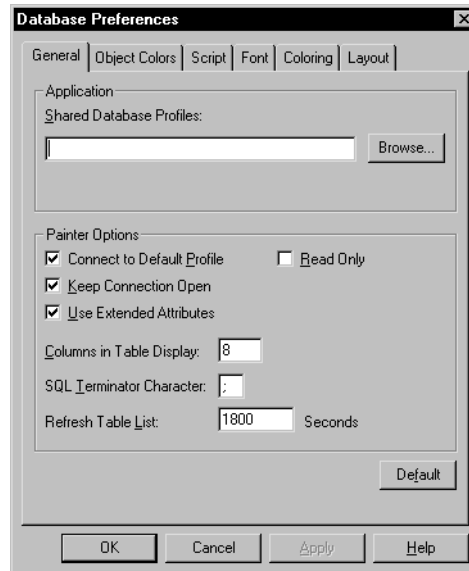
- **Setting Use Extended Attributes** Set the Use Extended Attributes database preference in the Database Preferences property sheet in the Database painter.
- **Setting Read Only** Set the Read Only database preference in the Database Preferences property sheet in the Database painter.
- **Granting permissions on the system tables** Grant explicit permissions on the system tables to users or groups at your site.

## Setting Use Extended Attributes or Read Only to control access

- ❖ **To control system table access by setting Use Extended Attributes or Read Only:**

- 1 Select Design>Options from the menu bar.

The Database Preferences dialog box displays. If necessary, click the General tab to display the General property page.



- 2 Set values for Use Extended Attributes or Read Only as follows:

Preference	What you do	Effect
Use Extended Attributes	Clear the check box	Does not create the PowerBuilder system tables if they do not exist. Instead, the painter uses the appropriate default values for extended attributes (such as headers, labels, and text color).  If the PowerBuilder system tables already exist, PowerBuilder does not use them when you create a new DataWindow object.
Read Only	Select the check box	If the PowerBuilder system tables already exist, PowerBuilder uses them when you create a new DataWindow object, <i>but does not update them</i> .  You <i>cannot</i> modify (update) information in the system tables or any other database tables in the DataWindow painter when the Read Only check box is selected.

- 3 Do one of the following:

- Click Apply to apply the preference settings to the current connection and all future connections without closing the Database Preferences property sheet.
- Click OK to apply the preference settings to the current connection and all future connections and close the Database Preferences property sheet.

PowerBuilder saves your preference settings in the registry.

## Granting permissions on system tables to control access

If your DBMS supports SQL GRANT and REVOKE statements, you can control access to the PowerBuilder system tables. The default authorization for each repository table is:

GRANT SELECT, UPDATE, INSERT, DELETE ON *table* TO PUBLIC

After the system tables are created, you can (for example) control access to them by granting SELECT authority to end users and SELECT, UPDATE, INSERT, and DELETE authority to developers. This technique offers security and flexibility that is enforced by the DBMS itself.

# Setting Additional Connection Parameters

## About this chapter

To fine-tune your database connection and take advantage of DBMS-specific features that your interface supports, you can set additional connection parameters at any time. These additional connection parameters include:

- DBParm parameters
- Database preferences

These connection parameters are described in the Database Connectivity section in the online Help.

This chapter describes how to set DBParm parameters and database preferences in PowerBuilder.

## Contents

Topic	Page
Basic steps for setting connection parameters	137
About the Database Profile Setup dialog box	138
Setting DBParm parameters	139
Setting database preferences	144

## Basic steps for setting connection parameters

This section gives basic steps for setting DBParm parameters and database preferences in PowerBuilder.

### ❖ To set DBParm parameters:

- 1 Learn how to set DBParm parameters in the development environment or PowerBuilder application script.  
See “Setting DBParm parameters” on page 139.

- 2 Determine the DBParm parameters you can set for your database interface.

For a table listing each supported database interface and the DBParm parameters you can use with that interface, see "DBParm parameters and supported database interfaces" in the online Help.

- 3 Read the description of the DBParm parameter you want to set in the online Help.
- 4 Set the DBParm parameter for your database connection.

❖ **To set database preferences:**

- 1 Learn how to set database preferences in the development environment or PowerBuilder application script.

See "Setting database preferences" on page 144.

- 2 Determine the database preferences you can set for your DBMS.

For a table listing each supported database interface and the database preferences you can use with that interface, see "DBParm parameters and supported database interfaces" in the online Help.

- 3 Read the description of the database preference you want to set in the online Help.
- 4 Set the database preference for your database connection.

## About the Database Profile Setup dialog box

The interface-specific Database Profile Setup dialog box makes it easy to set additional connection parameters in the development environment or in a PowerBuilder application script. You can:

- Supply values for connection options supported by your database interface

Each database interface has its own Database Profile Setup dialog box that includes settings only for those connection parameters supported by the interface. Similar parameters are grouped on the same tab page. The Database Profile Setup dialog box for *all* interfaces includes the Connection tab and Preview tab. Depending on the requirements and features of your interface, one or more other tab pages may also display.

- Easily set additional connection parameters in the development environment

You can specify additional connection parameters (DBParm parameters and transaction object properties) with easy-to-use check boxes, drop-down lists, and text boxes. PowerBuilder generates the proper syntax automatically when it saves your database profile in the system registry.

- Generate PowerScript connection syntax for use in your PowerBuilder application script

As you complete the Database Profile Setup dialog box in PowerBuilder, the correct PowerScript connection syntax for each selected option is generated on the Preview tab. PowerBuilder assigns the corresponding DBParm parameter or transaction object property name to each option and inserts quotation marks, commas, semicolons, and other characters where needed. You can copy the syntax you want from the Preview tab into your PowerBuilder script.

## Setting DBParm parameters

In PowerBuilder, you can set DBParm parameters by doing either of the following:

- Editing the Database Profile Setup dialog box for your connection in the development environment
- Specifying connection parameters in an application script

### Setting DBParm parameters in the development environment

Editing database profiles

To set DBParm parameters for a database connection in the PowerBuilder development environment, you must edit the database profile for that connection.

Character limit for DBParm strings

Strings containing DBParm parameters that you specify in the Database Profile Setup dialog box for your connection can be up to 999 characters in length.

This limit applies only to DBParm parameters that you set in a database profile in the development environment. DBParm strings specified in PowerBuilder scripts as properties of the Transaction object are *not* limited to a specified length.

What you do

You set DBParm parameters in the Database Profile Setup dialog box for your connection.

## Setting DBParm parameters in a PowerBuilder application script

If you are developing a PowerBuilder application that connects to a database, you must specify the required connection parameters in the appropriate script as properties of the default Transaction object (SQLCA) or a Transaction object that you create. For example, you might specify connection parameters in the script that opens the application.

One of the connection parameters you may want to specify in a script is DBParm. You can do this by:

- *(Recommended)* Copying PowerScript DBParm syntax from the Preview tab in the Database Profile Setup dialog box into your script
- Coding PowerScript to set values for the DBParm property of the Transaction object
- Reading DBParm values from an external text file

## Copying DBParm syntax from the Preview tab

The easiest way to specify DBParm parameters in a PowerBuilder application script is to copy the PowerScript DBParm syntax from the Preview tab in the Database Profile Setup dialog box into your script, modifying the default Transaction object name (SQLCA) if necessary.

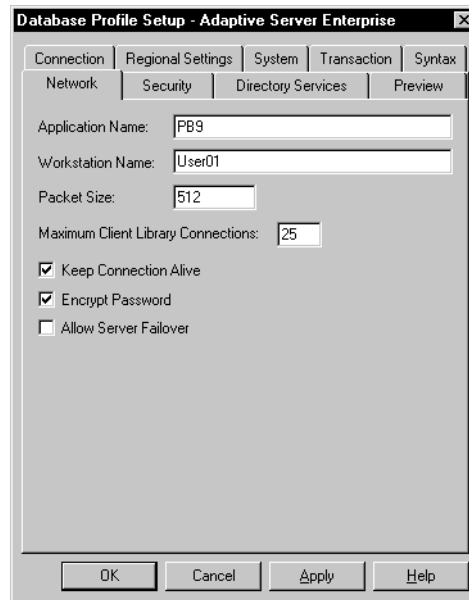
As you set DBParm parameters in the Database Profile Setup dialog box in the development environment, PowerBuilder generates the correct connection syntax on the Preview tab. Therefore, copying the syntax directly from the Preview tab ensures that you use the correct PowerScript DBParm syntax in your script.

❖ **To copy DBParm syntax from the Preview tab into your script:**

- 1 On one or more tab pages in the Database Profile Setup dialog box for your connection, supply values for any DBParm parameters you want to set.

For instructions, see “Setting DBParm parameters in the development environment” on page 139.

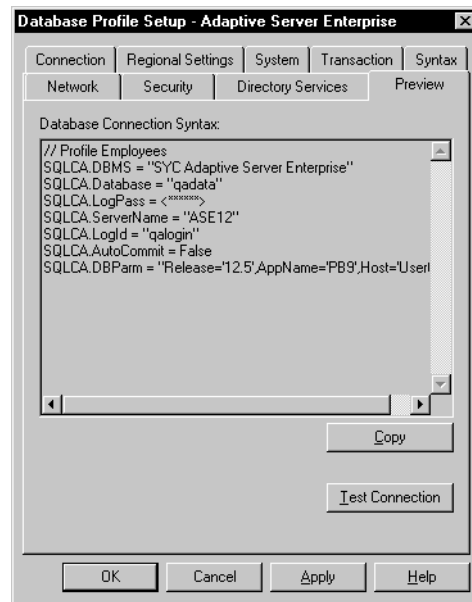
For example, the Network tab in the Database Profile Setup - Adaptive Server Enterprise dialog box contains settings for network-related DBParm parameters that the interface supports. This example shows nondefault settings for Application Name (corresponds to the AppName DBParm parameter) and Workstation Name (corresponds to the Host DBParm parameter):



For information about the DBParm parameters for your interface and the values to supply, click Help.

- 2 Click Apply to save your changes to the current tab without closing the Database Profile Setup dialog box.
- 3 Click the Preview tab.

The correct PowerScript DBParm syntax for each selected option displays in the Database Connection Syntax box. For example:



- 4 Select one or more lines of text in the Database Connection Syntax box and click Copy.  
PowerBuilder copies the selected text to the clipboard.
- 5 Click OK to close the Database Profile Setup dialog box.
- 6 Paste the selected text from the Preview tab into your script, modifying the default Transaction object name (SQLCA) if necessary.

## Coding PowerScript to set values for the DBParm property

Another way to specify connection parameters in a script is by coding PowerScript to assign values to properties of the Transaction object. PowerBuilder uses a special nonvisual object called a **Transaction object** to communicate with the database. The default Transaction object is named SQLCA, which stands for SQL Communications Area.

SQLCA has 15 properties, 10 of which are used to connect to your database. One of the 10 connection properties is DBParm. DBParm contains DBMS-specific parameters that let your application take advantage of various features supported by the database interface.



❖ **To set values for the DBParm property in a PowerBuilder script:**

- 1 Open the application script in which you want to specify connection parameters.

For instructions, see the *User's Guide*.

- 2 Use the following PowerScript syntax to specify DBParm parameters. Make sure you separate the DBParm parameters with commas, and enclose the entire DBParm string in double quotes.

**SQLCA.dbParm = "parameter\_1, parameter\_2, parameter\_n"**

For example, the following statement in a PowerBuilder script sets the DBParm property for an ODBC data source named Sales. In this example, the DBParm property consists of two parameters: ConnectString and Async.

```
SQLCA.dbParm="ConnectString='DSN=Sales;UID=PB;
PWD=xyz',Async=1"
```

- 3 Compile the PowerBuilder script to save your changes.

For instructions, see the *User's Guide*.

## Reading DBParm values from an external text file

As an alternative to setting the DBParm property in a PowerBuilder application script, you can use the PowerScript ProfileString function to read DBParm values from a specified section of an external text file, such as an application-specific initialization file.

❖ **To read DBParm values from an external text file:**

- 1 Open the application script in which you want to specify connection parameters.

For instructions, see the *User's Guide*.

- 2 Use the following PowerScript syntax to specify the ProfileString function with the SQLCA.DBParm property:

**SQLCA.dbParm = ProfileString ( file, section, key,  
default )**

For example, the following statement in a PowerBuilder script reads the DBParm values from the [Database] section of the *APP.INI* file:

```
SQLCA.dbParm=ProfileString("APP.INI", "Database",  
"dbParm", "" )
```

- 3 Compile the script to save your changes.

For instructions, see the *User's Guide*.

## Setting database preferences

How to set

The way you set connection-related database preferences in PowerBuilder varies, as summarized in the following table (AutoCommit and Lock are the only database preferences that you can set in a PowerBuilder application script).

**Table 7-1: Database preferences and where they can be set**

Database preference	Set in development environment by editing	Set in PowerBuilder application by editing
AutoCommit	Database Profile Setup dialog box for your connection	Application script
Lock	Database Profile Setup dialog box for your connection	Application script
Shared Database Profiles	Database Preferences property sheet	—
Connect to Default Profile	Database Preferences property sheet	—
Read Only	Database Preferences property sheet	—
Keep Connection Open	Database Preferences property sheet	—
Use Extended Attributes	Database Preferences property sheet	—
SQL Terminator Character	Database Preferences property sheet	—

The following sections give the steps for setting database preferences in the development environment and (for AutoCommit and Lock) in a PowerBuilder application script.

For more information

For information about using a specific database preference, see its description in the online Help.

## Setting database preferences in the development environment

There are two ways to set database preferences in the PowerBuilder development environment on *all* supported development platforms, depending on the preference you want to set:

- Set AutoCommit and Lock (Isolation Level) in the Database Profile Setup dialog box for your connection
- Set all other database preferences in the Database Preferences property sheet in the Database painter

### Setting AutoCommit and Lock in the database profile

The AutoCommit and Lock (Isolation Level) preferences are properties of the default Transaction object, SQLCA. For AutoCommit and Lock to take effect in the PowerBuilder development environment, you must specify them *before* you connect to a database. Changes to these preferences after the connection occurs have no effect on the current connection.

To set AutoCommit and Lock before PowerBuilder connects to your database, you specify their values in the Database Profile Setup dialog box for your connection.

❖ **To set AutoCommit and Lock (Isolation Level) in a database profile:**

- 1 Display the Database Profiles dialog box.
- 2 Click the plus sign (+) to the left of the interface you are using.  
*or*  
Double-click the interface name.

The list expands to display the database profiles defined for your interface.

- 3 Select the name of the profile you want and click Edit.

The Database Profile Setup dialog box for the selected profile displays.

- 4 On the Connection tab page, supply values for one or both of the following:

- **Isolation Level** If your database supports the use of locking and isolation levels, select the isolation level you want to use for this connection from the Isolation Level drop-down list. (The Isolation Level drop-down list contains valid lock values for your interface.)

- AutoCommit Mode** The setting of AutoCommit controls whether PowerBuilder issues SQL statements outside (True) or inside (False) the scope of a transaction. *If your database supports it*, select the AutoCommit Mode check box to set AutoCommit to True or clear the AutoCommit Mode check box (the default) to set AutoCommit to False.

For example, in addition to values for basic connection parameters (Server, Login ID, Password, and Database), the Connection tab page for the following Sybase Adaptive Server Enterprise profile named Sales shows nondefault settings for Isolation Level and AutoCommit Mode.

- (Optional) In PowerBuilder, click the Preview tab if you want to see the PowerScript connection syntax generated for Lock and AutoCommit.

PowerBuilder generates correct PowerScript connection syntax for each option you set in the Database Profile Setup dialog box. You can copy this syntax directly into a PowerBuilder application script.

For instructions, see “Copying DBParm syntax from the Preview tab” on page 140.

- Click OK to close the Database Profile Setup dialog box.

PowerBuilder saves your settings in the database profile entry in the registry.

For example, here is the database profile entry for Sales:

```
[Default]      [value not set]
AutoCommit     "TRUE"
Database       "Sales"
DatabasePassword "00"
DBMS           "SYC Sybase Adaptive Server Enterprise"
DbParm        "Release='12.5'"
Lock          "3"
LogId         "login"
LogPassword    "pass"
Prompt        "0"
ServerName     "Sales"
```

## Setting preferences in the Database Preferences property sheet

To set the following connection-related database preferences, complete the Database Preferences property sheet in the PowerBuilder Database painter:

- Shared Database Profiles
- Connect to Default Profile
- Read Only
- Keep Connection Open
- Use Extended Attributes
- SQL Terminator Character

---

### Other database preferences

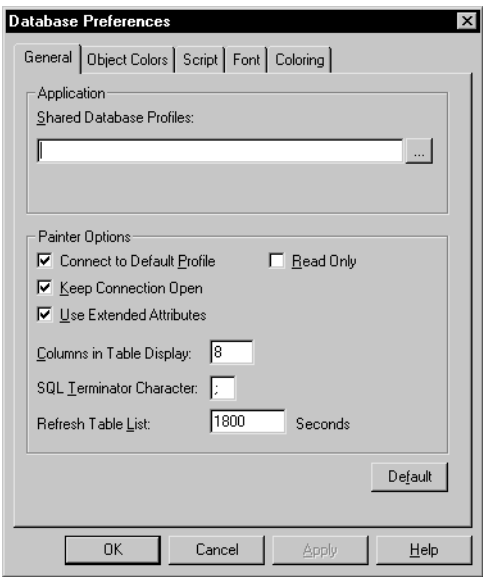
The Database Preferences property sheet also lets you set other database preferences that affect the behavior of the Database painter itself. For information about the other preferences you can set in the Database Preferences property sheet, see the *User's Guide*.

---

#### ❖ To set connection-related preferences in the Database Preferences property sheet:

- 1 Open the Database painter.
- 2 Select Design>Options from the menu bar.

The Database Preferences property sheet displays. If necessary, click the General tab to display the General property page.



- 3 Specify values for one or more of the connection-related database preferences in the following table.

**Table 7-2: Connection-related database preferences**

Preference	Description	For details, see
Shared Database Profiles	Specifies the pathname of the file containing the database profiles you want to share. You can type the pathname or click Browse to display it.	“Sharing database profiles” on page 126
Connect to Default Profile	Controls whether the Database painter establishes a connection to a database using a default profile when the painter is invoked. If not selected, the Database painter opens without establishing a connection to a database.	Connect to Default Profile in online Help

Preference	Description	For details, see
Read Only	<p>Specifies whether PowerBuilder should update the extended attribute system tables and any other tables in your database. Select or clear the Read Only check box as follows:</p> <ul style="list-style-type: none"> <li>• <b>Select the check box</b> Does not update the extended attribute system tables or any other tables in your database. You <i>cannot</i> modify (update) information in the extended attribute system tables or any other database tables from the DataWindow painter when the Read Only check box is selected.</li> <li>• <b>Clear the check box</b> (Default) Updates the extended attribute system tables and any other tables in your database.</li> </ul>	Read Only in the online Help
Keep Connection Open	<p>When you connect to a database in PowerBuilder without using a database profile, specifies when PowerBuilder closes the connection. Select or clear the Keep Connection Open check box as follows:</p> <ul style="list-style-type: none"> <li>• <b>Select the check box</b> (Default) Stays connected to the database throughout your session and closes the connection when you exit</li> <li>• <b>Clear the check box</b> Opens the connection only when a painter requests it and closes the connection when you close a painter or finish compiling a script</li> </ul> <hr/> <p><b>Not used with profile</b> This preference has no effect when you connect using a database profile.</p> <hr/>	Keep Connection Open in the online Help

Preference	Description	For details, see
Use Extended Attributes	<p>Specifies whether PowerBuilder should create and use the extended attribute system tables. Select or clear the Use Extended Attributes check box as follows:</p> <ul style="list-style-type: none"> <li>• <b>Select the check box</b> (Default) Creates and uses the extended attribute system tables</li> <li>• <b>Clear the check box</b> Does <i>not</i> create the extended attribute system tables</li> </ul>	Use Extended Attributes in the online Help
SQL Terminator Character	<p>Specifies the SQL statement terminator character used in the ISQL view in the Database painter in PowerBuilder.</p> <p>The default terminator character is a semicolon (;). If you are creating stored procedures and triggers in the ISQL view of the database painter, change the terminator character to one that you do not expect to use in the stored procedure or trigger syntax for your DBMS. A good choice is the backquote (`) character.</p>	SQL Terminator Character in the online Help

4 Do one of the following:

- Click Apply to apply the preference settings to the current connection without closing the Database Preferences property sheet.
- Click OK to apply the preference settings to the current connection and close the Database Preferences property sheet.

PowerBuilder saves your preference settings in the database section of *PB.INI*.

## Setting AutoCommit and Lock in a PowerBuilder application script

If you are developing a PowerBuilder application that connects to a database, you must specify the required connection parameters in the appropriate script as properties of the default Transaction object (SQLCA) or a Transaction object that you create. For example, you might specify connection parameters in the script that opens the application.



AutoCommit and Lock are properties of SQLCA. As such, they are the *only* database preferences that you can set in a PowerBuilder script. You can do this by:

- (*Recommended*) Copying PowerScript syntax for AutoCommit and Lock from the Preview tab in the Database Profile Setup dialog box into your script
- Coding PowerScript to set values for the AutoCommit and Lock properties of the Transaction object
- Reading AutoCommit and Lock values from an external text file

For more about using Transaction objects to communicate with a database in a PowerBuilder application, see *Application Techniques*.

## Copying AutoCommit and Lock syntax from the Preview tab

The easiest way to specify AutoCommit and Lock in a PowerBuilder application script is to copy the PowerScript syntax from the Preview tab in the Database Profile Setup dialog box into your script, modifying the default Transaction object name (SQLCA) if necessary.

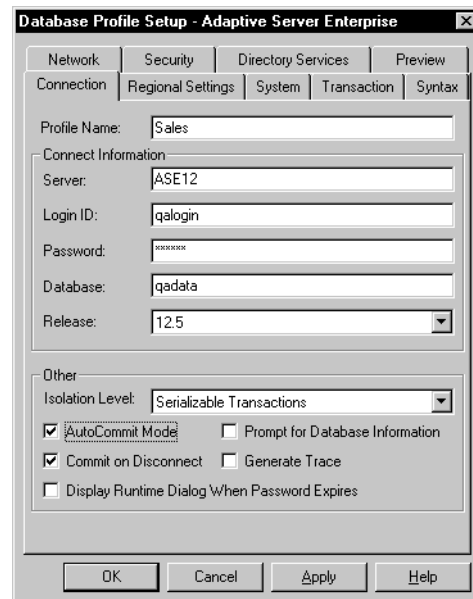
As you complete the Database Profile Setup dialog box in the development environment, PowerBuilder generates the correct connection syntax on the Preview tab for each selected option. Therefore, copying the syntax directly from the Preview tab ensures that you use the correct PowerScript syntax in your script.

### ❖ To copy AutoCommit and Lock syntax from the Preview tab into your script:

- 1 On the Connection tab in the Database Profile Setup dialog box for your connection, supply values for AutoCommit and Lock (Isolation Level) as required.

For instructions, see “Setting AutoCommit and Lock in the database profile” on page 145.

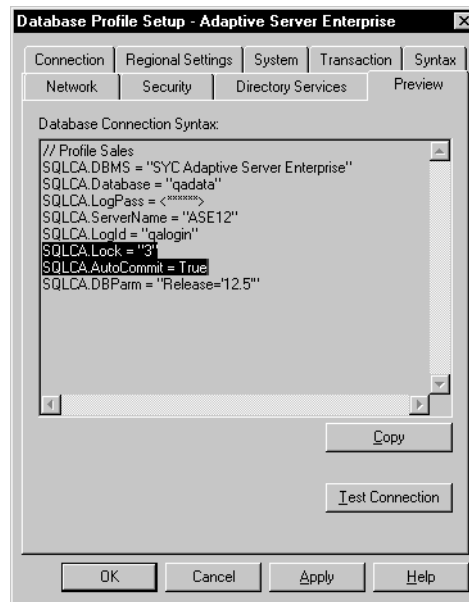
For example, in addition to values for basic connection parameters (Server, Login ID, Password, and Database), the Connection tab for the following Adaptive Server profile named Sales shows nondefault settings for Isolation Level and AutoCommit Mode.



For information about the DBParm parameters for your interface and the values to supply, click Help.

- 2 Click Apply to save your changes to the current tab without closing the Database Profile Setup dialog box.
- 3 Click the Preview tab.

The correct PowerScript syntax for each selected option displays in the Database Connection Syntax box. For example:



- 4 Select one or more lines of text in the Database Connection Syntax box and click Copy.

PowerBuilder copies the selected text to the clipboard.

- 5 Click OK to close the Database Profile Setup dialog box.
- 6 Paste the selected text from the Preview tab into your script, modifying the default Transaction object name (SQLCA) if necessary.

## Coding PowerScript to set values for AutoCommit and Lock

Another way to specify the AutoCommit and Lock properties in a script is by coding PowerScript to assign values to the AutoCommit and Lock properties of the Transaction object. PowerBuilder uses a special nongraphic object called a **Transaction object** to communicate with the database. The default Transaction object is named SQLCA, which stands for SQL Communications Area.

SQLCA has 15 properties, 10 of which are used to connect to your database. Two of the connection properties are AutoCommit and Lock, which you can set as described in the following procedure.

❖ **To set the AutoCommit and Lock properties in a PowerBuilder script:**

- 1 Open the application script in which you want to set connection properties.

For instructions, see the *User's Guide*.

- 2 Use the following PowerScript syntax to set the AutoCommit and Lock properties. (This syntax assumes you are using the default Transaction object SQLCA, but you can also define your own Transaction object.)

**SQLCA.AutoCommit = value**

**SQLCA.Lock = "value"**

For example, the following statements in a PowerBuilder script use the default Transaction object SQLCA to connect to a Sybase Adaptive Server Enterprise database named Test. SQLCA.AutoCommit is set to True and SQLCA.Lock is set to isolation level 3 (Serializable transactions).

```
SQLCA.DBMS      = "SYC"
SQLCA.Database  = "Test"
SQLCA.LogID     = "Frans"
SQLCA.LogPass   = "xyyyzz"
SQLCA.ServerName = "HOST1"
SQLCA.AutoCommit = True
SQLCA.Lock      = "3"
```

For more information, see AutoCommit or Lock in the online Help.

- 3 Compile the script to save your changes.

For instructions, see the *User's Guide*.

## Reading AutoCommit and Lock values from an external text file

As an alternative to setting the AutoCommit and Lock properties in a PowerBuilder application script, you can use the PowerScript ProfileString function to read the AutoCommit and Lock values from a specified section of an external text file, such as an application-specific initialization file.

❖ **To read AutoCommit and Lock values from an external text file:**

- 1 Open the application script in which you want to set connection properties.

For instructions, see the *User's Guide*.

- 2 Use the following PowerShell syntax to specify the ProfileString function with the SQLCA.Lock property:

**SQLCA.Lock = ProfileString ( file, section, key, default )**

The AutoCommit property is a boolean, so you need to convert the string returned by ProfileString to a boolean. For example, the following statements in a PowerBuilder script read the AutoCommit and Lock values from the [Database] section of the *APP.INI* file:

```
string ls_string
ls_string=Upper(ProfileString("APP.INI","Database",
    "Autocommit",""))
if ls_string = "TRUE" then
    SQLCA.Autocommit = TRUE
else
    SQLCA.Autocommit = FALSE
end if
SQLCA.Lock=ProfileString("APP.INI","Database",
    "Lock","")
```

- 3 Compile the script to save your changes.

## Getting values from the registry

If the AutoCommit and Lock values are stored in an application settings key in the registry, use the RegistryGet function to obtain them. For example:

```
string ls_string
RegistryGet("HKEY_CURRENT_USER\Software\MyCo\MyApp", &
    "Autocommit", RegString!, ls_string)
if Upper(ls_string) = "TRUE" then
    SQLCA.Autocommit = TRUE
else
    SQLCA.Autocommit = FALSE
end if
RegistryGet("HKEY_CURRENT_USER\Software\MyCo\MyApp", &
    "Lock", RegString!, ls_string)
```



# Troubleshooting Your Connection

## About this chapter

This chapter describes how to troubleshoot your database connection in PowerBuilder by using the following tools:

- Database Trace
- ODBC Driver Manager Trace
- JDBC Driver Manager Trace

## Contents

Topic	Page
Overview of troubleshooting tools	157
Using the Database Trace tool	158
Using the ODBC Driver Manager Trace	172
Using the JDBC Driver Manager Trace	184

## Overview of troubleshooting tools

When you use PowerBuilder, there are several tools available to trace your database connection in order to troubleshoot problems:

**Table 8-1: Database trace tools**

Use this tool	To trace a connection to
Database Trace	Any database that PowerBuilder accesses through one of the database interfaces
ODBC Driver Manager Trace	An ODBC data source only
JDBC Driver Manager Trace	A JDBC database only

## Using the Database Trace tool

The section describes how to use the Database Trace tool.

### About the Database Trace tool

The Database Trace tool records the internal commands that PowerBuilder executes while accessing a database. You can trace a database connection in the development environment or in a PowerBuilder application that connects to a database.

PowerBuilder writes the output of Database Trace to a log file named *PBTRACE.LOG* (by default) or to a nondefault log file that you specify. When you enable database tracing for the first time, PowerBuilder creates the log file on your computer. Tracing continues until you disconnect from the database.

---

#### Using the Database Trace tool with one connection

You can use the Database Trace tool for only one DBMS at a time and for one database connection at a time.

For example, if your application connects to both an ODBC data source and an Adaptive Server Enterprise database, you can trace either the ODBC connection or the Adaptive Server Enterprise connection, but not both connections at the same time.

---

### How you can use the Database Trace tool

You can use information from the Database Trace tool to help you understand what PowerBuilder is doing *internally* when you work with your database. Examining the information in the log file can help you:

- Understand how PowerBuilder interacts with your database
- Identify and resolve problems with your database connection
- Provide useful information to Technical Support if you call them for help with your database connection

If you are familiar with PowerBuilder and your DBMS, you can use the information in the log to help troubleshoot connection problems on your own.



If you are less experienced or need help, run the Database Trace tool *before* you call Technical Support. You can then report or send the results of the trace to the Technical Support representative who takes your call.

## Location of the Database Trace log

### *PBTRACE.LOG*

By default, PowerBuilder writes output of the Database Trace tool to a file named *PBTRACE.LOG* in your Windows directory.

### Nondefault log file

If you prefer, you can specify a nondefault name and location for the log file when you use Database Trace. For instructions, see “Specifying a nondefault Database Trace log” on page 167.

## Contents of the Database Trace log

The Database Trace tool records the following information in the log file when you trace a database connection:

- Parameters used to connect to the database
- Time to perform each database operation (in milliseconds)
- The internal commands executed to retrieve and display table and column information from your database. Examples include:
  - Preparing and executing SQL statements such as SELECT, INSERT, UPDATE, and DELETE
  - Getting column descriptions
  - Fetching table rows
  - Binding user-supplied values to columns (if your database supports bind variables)
  - Committing and rolling back database changes
- Disconnecting from the database
- Shutting down the database interface

## Format of the Database Trace log

The specific content of the Database Trace log file depends on the database you are accessing and the operations you are performing. However, the log uses the following basic format to display output:

*COMMAND: (time)*  
*{additional\_information}*

Parameter	Description
<i>COMMAND</i>	The internal command that PowerBuilder executes to perform the database operation.
<i>time</i>	The number of milliseconds it takes PowerBuilder to perform the database operation. The precision used depends on your operating system's timing mechanism.
<i>additional_information</i>	(Optional) Additional information about the command. The information provided depends on the database operation.

### Example

The following portion of the log file shows the commands PowerBuilder executes to fetch two rows from a database table:

```
FETCH NEXT: (77 MilliSeconds)
    dept_id=    dept_name=Business Services
FETCH NEXT: (4 MilliSeconds)
    dept_id=    dept_name=Corporate Management
```

For a more complete example of Database Trace output, see “Sample Database Trace output” on page 170.

## Starting the Database Trace tool

By default, the Database Trace tool is turned off in PowerBuilder. You can start it in the PowerBuilder development environment or in a PowerBuilder application to trace your database connection.

---

### Turning tracing on and off

To turn tracing on or off you must reconnect. Setting and resetting are not sufficient.

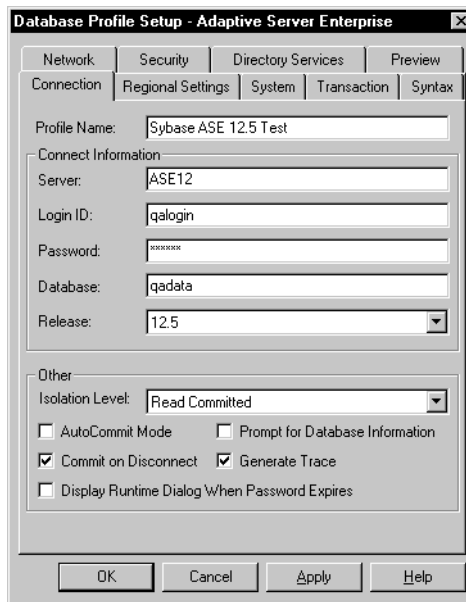
---

## Starting Database Trace in the development environment

To start the Database Trace tool in the PowerBuilder development environment, edit the database profile for the connection you want to trace, as described in the following procedure.

❖ **To start the Database Trace tool by editing a database profile:**

- 1 Open the Database Profile Setup dialog box for the connection you want to trace.
- 2 On the Connection tab, select the Generate Trace check box and click OK or Apply. (The Generate Trace check box is located on the System tab in the OLE DB Database Profile Setup dialog box.)



The Database Profiles dialog box displays with the name of the edited profile highlighted.

For example, here is the relevant portion of the database profile entry for Adaptive Server 12.5 Test. The setting that starts Database Trace is DBMS:

```
[Default]          [value not set]
AutoCommit         "FALSE"
Database           "qadata"
DatabasePassword   "00"
DBMS                "TRACE SYC Adaptive Server Enterprise"
```

DbParm	"Release='12.5' "
Lock	" "
LogId	"qalotin"
LogPassword	"00171717171717"
Prompt	"FALSE"
ServerName	"Host125 "
UserID	" "

- 3 Click Connect in the Database Profiles dialog box to connect to the database.

A message box displays stating that database tracing is enabled and indicating where PowerBuilder will write the output. (By default, PowerBuilder writes Database Trace output to a log file named *PBTRACE.LOG*.)

For instructions on specifying your own name and location for the Database Trace log file, see “Specifying a nondefault Database Trace log” on page 167.

- 4 Click OK.

PowerBuilder connects to the database and starts tracing the connection.

## Starting Database Trace in a PowerBuilder application

In a PowerBuilder application that connects to a database, you must specify the required connection parameters in the appropriate script. For example, you might specify them in the script that opens the application.

To trace a database connection in a PowerBuilder script, you specify the name of the DBMS preceded by the word *trace* and a single space. You can do this by:

- Copying the PowerScript DBMS trace syntax from the Preview tab in the Database Profile Setup dialog box into your script
- Coding PowerScript to set a value for the DBMS property of the Transaction object
- Reading the DBMS value from an external text file

For more about using Transaction objects to communicate with a database in a PowerBuilder application, see *Application Techniques*.

**Copying DBMS trace syntax from the Preview tab**

One way to start Database Trace in a PowerBuilder application script is to copy the PowerScript DBMS trace syntax from the Preview tab in the Database Profile Setup dialog box into your script, modifying the default Transaction object name (SQLCA) if necessary.

As you complete the Database Profile Setup dialog box in the development environment, PowerBuilder generates the correct connection syntax on the Preview tab for each selected option, including Generate Trace. Therefore, copying the syntax directly from the Preview tab ensures that it is accurate in your script.

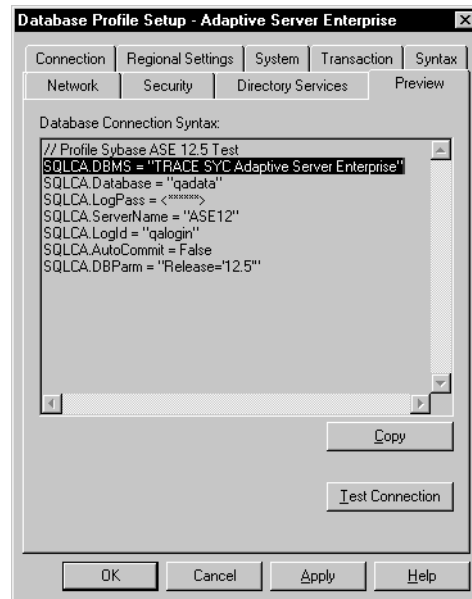
❖ **To copy DBMS trace syntax from the Preview tab into your script:**

- 1 On the Connection tab (or System tab in the case of OLE DB) in the Database Profile Setup dialog box for your connection, select the Generate Trace check box to turn on Database Trace.

For instructions, see “Starting Database Trace in the development environment” on page 161.

- 2 Click Apply to save your changes to the Connection tab without closing the Database Profile Setup dialog box.
- 3 Click the Preview tab.

The correct PowerScript connection syntax for the Generate Trace and other selected options displays in the Database Connection Syntax box. For example:



- 4 Select the `SQLCA.DBMS` line and any other syntax you want to copy to your script and click Copy.
- PowerBuilder copies the selected text to the clipboard.
- 5 Click OK to close the Database Profile Setup dialog box.
- 6 Paste the selected text from the Preview tab into your script, modifying the default Transaction object name (`SQLCA`) if necessary.

### Coding PowerScript to set a value for the DBMS property

Another way to start the Database Trace tool in a PowerBuilder script is to specify it as part of the DBMS property of the Transaction object. The **Transaction object** is a special nonvisual object that PowerBuilder uses to communicate with the database. The default Transaction object is named `SQLCA`, which stands for SQL Communications Area.

`SQLCA` has 15 properties, 10 of which are used to connect to your database. One of the 10 connection properties is `DBMS`. The `DBMS` property contains the name of the database to which you want to connect.

❖ **To start the Database Trace tool by specifying the DBMS property:**

- Use the following PowerScript syntax to specify the DBMS property. (This syntax assumes you are using the default Transaction object SQLCA, but you can also define your own Transaction object.)

**SQLCA.DBMS = "trace DBMS\_name"**

For example, the following statements in a PowerBuilder script set the SQLCA properties required to connect to an Adaptive Server database named Test. The keyword *trace* in the DBMS property indicates that you want to trace the database connection.

```
SQLCA.DBMS           = "trace SYC"
SQLCA.database       = "Test"
SQLCA.logId          = "Frans"
SQLCA.LogPass        = "xxyyzz"
SQLCA.ServerName     = "Tomlin"
```

**Reading the DBMS value from an external text file or the registry**

As an alternative to setting the DBMS property in your PowerBuilder application script, you can use the PowerScript ProfileString function to read the DBMS value from a specified section of an external text file, such as an application-specific initialization file, or from an application settings key in the registry.

The following procedure assumes that the DBMS value read from the database section in your initialization file or uses the following syntax to enable database tracing:

**DBMS = trace DBMS\_name**

❖ **To start the Database Trace tool by reading the DBMS value from an external text file:**

- Use the following PowerScript syntax to specify the ProfileString function with the DBMS property:

**SQLCA.DBMS = ProfileString(file, section, variable, default\_value)**

For example, the following statement in a PowerBuilder script reads the DBMS value from the [Database] section of the APP.INI file:

```
SQLCA.DBMS=ProfileString("APP.INI", "Database",
    "DBMS", " ")
```

For how to get a value from a registry file instead, see “Getting values from the registry” on page 155.

## Stopping the Database Trace tool

Once you start tracing a particular database connection, PowerBuilder continues sending trace output to the log until you do one of the following:

- Reconnect to the same database with tracing stopped
- Connect to another database for which you have not enabled tracing

## Stopping Database Trace in the development environment

- ❖ **To stop the Database Trace tool by editing a database profile:**
  - 1 In the Database Profile Setup dialog box for the database you are tracing, clear the Generate Trace check box on the Connection tab.
  - 2 Click OK in the Database Profile Setup dialog box.

The Database Profiles dialog box displays with the name of the edited profile highlighted.
  - 3 Right-click on the connected database and select Re-connect from the dropdown menu in the Database Profiles dialog box.

PowerBuilder connects to the database and stops tracing the connection.

## Stopping Database Trace in a PowerBuilder application

To stop Database Trace in a PowerBuilder application script, you must delete the word *trace* from the DBMS property. You can do this by:

- Editing the value of the DBMS property of the Transaction object
- Reading the DBMS value from an external text file

You must reconnect for the change to take effect.

### Editing the DBMS property

- ❖ **To stop Database Trace by editing the DBMS value in a PowerBuilder script:**
  - Delete the word *trace* from the DBMS connection property in your application script.



For example, here is the DBMS connection property in a PowerBuilder script that enables the Database Trace. (This syntax assumes you are using the default Transaction object SQLCA, but you can also define your own Transaction object.)

```
SQLCA.DBMS = "trace SYC"
```

Here is how the same DBMS connection property should look after you edit it to stop tracing:

```
SQLCA.DBMS = "SYC"
```

### Reading the DBMS value from an external text file

As an alternative to editing the DBMS property in your PowerBuilder application script, you can use the PowerScript ProfileString function to read the DBMS value from a specified section of an external text file, such as an application-specific initialization file.

This assumes that the DBMS value read from your initialization file *does not include* the word *trace*, as shown in the preceding example in "Editing the DBMS property."

## Specifying a nondefault Database Trace log

In the PowerBuilder development environment, you can specify a nondefault name and location for the log file when you use Database Trace.

What you can do

By specifying a nondefault Database Trace log to use in the development environment, you can:

- Control where PowerBuilder writes the output of the Database Trace tool
- Give the log file a name and location that best meets the development needs at your site

By default, PowerBuilder writes Database Trace output to a log file named *PBTRACE.LOG* located in your Windows directory. You can override this default in the development environment by editing your PowerBuilder initialization file.

How to do it

❖ **To specify a nondefault Database Trace log file:**

- 1 Open the PowerBuilder initialization file for editing.

You can use the File Editor (in PowerBuilder) or any text editor (outside PowerBuilder).

- 2 Create an entry named DBTraceFile in the [Database] section of the initialization file, using the following syntax to specify a nondefault log file:

**DBTraceFile=log\_file\_pathname**

For example:

```
[Database]
...
DBTraceFile=c:\temp\mydbtrce.log
```

- 3 Save your changes to the initialization file.

The next time you use the Database Trace tool to trace a connection in the development environment, PowerBuilder writes the output to the log file you specified instead of to the default *PBTRACE.LOG* file.

For instructions on starting Database Trace, see “Starting Database Trace in the development environment” on page 161.

## Using the Database Trace log

PowerBuilder writes the output of the Database Trace tool to a file named *PBTRACE.LOG* (by default) or to a nondefault log file that you specify. To use the trace log, you can do the following anytime:

- View the Database Trace log with any text editor
- Annotate the Database Trace log with your own comments
- Delete the Database Trace log or clear its contents when it becomes too large

For information about where to find *PBTRACE.LOG*, see “Location of the Database Trace log” on page 159.

For instructions about specifying your own Database Trace log to use in the development environment, see “Specifying a nondefault Database Trace log” on page 167.

## Viewing the Database Trace log

You can display the contents of the log file anytime during a PowerBuilder session.

❖ **To view the contents of the log file:**

- Open the log file in one of the following ways:
  - Use the File Editor in PowerBuilder. (For instructions, see the *User's Guide*.)
  - Use any text editor outside PowerBuilder.

---

### Leaving the log file open

If you leave the log file open as you work in PowerBuilder, the Database Trace tool *does not update* the log.

---

## Annotating the Database Trace log

When you use the Database Trace log as a troubleshooting tool, it may be helpful to add your own comments or notes to the file. For example, you can specify the date and time of a particular connection, the versions of database server and client software you used, or any other useful information.

❖ **To annotate the log file:**

- 1 Open the *PBTRACE.LOG* file in one of the following ways:
  - Use the File Editor in PowerBuilder. (For instructions, see the *User's Guide*.)
  - Use any text editor outside PowerBuilder.
- 2 Edit the log file with your comments.
- 3 Save your changes to the log file.

## Deleting or clearing the Database Trace log

Each time you connect to a database with tracing enabled, PowerBuilder appends the trace output of your connection to the existing log. As a result, the log file can become very large over time, especially if you frequently enable tracing when connected to a database.

❖ **To keep the size of the log file manageable:**

- Do either of the following periodically:
  - Open the log file, clear its contents, and save the empty file.  
 Provided that you use the default *PBTRACE.LOG* or the same nondefault file the next time you connect to a database with tracing enabled, PowerBuilder will write to this empty file.
  - Delete the log file.  
 PowerBuilder will automatically create a new log file the next time you connect to a database with tracing enabled.

## Sample Database Trace output

This section gives an example of Database Trace output that you might see in the log file and briefly explains each portion of the output.

The example traces a connection to an ODBC database named Sample. The output was generated while running a PowerBuilder application that displays information about employees in each department. The **SELECT** statement shown retrieves information from the Employee table to display the names of employees in department 100.

The precision (for example, milliseconds) used when Database Trace records internal commands depends on your operating system's timing mechanism. Therefore, the timing precision in your Database Trace log may vary from this example.

Connect to database	DIALOG CONNECT TO TRACE SYCAptive Server Enterprise: USERID= SERVER=HOST12 DATA=QATEST DPPARM=Release='12'
Prepare SELECT statement	PREPARE: SELECT employee.emp_id, employee.emp_lname, employee.emp_fname, employee.dept_id FROM employee WHERE ( employee.dept_id = 100 ) ORDER BY employee.emp_lname ASC (94 MilliSeconds)
Get column descriptions	DESCRIBE: (0 MilliSeconds) name=emp_id,len=4,type=????,pbt5,dbt3,ct0,dec0 name=emp_lname,len=21,type=CHAR,pbt1,dbt1,ct0,dec0 name=emp_fname,len=21,type=CHAR,pbt1,dbt1,ct0,dec0 name=dept_id,len=4,type=????,pbt5,dbt3,ct0,dec0

Bind memory buffers  
to columns

```
BIND SELECT OUTPUT BUFFER (DataWindow):
(890 MilliSeconds)
name=emp_id,len=4,type=FLOAT,pbt3,dbt3,ct0,dec0
name=emp_lname,len=21,type=CHAR,pbt1,dbt1,ct0,dec0
name=emp_fname,len=21,type=CHAR,pbt1,dbt1,ct0,dec0
name=dept_id,len=4,type=FLOAT,pbt3,dbt3,ct0,dec0
```

Execute SELECT  
statement

```
EXECUTE: (0 MilliSeconds)
```

Fetch rows from result  
set

```
FETCH NEXT: (156 MilliSeconds)
    emp_id=    emp_lname=Jones    emp_fname=Alan
    dept_id=
FETCH NEXT: (0 MilliSeconds)
    emp_id=    emp_lname=Cicccone    emp_fname=Peter
    dept_id=
FETCH NEXT: (0 MilliSeconds)
    emp_id=    emp_lname=Houston    emp_fname=Mary
    dept_id=
FETCH NEXT: (0 MilliSeconds)
    emp_id=    emp_lname=Smith        emp_fname=Susan
    dept_id=
FETCH NEXT: (0 MilliSeconds)
    emp_id=    emp_lname=Stein        emp_fname=David
    dept_id=
FETCH NEXT: (0 MilliSeconds)
    emp_id=    emp_lname=Watson    emp_fname=Linda
    dept_id=
FETCH NEXT: (0 MilliSeconds)
Error 1 (rc 100)
```

Commit database  
changes

```
COMMIT: (55 MilliSeconds )
```

Disconnect from  
database

```
DISCONNECT: (0 MilliSeconds)
```

Shut down database  
interface

```
SHUTDOWN DATABASE INTERFACE: (203 MilliSeconds)
```

## Using the ODBC Driver Manager Trace

This section describes how to use the ODBC Driver Manager Trace tool.

### About ODBC Driver Manager Trace

You can use the ODBC Driver Manager Trace tool to trace a connection to any ODBC data source that you access in PowerBuilder through the ODBC interface.

Unlike the Database Trace tool, the ODBC Driver Manager Trace tool *cannot* trace connections through one of the native database interfaces.

What this tool does	ODBC Driver Manager Trace records information about ODBC API calls (such as <code>SQLDriverConnect</code> , <code>SQLGetInfo</code> , and <code>SQLFetch</code> ) made by PowerBuilder while connected to an ODBC data source. It writes this information to a default log file named <i>SQL.LOG</i> or to a log file that you specify.
What both tools do	<p>The information from ODBC Driver Manager Trace, like Database Trace, can help you:</p> <ul style="list-style-type: none"><li>• Understand what PowerBuilder is doing <i>internally</i> while connected to an ODBC data source</li><li>• Identify and resolve problems with your ODBC connection</li><li>• Provide useful information to Technical Support if you call them for help with your database connection</li></ul>
When to use this tool	Use ODBC Driver Manager Trace <i>instead</i> of the Database Trace tool if you want more detailed information about the ODBC API calls made by PowerBuilder.

---

#### Performance considerations

Turning on ODBC Driver Manager Trace can slow your performance while working in PowerBuilder. Therefore, use ODBC Driver Manager Trace for debugging purposes only and keep it turned off when you are not debugging.

---

SQL.LOG file	PowerBuilder writes ODBC Driver Manager Trace output to a default log file named <i>SQL.LOG</i> or to a log file that you specify. The default location of <i>SQL.LOG</i> is in your root directory.
--------------	--

## Starting ODBC Driver Manager Trace

By default, ODBC Driver Manager Trace is turned off in PowerBuilder. You can start it in order to trace your ODBC connection in two ways:

- Edit your database profile in the PowerBuilder development environment
- Edit a script in a PowerBuilder application

### Starting ODBC Driver Manager Trace in the development environment

To start ODBC Driver Manager Trace in the PowerBuilder development environment, edit the database profile for the connection you want to trace, as described in the following procedure.

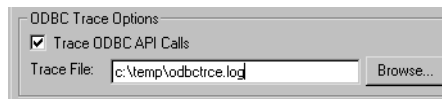
❖ **To start ODBC Driver Manager Trace by editing the database profile:**

- 1 Open the Database Profile Setup-ODBC dialog box for the ODBC connection you want to trace.
- 2 On the Options tab, select the Trace ODBC API Calls check box.
- 3 (Optional) To specify a log file where you want PowerBuilder to write the output of ODBC Driver Manager Trace, type the pathname in the Trace File box.

*or*

(Optional) Click Browse to display the pathname of an existing log file in the Trace File box.

By default, if the Trace ODBC API Calls check box is selected and no trace file is specified, PowerBuilder sends ODBC Driver Manager Trace output to the default *SQL.LOG* file.



- 4 Click OK or Apply.

*or*

Right-click on the connected database and select Re-connect from the dropdown menu in the Database Profiles dialog box.

The Database Profiles dialog box displays with the name of the edited profile highlighted.

PowerBuilder saves your settings in the database profile entry in the registry in the *HKEY\_CURRENT\_USER\Software\Sybase\PowerBuilder\9.0\DatabaseProfiles* key.

For example, here is the relevant portion of a database profile entry for an ODBC data source named Employee. The settings that start ODBC Driver Manager Trace (corresponding to the ConnectOption DBParm parameter) are emphasized.

```
DBMS      "ODBC "  
...  
DbParm    "ConnectionString='DSN=Employee;UID=dba;  
PWD=00c61737',ConnectOption='SQL_OPT_TRACE,SQL_OPT_  
TRACE_ON;SQL_OPT_TRACEFILE,C:\Temp\odbctrce.log'"
```

- 5 Click Connect in the Database Profiles dialog box to connect to the database.

*or*

Right-click on the connected database and select Re-connect from the dropdown menu in the Database Profiles dialog box.

PowerBuilder connects to the database, starts tracing the ODBC connection, and writes output to the log file you specified.

## Starting ODBC Driver Manager Trace in a PowerBuilder application

To start ODBC Driver Manager Trace in a PowerBuilder application, you must specify certain values for the ConnectOption DBParm parameter in the appropriate script. For example, you might include them in the script that opens the application.

You can specify the required ConnectOption values in a PowerBuilder script by:

- *(Recommended)* Copying the PowerScript ConnectOption DBParm syntax from the Preview tab in the Database Profile Setup dialog box into your script
- Coding PowerScript to set a value for the DBParm property of the Transaction object
- Reading the DBParm values from an external text file

For more about using Transaction objects to communicate with a database in a PowerBuilder application, see *Application Techniques*.



### About the ConnectOption DBParm parameter

ConnectOption includes several parameters, two of which control the operation of ODBC Driver Manager Trace for any ODBC-compatible driver you are using in PowerBuilder:

**Table 8-2: ConnectOption parameters for ODBC Driver Manager Trace**

Parameter	Description
SQL_OPT_TRACE	<p><b>Purpose</b> Starts or stops ODBC Driver Manager Trace in PowerBuilder.</p> <p><b>Values</b> The values you can specify are:</p> <ul style="list-style-type: none"> <li>• <b>SQL_OPT_TRACE_OFF</b> (Default) Stops ODBC Driver Manager Trace</li> <li>• <b>SQL_OPT_TRACE_ON</b> Starts ODBC Driver Manager Trace</li> </ul>
SQL_OPT_TRACEFILE	<p><b>Purpose</b> Specifies the name of the trace file where you want to send the output of ODBC Driver Manager Trace. PowerBuilder appends the output to the trace file you specify until you stop the trace. To display the trace file, you can use the File Editor (in PowerBuilder) or any text editor (outside PowerBuilder).</p> <p><b>Values</b> You can specify any filename for the trace file, <i>following the naming conventions of your operating system</i>. By default, if tracing is on and you have not specified a trace file, PowerBuilder sends ODBC Driver Manager Trace output to a file named <i>SQL.LOG</i>.</p> <p>For information about the location of <i>SQL.LOG</i> on different platforms, see “About ODBC Driver Manager Trace” on page 172.</p>

### Copying ConnectOption syntax from the Preview tab

The easiest way to start ODBC Driver Manager Trace in a PowerBuilder application script is to copy the PowerScript ConnectString DBParm syntax from the Preview tab in the Database Profile Setup - ODBC dialog box into your script, modifying the default Transaction object name (SQLCA) if necessary.

As you complete the Database Profile Setup dialog box in the development environment, PowerBuilder generates the correct connection syntax on the Preview tab. Therefore, copying the syntax directly from the Preview tab into your script ensures that it is accurate.

❖ **To copy ConnectOption syntax from the Preview tab into your script:**

- 1 On the Options tab in the Database Profile Setup - ODBC dialog box for your connection, select the Trace ODBC API Calls check box and (optionally) specify a log file in the Trace File box to start ODBC Driver Manager Trace.
- 2 Click Apply to save your changes to the Options tab without closing the dialog box.
- 3 Click the Preview tab.

The correct PowerScript syntax for ODBC Driver Manager Trace and other selected options displays in the Database Connection Syntax box.

The following example shows the PowerScript syntax that starts ODBC Driver Manager Trace and sends output to the file *C:\TEMP\ODBCTRCE.LOG*.

```
// Profile Employee
SQLCA.DBMS = "ODBC"
SQLCA.AutoCommit = False
SQLCA.DBParm = "Connectstring='DSN=Employee',
               ConnectOption='SQL_OPT_TRACE,SQL_OPT_TRACE_ON;
               SQL_OPT_TRACEFILE,c:\temp\odbctrce.log' "
```

- 4 Select the SQLCA.DBParm line and any other syntax you want to copy to your script and click Copy.

PowerBuilder copies the selected text to the clipboard.

- 5 Paste the selected text from the Preview tab into your script, modifying the default Transaction object name (SQLCA) if necessary.

### **Coding PowerScript to set a value for the DBParm property**

Another way to start ODBC Driver Manager Trace in a PowerBuilder application script is to include the ConnectOption parameters that control tracing as values for the DBParm property of the Transaction object.

❖ **To start ODBC Driver Manager Trace by setting the DBParm property:**

- In your application script, set the `SQL_OPT_TRACE` and (optionally) `SQL_OPT_TRACEFILE` ConnectOption parameters to start the trace and to specify a nondefault trace file, respectively.

For example, the portion of the following statement shown below starts ODBC Driver Manager Trace in your application and sends output to a file named *MYTRACE.LOG*. Insert a comma to separate the `ConnectionString` and `ConnectOption` values. (This example assumes you are using the default Transaction object `SQLCA`, but you can also define your own Transaction object.)

```
SQLCA.DBParm="ConnectionString='DSN=Test;UID=PB;
PWD=xyz',ConnectOption='SQL_OPT_TRACE,
SQL_OPT_TRACE_ON;SQL_OPT_TRACEFILE,C:\TRC.LOG'"
```

**Reading the DBParm value from an external text file**

As an alternative to setting the `DBParm` property in your PowerBuilder application script, you can use the `ProfileString` function to read `DBParm` values from a specified section of an external text file, such as an application-specific initialization file.

This assumes that the `DBParm` value read from your initialization file includes the `ConnectOption` parameter to start ODBC Driver Manager Trace, as shown in the preceding example.

❖ **To start ODBC Driver Manager Trace by reading DBParm values from an external text file:**

- Use the following PowerScript syntax to specify the `ProfileString` function with the `DBParm` property:

```
SQLCA.dbParm = ProfileString(file, section, variable,  
default_value)
```

For example, the following statement in a PowerBuilder script reads the `DBParm` values from the [Database] section of the *APP.INI* file:

```
SQLCA.dbParm =  
ProfileString("APP.INI", "Database", "DBParm", "")
```

## Stopping ODBC Driver Manager Trace

Once you start tracing an ODBC connection with ODBC Driver Manager Trace, PowerBuilder continues sending trace output to the log file until you stop tracing. After you stop tracing as described in the following sections, you must reconnect to have the changes take effect.

### Stopping ODBC Driver Manager Trace in the development environment

❖ **To stop ODBC Driver Manager Trace by editing a database profile:**

- 1 Open the Database Profile Setup - ODBC dialog box for the connection you are tracing.

For instructions, see “Starting ODBC Driver Manager Trace in the development environment” on page 173.

- 2 On the Options tab, clear the Trace ODBC API Calls check box.

If you supplied the pathname of a log file in the Trace File box, you can leave it specified in case you want to restart tracing later.

- 3 Click OK in the Database Profile Setup - ODBC dialog box.

The Database Profiles dialog box displays, with the name of the edited profile highlighted.

- 4 Click Connect in the Database Profiles dialog box.

*or*

Right-click on the connected database and select Re-connect from the dropdown menu in the Database Profiles dialog box.

PowerBuilder connects to the database and stops tracing the connection.

### Stopping ODBC Driver Manager Trace in a PowerBuilder application

To stop ODBC Driver Manager Trace in a PowerBuilder application script, you must change the SQL\_OPT\_TRACE ConnectOption parameter to SQL\_OPT\_TRACE\_OFF. You can do this by:

- Editing the value of the DBParm property of the Transaction object
- Reading the DBParm values from an external text file

### Editing the DBParm property

One way to change the ConnectOption value in a PowerBuilder script is to edit the DBParm property of the Transaction object.

❖ **To stop ODBC Driver Manager Trace by editing the DBParm property:**

- In your application script, edit the DBParm property of the Transaction object to change the value of the SQL\_OPT\_TRACE ConnectOption parameter to SQL\_OPT\_TRACE\_OFF.

For example, the following statement starts ODBC Driver Manager Trace in your application and sends the output to a file named *MYTRACE.LOG*. (This example assumes you are using the default Transaction object SQLCA, but you can also define your own Transaction object.)

```
SQLCA.DBParm="ConnectionString='DSN=Test;UID=PB;  
PWD=xyz',ConnectOption='SQL_OPT_TRACE,  
SQL_OPT_TRACE_ON;SQL_OPT_TRACEFILE,C:\TRC.LOG' "
```

Here is how the same statement should look after you edit it to stop ODBC Driver Manager Trace. (You can leave the name of the trace file specified in case you want to restart tracing later.)

```
SQLCA.DBParm="ConnectionString='DSN=Test;UID=PB;  
PWD=xyz',ConnectOption='SQL_OPT_TRACE,  
SQL_OPT_TRACE_OFF;SQL_OPT_TRACEFILE,C:\TRC.LOG' "
```

### Reading DBParm values

As an alternative to editing the DBParm property in your PowerBuilder application script, you can use the PowerScript ProfileString function to read DBParm values from a specified section of an external text file, such as an application-specific initialization file.

This assumes that the DBParm value read from your initialization file sets the value of SQL\_OPT\_TRACE to SQL\_OPT\_TRACE\_OFF, as shown in the preceding example.

## Viewing the ODBC Driver Manager Trace log

You can display the contents of the ODBC Driver Manager Trace log file anytime during a PowerBuilder session.

---

### Location of SQL.LOG

For information about where to find the default SQL.LOG file, see “About ODBC Driver Manager Trace” on page 172.

---

❖ **To view the contents of the log file:**

- Open SQL.LOG or the log file you specified in one of the following ways:
  - Use the File Editor in PowerBuilder. (For instructions, see the *User's Guide*.)
  - Use any text editor outside PowerBuilder.

---

### Leaving the log file open

If you leave the log file open as you work in PowerBuilder, ODBC Driver Manager Trace *does not update it*.

---

## Sample ODBC Driver Manager Trace output

This section shows a partial example of output from ODBC Driver Manager Trace to give you an idea of the information it provides. The example is part of the trace on an ODBC connection to the EAS Demo DB.

For more about a particular ODBC API call, see your ODBC documentation.

```
PB90 179:192  EXIT  SQLSetConnectOption  with return
code 0 (SQL_SUCCESS)
      HDBC 0x036e1300
      UWORD 104 <SQL_OPT_TRACE>
      UDWORD 1
```

```
PB90 179:192  ENTER SQLSetConnectOption
      HDBC 0x036e1300
      UWORD 105 <SQL_OPT_TRACEFILE>
      UDWORD 160694373
PB90 179:192  EXIT  SQLSetConnectOption  with return
code 0 (SQL_SUCCESS)
      HDBC 0x036e1300
      UWORD 105 <SQL_OPT_TRACEFILE>
```

```
UDWORD 160694373
PB90 179:192  ENTER SQLDriverConnectW
HDBC 0x036e1300
HWND 0x004607fa
WCHAR * 0x1f4be068 [      -3] "*****\ 0"
SWORD -3
WCHAR * 0x1f4be068
SWORD 8
SWORD * 0x00000000
UWORD 1 <SQL_DRIVER_COMPLETE>
PB90 179:192  EXIT SQLDriverConnectW with return code
0 (SQL_SUCCESS)
HDBC 0x036e1300
HWND 0x004607fa
WCHAR * 0x1f4be068 [      -3] "*****\ 0"
SWORD -3
WCHAR * 0x1f4be068
SWORD 8
SWORD * 0x00000000
UWORD 1 <SQL_DRIVER_COMPLETE>
PB90 179:192  ENTER SQLGetInfoW
HDBC 0x036e1300
UWORD 6 <SQL_DRIVER_NAME>
PTR 0x036e2098
SWORD 6
SWORD * 0x0012cd30
PB90 179:192  EXIT SQLGetInfoW with return code 1
(SQL_SUCCESS_WITH_INFO)
HDBC 0x036e1300
UWORD 6 <SQL_DRIVER_NAME>
PTR 0x036e2098 [      6] "DB\ 0"
SWORD 6
SWORD * 0x0012cd30 (22)
DIAG [01004] [Sybase][ODBC Driver]Data truncated (0)
PB90 179:192  ENTER SQLGetInfoW
HDBC 0x036e1300
UWORD 10 <SQL_ODBC_VER>
PTR 0x036e39f8
SWORD 100
SWORD * 0x0012cd38
PB90 179:192  EXIT SQLGetInfoW with return code 0
(SQL_SUCCESS)
HDBC 0x036e1300
UWORD 10 <SQL_ODBC_VER>
PTR 0x036e39f8 [     20] "03.51.0000"
SWORD 100
```

```
        SWORD * 0x0012cd38 (20)
PB90 179:192  ENTER SQLGetInfoW
        HDBC 0x036e1300
        UWORD 2 <SQL_DATA_SOURCE_NAME>
        PTR 0x036e3c88
        SWORD 512
        SWORD * 0x0012cc32
PB90 179:192  EXIT  SQLGetInfoW  with return code 0
(SQL_SUCCESS)
        HDBC 0x036e1300
        UWORD 2 <SQL_DATA_SOURCE_NAME>
        PTR 0x036e3c88 [      28] "EAS Demo DB V4"
        SWORD 512
        SWORD * 0x0012cc32 (28)
PB90 179:192  ENTER SQLGetInfoW
        HDBC 0x036e1300
        UWORD 16 <SQL_DATABASE_NAME>
        PTR 0x036e3c88
        SWORD 512
        SWORD * 0x0012cc32
PB90 179:192  EXIT  SQLGetInfoW  with return code 0
(SQL_SUCCESS)
        HDBC 0x036e1300
        UWORD 16 <SQL_DATABASE_NAME>
        PTR 0x036e3c88 [      16] "easdemo4"
        SWORD 512
        SWORD * 0x0012cc32 (16)
PB90 179:192  ENTER SQLGetInfoW
        HDBC 0x036e1300
        UWORD 25 <SQL_DATA_SOURCE_READ_ONLY>
        PTR 0x036e3c88
        SWORD 512
        SWORD * 0x0012cc32
PB90 179:192  EXIT  SQLGetInfoW  with return code 0
(SQL_SUCCESS)
        HDBC 0x036e1300
        UWORD 25 <SQL_DATA_SOURCE_READ_ONLY>
        PTR 0x036e3c88 [        2] "N"
        SWORD 512
        SWORD * 0x0012cc32 (2)
PB90 179:192  ENTER SQLGetInfoW
        HDBC 0x036e1300
        UWORD 13 <SQL_SERVER_NAME>
        PTR 0x036e3c88
        SWORD 512
        SWORD * 0x0012cc32
```



```
PB90 179:192  EXIT  SQLGetInfoW  with return code 0
(SQL_SUCCESS)
    HDBC 0x036e1300
    UWORD 13 <SQL_SERVER_NAME>
    PTR 0x036e3c88 [      16] "easdemo4"
    SWORD 512
    SWORD * 0x0012cc32 (16)
PB90 179:192  ENTER SQLGetInfoW
    HDBC 0x036e1300
    UWORD 17 <SQL_DBMS_NAME>
    PTR 0x036e3c88
    SWORD 512
    SWORD * 0x0012cab6
PB90 179:192  EXIT  SQLGetInfoW  with return code 0
(SQL_SUCCESS)
    HDBC 0x036e1300
    UWORD 17 <SQL_DBMS_NAME>
    PTR 0x036e3c88 [      48] "Adaptive Server
Anywhere"
    SWORD 512
    SWORD * 0x0012cab6 (48)
PB90 179:192  ENTER SQLGetInfoW
    HDBC 0x036e1300
    UWORD 6 <SQL_DRIVER_NAME>
    PTR 0x036e1a10
    SWORD 550
    SWORD * 0x0012cbbc
PB90 179:192  EXIT  SQLGetInfoW  with return code 0
(SQL_SUCCESS)
    HDBC 0x036e1300
    UWORD 6 <SQL_DRIVER_NAME>
    PTR 0x036e1a10 [      22] "DBODBC7.DLL"
    SWORD 550
    SWORD * 0x0012cbbc (22)
PB90 179:192  ENTER SQLAllocStmt
    HDBC 0x036e1300
    HSTMT * 0x0012d0b4
PB90 179:192  EXIT  SQLAllocStmt  with return code 0
(SQL_SUCCESS)
    HDBC 0x036e1300
    HSTMT * 0x0012d0b4 ( 0x036e1c48)
PB90 179:192  ENTER SQLGetTypeInfo
    HSTMT 0x036e1c48
    SWORD 0 <SQL_ALL_TYPES>
```

## Using the JDBC Driver Manager Trace

This section describes how to use the JDBC Driver Manager Trace tool.

### About JDBC Driver Manager Trace

You can use the JDBC Driver Manager Trace tool to trace a connection to any database that you access in PowerBuilder through the JDBC interface.

Unlike the Database Trace tool, the JDBC Driver Manager Trace tool *cannot* trace connections through one of the native database interfaces.

What this tool does      JDBC Driver Manager Trace logs errors and informational messages originating from the Driver object currently loaded (such as Sybase's jConnect JDBC driver) when PowerBuilder connects to a database through the JDBC interface. It writes this information to a default log file named *JDBC.LOG* or to a log file that you specify. The amount of trace output varies depending on the JDBC driver being used.

What both tools do      The information from JDBC Driver Manager Trace, like Database Trace, can help you:

- Understand what PowerBuilder is doing *internally* while connected to a database through the JDBC interface
- Identify and resolve problems with your JDBC connection
- Provide useful information to Technical Support if you call them for help with your database connection

When to use this tool      Use JDBC Driver Manager Trace *instead* of the Database Trace tool if you want more detailed information about the JDBC driver.

---

#### Performance considerations

Turning on JDBC Driver Manager Trace can slow your performance while working in PowerBuilder. Therefore, use JDBC Driver Manager Trace for debugging purposes only and keep it turned off when you are not debugging.

---

JDBC.LOG file      PowerBuilder writes JDBC Driver Manager Trace output to a default log file named *JDBC.LOG* or to a log file that you specify. The default location of *JDBC.LOG* is a temp directory.

## Starting JDBC Driver Manager Trace

By default, JDBC Driver Manager Trace is turned off in PowerBuilder. You can start it in order to trace your JDBC connection in two ways:

- Edit your database profile in the PowerBuilder development environment
- Edit a script in a PowerBuilder application

### Starting JDBC Driver Manager Trace in the development environment

To start JDBC Driver Manager Trace in the PowerBuilder development environment, edit the database profile for the connection you want to trace, as described in the following procedure.

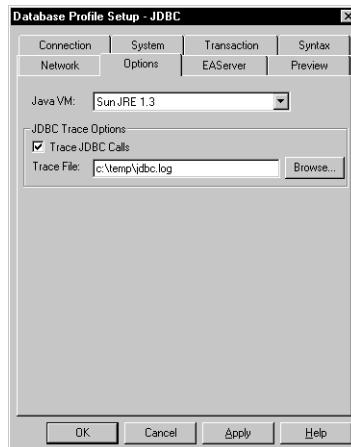
❖ **To start JDBC Driver Manager Trace by editing the database profile:**

- 1 Open the Database Profile Setup - JDBC dialog box for the JDBC connection you want to trace.
- 2 On the Options tab, select the Trace JDBC Calls check box.
- 3 (Optional) To specify a log file where you want PowerBuilder to write the output of JDBC Driver Manager Trace, type the pathname in the Trace File box.

*or*

(Optional) Click Browse to display the pathname of an existing log file in the Trace File box.

By default, if the Trace JDBC Calls check box is selected and no alternative trace file is specified, PowerBuilder sends JDBC Driver Manager Trace output to the default *JDBC.LOG* file.



- 4 Click OK or Apply.

The Database Profiles dialog box displays with the name of the edited profile highlighted.

PowerBuilder saves your settings in the database profile entry in the registry.

For example, here are the DBMS and DBParm string values of a database profile entry for a database named Employee. The settings that start JDBC Driver Manager Trace (corresponding to the TraceFile DBParm parameter) are emphasized.

```
DBMS      "TRACE JDBC"
DbParm    "Driver='com.sybase.jdbc.SybDriver',
          URL='jdbc:sybase:Tds:199.93.178.151:
          5007/tsdata',JavaVM='Sun1.3',TraceFile=
          'c:\temp\jdbc.log'"
```

- 5 Click Connect in the Database Profiles dialog box to connect to the database.

*or*

Right-click on the connected database and select Re-connect from the dropdown menu in the Database Profiles dialog box.

PowerBuilder connects to the database, starts tracing the JDBC connection, and writes output to the log file you specified.

## Starting JDBC Driver Manager Trace in a PowerBuilder application

To start JDBC Driver Manager Trace in a PowerBuilder application, you must specify the TraceFile DBParm parameter in the appropriate script. For example, you might include it in the script that opens the application.

You can specify the TraceFile parameter in a PowerBuilder script by:

- (Recommended) Copying the PowerScript TraceFile DBParm syntax from the Preview tab in the Database Profile Setup dialog box into your script
- Coding PowerScript to set a value for the DBParm property of the Transaction object
- Reading the DBParm values from an external text file

For more about using Transaction objects to communicate with a database in a PowerBuilder application, see *Application Techniques*.

### About the TraceFile DBParm parameter

TraceFile controls the operation of JDBC Driver Manager Trace for any JDBC-compatible driver you are using in PowerBuilder:

### Copying TraceFile syntax from the Preview tab

The easiest way to start JDBC Driver Manager Trace in a PowerBuilder application script is to copy the PowerScript TraceFile DBParm syntax from the Preview tab in the Database Profile Setup - JDBC dialog box into your script, modifying the default Transaction object name (SQLCA) if necessary.

As you complete the Database Profile Setup dialog box in the development environment, PowerBuilder generates the correct connection syntax on the Preview tab. Therefore, copying the syntax directly from the Preview tab into your script ensures that it is accurate.

#### ❖ To copy TraceFile syntax from the Preview tab into your script:

- 1 On the Options tab in the Database Profile Setup - JDBC dialog box for your connection, select the Trace JDBC Calls check box and (optionally) specify a log file in the Trace File box to start JDBC Driver Manager Trace.

For instructions, see “Starting JDBC Driver Manager Trace in the development environment” on page 185.

- 2 Click Apply to save your changes to the Options tab without closing the dialog box.
- 3 Click the Preview tab.

The correct PowerScript syntax for JDBC Driver Manager Trace and other selected options displays in the Database Connection Syntax box.

The following example shows the PowerScript syntax that starts JDBC Driver Manager Trace and sends output to the file *C:\TEMP\JDBC.LOG*.

```
// Profile Employee
SQLCA.DBMS = "TRACE JDBC"
SQLCA.DBParm = "Driver='com.sybase.jdbc.SybDriver',
URL='jdbc:sybase:Tds:199.93.178.151:5007/tsdata',
JavaVM='Sun1.3',TraceFile='c:\temp\jdbc.log' "
```

- 4 Select the DBParm line and any other syntax you want to copy to your script and click Copy.  
  
PowerBuilder copies the selected text to the clipboard.
- 5 Paste the selected text from the Preview tab into your script, modifying the default Transaction object name (SQLCA) if necessary.

### Coding PowerScript to set a value for the DBParm property

Another way to start JDBC Driver Manager Trace in a PowerBuilder application script is to include the TraceFile parameter as a value for the DBParm property of the Transaction object.

❖ **To start JDBC Driver Manager Trace by setting the DBParm property:**

- In your application script, include the TraceFile parameter to start the trace and specify a nondefault trace file.

For example, this statement starts JDBC Driver Manager Trace in your application and sends output to a file named *MYTRACE.LOG*. (This example assumes you are using the default Transaction object SQLCA, but you can also define your own Transaction object.)

```
SQLCA.DBParm = "Driver='com.sybase.jdbc.SybDriver',  
URL='jdbc:sybase:Tds:199.93.178.151:5007/tsdata',  
JavaVM='Sun1.3',TraceFile='c:\MYTRACE.LOG'"
```

### Reading the DBParm value from an external text file

As an alternative to setting the DBParm property in your PowerBuilder application script, you can use the PowerScript ProfileString function to read DBParm values from a specified section of an external text file, such as an application-specific initialization file.

This assumes that the DBParm value read from your initialization file includes the TraceFile parameter to start JDBC Driver Manager Trace, as shown in the preceding example.

❖ **To start JDBC Driver Manager Trace by reading DBParm values from an external text file:**

- Use the following PowerScript syntax to specify the ProfileString function with the DBParm property:

```
SQLCA.dbParm = ProfileString(file, section, variable,  
default_value)
```

For example, the following statement in a PowerBuilder script reads the DBParm values from the [Database] section of the *APP.INI* file:

```
SQLCA.dbParm =  
    ProfileString( "APP.INI", "Database", "DBParm", " " )
```

## Stopping JDBC Driver Manager Trace

Once you start tracing a JDBC connection with JDBC Driver Manager Trace, PowerBuilder continues sending trace output to the log file until you stop tracing.

### Stopping JDBC Driver Manager Trace in the development environment

❖ **To stop JDBC Driver Manager Trace by editing a database profile:**

- 1 Open the Database Profile Setup - JDBC dialog box for the connection you are tracing.

For instructions, see “Starting JDBC Driver Manager Trace in the development environment” on page 185.

- 2 On the Options tab, clear the Trace JDBC Calls check box.

If you supplied the pathname of a log file in the Trace File box, you can leave it specified in case you want to restart tracing later.

- 3 Click OK in the Database Profile Setup - JDBC dialog box.

The Database Profiles dialog box displays, with the name of the edited profile highlighted.

- 4 Click Connect in the Database Profiles dialog box.

*or*

Right click on the connected database and select Re-connect from the dropdown menu in the Database Profiles dialog box.

PowerBuilder connects to the database and stops tracing the connection.

## Stopping JDBC Driver Manager Trace in a PowerBuilder application

To stop JDBC Driver Manager Trace in a PowerBuilder application script, you must delete the TraceFile parameter. You can do this by:

- Editing the value of the DBParm property of the Transaction object
- Reading the DBParm values from an external text file

### Editing the DBParm property

One way to change the TraceFile parameter in a PowerBuilder script is to edit the DBParm property of the Transaction object.

#### ❖ To stop JDBC Driver Manager Trace by editing the DBParm property:

- In your application script, edit the DBParm property of the Transaction object to delete the TraceFile parameter.

For example, the following statement starts JDBC Driver Manager Trace in your application and sends the output to a file named *MYTRACE.LOG*. (This example assumes you are using the default Transaction object SQLCA, but you can also define your own Transaction object.)

```
SQLCA.DBParm = "Driver='com.sybase.jdbc.SybDriver',  
URL='jdbc:sybase:Tds:199.93.178.151:5007/tsdata',  
JavaVM='Sun1.3',TraceFile='c:\MYTRACE.LOG' "
```

Here is how the same statement should look after you edit it to stop JDBC Driver Manager Trace.

```
SQLCA.DBParm = "Driver='com.sybase.jdbc.SybDriver',  
URL='jdbc:sybase:Tds:199.93.178.151:5007/tsdata',  
JavaVM='Sun1.3' "
```

### Reading DBParm values

As an alternative to editing the DBParm property in your PowerBuilder application script, you can use the PowerScript ProfileString function to read DBParm values from a specified section of an external text file, such as an application-specific initialization file, or you can use RegistryGet to obtain values from a registry key.

This assumes that the DBParm is no longer read from your initialization file or registry key, as shown in the preceding example. You must disconnect and reconnect for this to take effect.



## Viewing the JDBC Driver Manager Trace log

You can display the contents of the JDBC Driver Manager Trace log file anytime during a PowerBuilder session.

---

### Location of JDBC.LOG

For information about where to find the default *JDBC.LOG* file, see “About JDBC Driver Manager Trace” on page 184.

---

❖ **To view the contents of the log file:**

- Open *JDBC.LOG* or the log file you specified in one of the following ways:
  - Use the File Editor in PowerBuilder. (For instructions, see the *User's Guide*.)
  - Use any text editor outside PowerBuilder.

---

### Leaving the log file open

If you leave the log file open as you work in PowerBuilder, JDBC Driver Manager Trace *does not update the log*.

---



## PART 5

# Working with Transaction Servers

This part describes how to make database connections for transactional components.



# Making Database Connections in PowerBuilder Components

This chapter describes the database connections you can make if you are developing a PowerBuilder component that will be deployed to a transaction server. It also describes how to create a profile to simplify connections to EAServer.

Topic	Page
Deploying a component to EAServer	195
Deploying a COM component to MTS	199
DBParm support for PowerBuilder components	200

## Deploying a component to EAServer

If you are developing a PowerBuilder custom class user object containing business logic that will be deployed to a transaction server, there are some database connectivity issues to keep in mind.

For detailed information about the files you need to deploy with applications or components you build in PowerBuilder, see the chapter on deploying your application in *Application Techniques*.

If you want the component you are developing to take advantage of EAServer's support for connection pooling and transaction management, you *must* use one of the database interfaces supported by the transaction coordinator being used by EAServer. EAServer supports three different transaction coordinators:

- Shared Connection
- Microsoft DTC
- OTS/XA Transactions

The default coordinator is the Shared Connection coordinator.

---

**Setting the transaction coordinator**

The transaction coordinator is set through Jaguar Manager using the Transaction tab of the Server Properties dialog box.

---

## Supported database connections when using Shared Connection

The pseudo-coordinator shared connection is built into EAServer. In this model, all components participating in a transaction share a single connection. To use this model, all of your application data must reside on one data server, and all components that participate in a transaction must use a connection with the same user name, password and server name or the same EAServer connection cache name as defined in the CacheName DBParm. It supports the following database interfaces to connect to the database:

- ODBC database interface, which provides connectivity to a variety of databases through ODBC drivers. The same ODBC drivers shipped with PowerBuilder are also supported on EAServer.
- Sybase SYJ database interface, which provides connectivity to Adaptive Server Enterprise 11.5 or later. (Some versions of Open Client and Adaptive Server currently do not support OTS/XA transactions on Windows NT.)
- JDB database interface, which provides connectivity through either Sun's or Microsoft's Java Virtual Machine to a JDBC driver such as Sybase jConnect. The JDM and JDS interfaces are not supported in PowerBuilder 9.0.
- Oracle O73, O84, and O90 database interfaces, which provide connectivity to Oracle 7.3, Oracle 8.0.x and Oracle8i, and Oracle9i databases.

## Supported database connections when using Microsoft DTC

Microsoft Distributed Transaction Coordinator (DTC) uses two-phase commit to coordinate transactions among multiple databases. This transaction coordinator supports the following database interfaces to connect to the database:

- ODBC database interface. Support is limited to the following ODBC drivers: Microsoft SQL Server 6.5 or later and Microsoft ODBC driver for Oracle.
- JDB database interface, which provides connectivity through either Sun's or Microsoft's Java Virtual Machine to a JDBC driver which acts as a JDBC-ODBC bridge. The JDM and JDS interfaces are not supported in PowerBuilder 9.0.

## Supported database connections when using OTS/XA

This option uses the Transarc Encina transaction coordinator that is built into EAServer. The Encina transaction coordinator uses two-phase commit to coordinate transactions among multiple databases. This transaction coordinator supports the following database interfaces to connect to the database:

- The SYJ database interface, which provides connectivity to Adaptive Server Enterprise 11.5 or later. (Some versions of Adaptive Server do not support OTS/XA transactions on Windows NT.)
- The JDB database interface, which provides connectivity through Sun's Java Virtual Machine (JRE 1.2 or later) to a JDBC driver that supports the Java Transaction API (JTA) such as Sybase jConnect 5.2.
- The O73, O84, and O90 database interfaces, which provide connectivity to Oracle 7.3, Oracle 8.0.x and Oracle8i, and Oracle 9i databases.

## Using the SYJ database interface

EAServer uses a slightly different version of the Sybase Open Client CT-Library (CT-Lib) software from PowerBuilder. Therefore, at runtime, you need to use SYJ rather than SYC to connect to an Adaptive Server Enterprise database. The SYJ Database Profile Setup dialog box provides a convenient way to set the appropriate connection parameters and then copy the syntax from the Preview tab into the script for your Transaction object.

You cannot use the SYJ interface, however, to connect to the database in the PowerBuilder development environment. Therefore, during the development phase (before the component has been deployed to Jaguar), you must use SYC to connect to the database.

Note that the SYJ database interface supports only those DBParms relevant at runtime. It does not support any DBParm parameters which have to be set before PowerBuilder establishes a database connection. The following DBParms, which are included on the SYJ Profile Setup dialog box, are not supported by SYJ:

- All the DBParms on the Regional Settings tab including CharSet, Language, and Locale
- All the Directory services DBParms on the Directory Services tab
- All the Security services DBParms on the Security tab
- All the DBParms on the Network tab including AppName, Host, MaxConnect, PacketSize, and PWEncrypt
- The Release DBParm on the Connection tab
- The TableCriteria DBParm on the System tab
- The Asynchronous Operations DBParms, Async and DBGetTime, on the Transaction tab

## **Using the JDB database interface**

When you deploy a component developed using the JDB interface to EAServer, PowerBuilder checks the version of the JVM EAServer is using against the version PowerBuilder is using. If the versions do not match, a warning is entered in the Jaguar log file. PowerBuilder uses the version loaded by EAServer. The Jaguar log file records errors relating to component execution. You can view its contents using the Jaguar Manager File Viewer.



## Specifying AutoCommit mode

For those DBMSs and database interfaces that support it (ODBC, SYJ, and JDB), AutoCommit controls whether PowerBuilder issues SQL statements outside or inside the scope of a transaction. When AutoCommit is set to False (the default), PowerBuilder issues SQL statements *inside* the scope of a transaction. When AutoCommit is set to True, PowerBuilder issues SQL statements *outside* the scope of a transaction. AutoCommit is set using the AutoCommit Mode check box on the Connection tab in the Database Profile Setup dialog box or by giving it a value in a PowerBuilder application script.

However, if the component you are developing participates in an EAServer transaction, the AutoCommit setting is ignored. Instead, EAServer determines how the component's database operations execute as part of the transaction.

## Deploying a COM component to MTS

If you want the COM component you are developing to take advantage of MTS support for connection pooling and transaction management, you *must* use one of the following database interfaces to connect to the database:

- ODBC database interface, which provides connectivity to a variety of databases through ODBC drivers.
- Oracle O84 or O90 database interfaces, which provide connectivity to Oracle8i and Oracle9i databases.

## Using the ODBC database interface

If you require support for connection pooling only, you can use any thread-safe ODBC driver. If you also require support for transactions, you must use a driver that supports the Microsoft Distributed Transaction Coordinator (DTC), such as the Microsoft ODBC driver for Oracle or the Microsoft ODBC driver for SQL Server.

## Using the Oracle database interface

A component deployed to MTS can participate in a database transaction only when connecting to an Oracle8i or later database server and when Oracle Services for Microsoft Transaction Server are installed and configured.

## DBParm support for PowerBuilder components

There are several connection options that are relevant only to a PowerBuilder custom class user object that is deployed as a transaction server component. These DBParm parameters can be set through the EAServer or EAServer/MTS tab of the Database Profile Setup dialog box for the appropriate database interface or by giving them a value in a PowerBuilder application script.

<b>DBParm</b>	<b>Relevant when a component is deployed to</b>
UseContextObject	EAServer or MTS
CacheName	EAServer only (Not applicable when using the OTS/XA)
GetConnectionOption	EAServer only
ProxyUserName	EAServer only
ReleaseConnectionOption	EAServer only
OraMTSConFlgs	MTS only

For more information on these DBParms, refer to the online Help.

# Appendix

The Appendix describes how to modify the PBODB90 initialization file.



# Adding Functions to the PBODB90 Initialization File

## About this appendix

In general, *you do not need to modify the PBODB90 initialization file*. In certain situations, however, you may need to add functions to the PBODB90 initialization file for connections to your back-end DBMS through the ODBC interface in PowerBuilder.

This appendix describes how to add functions to the PBODB90 initialization file if necessary.

## Contents

Topic	Page
About the PBODB90 initialization file	203
Adding functions to the PBODB90 initialization file	204

## About the PBODB90 initialization file

### What is the PBODB90 initialization file?

When you access data through the ODBC interface, PowerBuilder uses the PBODB90 initialization file to maintain access to extended functionality in the back-end DBMS for which ODBC does not provide an API call. Examples of extended functionality are SQL syntax or function calls specific to a particular DBMS.

### Editing the PBODB90 initialization file

In most cases, you do *not* need to modify the PBODB90 initialization file. Changes to this file can adversely affect PowerBuilder. Change the PBODB90 initialization file only if you are asked to do so by a Technical Support representative.

However, you *can* edit the PBODB90 initialization file if you need to add functions for your back-end DBMS.

Name of the  
PBODB90  
initialization file

If you modify the PBODB90 initialization file, first make a copy of the existing file. Then keep a record of all changes you make. If you call Technical Support after modifying the PBODB90 initialization file, tell the representative that you changed the file and describe the changes you made.

The name of the PBODB90 initialization file is *PBODB90.INI*.

For simplicity, this appendix refers generically to the PBODB90 initialization file throughout.

## Adding functions to the PBODB90 initialization file

The PBODB90 initialization file lists the functions for certain DBMSs that have ODBC drivers. If you need to add a function to the PBODB90 initialization file for use with your back-end DBMS, you can do either of the following:

- **Existing sections** Add the function to the Functions section for your back-end database if this section exists in the PBODB90 initialization file.
- **New sections** Create new sections for your back-end DBMS in the PBODB90 initialization file and add the function to the newly created Functions section.

## Adding functions to an existing section in the file

If sections for your back-end DBMS *already exist* in the PBODB90 initialization file, use the following procedure to add new functions.

### ❖ To add functions to an existing section in the PBODB90 initialization file:

- 1 Open the PBODB90 initialization file in one of the following ways:
  - Use the File Editor in PowerBuilder. (For instructions, see the *User's Guide*.)
  - Use any text editor outside PowerBuilder.
- 2 Locate the entry for your back-end DBMS in the DBMS Driver/DBMS Settings section of the PBODB90 initialization file.

For example, here is the PBODB90 initialization file entry for Adaptive Server Anywhere:

```
;*****
;DBMS Driver/DBMS Settings see comments at end
;of file
;*****
...
[Adaptive Server Anywhere]
PBSyntax='WATCOM50_SYNTAX'
PBDateTime='STANDARD_DATETIME'
PBFunctions='ASA_FUNCTIONS'
PBDefaultValues='autoincrement,current date,
                current time,current timestamp,timestamp,
                null,user'
PBDefaultCreate='YES'
PBDefaultAlter='YES'
PBDefaultExpressions='YES'
DelimitIdentifier='YES'
PBDateTimeInvalidInSearch='NO'
PBTimeInvalidInSearch='YES'
PBQualifierIsOwner='NO'
PBSpecialDataTypes='WATCOM_SPECIALDATATYPES'
IdentifierQuoteChar='"'
PBSystemOwner='sys,dbo'
PBUseProcOwner='YES'
SQLSrvrTSName='YES'
SQLSrvrTSQuote='YES'
SQLSrvrTSDelimit='YES'
ForeignKeyDeleteRule='Disallow if Dependent Rows
                      Exist (RESTRICT),Delete any Dependent Rows
                      (CASCADE),Set Dependent Columns to NULL
                      (SET NULL)'
TableListType='GLOBAL TEMPORARY'
```

- 3 Find the name of the section in the PBODB90 initialization file that contains function information for your back-end DBMS.

To find this section, look for a line similar to the following in the DBMS Driver/DBMS Settings entry:

```
PBFunctions='section_name'
```

For example, the following line in the DBMS Driver/DBMS Settings entry for Adaptive Server Anywhere indicates that the name of the Functions section is ASA\_FUNCTIONS:

```
PBFunctions='ASA_FUNCTIONS'
```

- 4 Find the Functions section for your back-end DBMS in the PBODB90 initialization file.

For example, here is the Functions section for Adaptive Server Anywhere:

```
;*****
;Functions
;*****
[ASA_FUNCTIONS]
AggrFuncs=avg(x),avg(distinct x),count(x),
          count(distinct x),count(*),list(x),
          list(distinct x),max(x),max(distinct x),
          min(x),min(distinct x),sum(x),sum(distinct
x)
Functions=abs(x),acos(x),asin(x),atan(x),

atan2(x,y),ceiling(x),cos(x),cot(x),degrees(x),
          exp(x),floor(x),log(x),log10(x),
          mod(dividend,divisor),pi(*),power(x,y),
          radians(x),rand(),rand(x),
          remainder(dividend,divisor),round(x,y),
          sign(x),sin(x),sqrt(x),tan(x),
          "truncate"(x,y),ascii(x),byte_length(x),
          byte_substr(x,y,z),char(x),char_length(x),

charindex(x,y),difference(x,y)insertstr(x,y,z),
          lcase(x),left(x,y),length(x), locate(x,y,z),

lower(x),ltrim(x),patindex('x',y),repeat(x,y),
          replicate(x,y),right(x,y),rtrim(x),
          similar(x,y),soundex(x),space(x),str(x,y,z),
          string(x,...),stuff(w,x,y,z),substr(x,y,z),
          trim(x),ucase(x),upper(x),date(x),
          dateformat(x,y),datetime(x,y),day(x),
          dayname(x),days(x),dow(x),hour(x),hours(x),
          minute(x),minutes(x),minutes(x,y),month(x),
          monthname(x),months(x),months(x,y),now(*),

quarter(x),second(x),seconds(x),seconds(x,y),

today(*),weeks(x),weeks(x,y),year(x),years(x),
          years(x,y),ymd(x,y,z),dateadd(x,y,z),
          datediff(x,y,z),datetime(x,y),datepart(x,y),
          getdate(),cast(x as y),convert(x,y,z),
          hextoint(x),inttohex(x),
          connection_property(x,...),datalength(x),
          db_id(x),db_name(x),db_property(x),
```



```

next_connection(x),next_database(x),
property(x),property_name(x),
property_number(x),property_description(x),
argn(x,y,...),coalesce(x,...),
estimate(x,y,z),estimate_source(x,y,z),
experience_estimate(x,y,z),ifnull(x,y,z),
index_estimate(x,y,z),isnull(x,...),
number(*),plan(x),traceback(*)

```

- 5 To add a new function, type a comma followed by the function name at the end of the appropriate function list, as follows:
  - **Aggregate functions** Add aggregate functions to the end of the AggrFuncs list.
  - **All other functions** Add all other functions to the end of the Functions list.

---

### Case sensitivity

If the back-end DBMS you are using is case sensitive, be sure to use the required case when you add the function name.

---

The following example shows a new function for Adaptive Server Anywhere added at the end of the Functions list:

```

;*****
;Functions
;*****
[ASA_FUNCTIONS]
AggrFuncs=avg(x),avg(distinct x),count(x),
count(distinct x),count(*),list(x),
list(distinct x),max(x),max(distinct x),
min(x),min(distinct x),sum(x),sum(distinct
x)
Functions=abs(x),acos(x),asin(x),atan(x),

atan2(x,y),ceiling(x),cos(x),cot(x),degrees(x),
exp(x),floor(x),log(x),log10(x),
mod(dividend,divisor),pi(*),power(x,y),
radians(x),rand(),rand(x),
...
number(*),plan(x),traceback(*),newfunction()

```

- 6 Save your changes to the PBODB90 initialization file.

## Adding functions to a new section in the file

If entries for your back-end DBMS *do not exist* in the PBODB90 initialization file, use the following procedure to create the required sections and add the appropriate functions.

---

### Before you start

For more about the settings to supply for your back-end DBMS in the PBODB90 initialization file, read the comments at the end of the file.

---

❖ **To add functions to a new section in the PBODB90 initialization file:**

- 1 Open the PBODB90 initialization file in one of the following ways:
  - Use the File Editor in PowerBuilder. (For instructions, see the *User's Guide*.)
  - Use any text editor outside PowerBuilder.
- 2 Edit the DBMS Driver/DBMS Settings section of the PBODB90 initialization file to add an entry for your back-end DBMS.

---

### Finding the name

The name required to identify the entry for your back-end DBMS in the DBMS Driver/DBMS Settings section is in the ODBC initialization file.

For information about the name and location of the ODBC initialization file on your platform, see “How PowerBuilder accesses the data source” on page 27.

---

Make sure that you:

- Follow the instructions in the comments at the end of the PBODB90 initialization file.
- Use the same syntax as existing entries in the DBMS Driver/DBMS Settings section of the PBODB90 initialization file.
- Include a section name for PBFunctions.

For example, here is the relevant portion of an entry for a DB2/2 database:

```
;*****
;DBMS Driver/DBMS Settings
;*****
[DB2/2]
...
PBFunctions='DB22_FUNCTIONS'
...
```

- 3 Edit the Functions section of the PBODB90 initialization file to add an entry for your back-end DBMS.

Make sure that you:

- Follow the instructions in the comments at the end of the PBODB90 initialization file.
- Use the same syntax as existing entries in the Functions section of the PBODB90 initialization file.
- Give the Functions section the name that you specified for PBFunctions in the DBMS Driver/DBMS Settings entry.

For example:

```
;*****
;Functions
;*****
[DB22_FUNCTIONS]
AggrFuncs=avg(),count(),list(),max(),min(),sum()
Functions=curdate(),curtime(),hour(), ...
```

- 4 Type a comma followed by the function name at the end of the appropriate function list, as follows:
  - **Aggregate functions** Add aggregate functions to the end of the AggrFuncs list.
  - **All other functions** Add all other functions to the end of the Functions list.

---

#### Case sensitivity

If the back-end DBMS you are using is case sensitive, be sure to use the required case when you add the function name.

---

The following example shows (in bold) a new DB2/2 function named `substr()` added at the end of the Functions list:

```
;*****  
;Functions  
;*****  
[DB22_FUNCTIONS]  
AggrFuncs=avg(),count(),list(),max(),min(),sum()  
Functions=curdate(),curtime(),hour(), substr()
```

- 5 Save your changes to the PBODB90 initialization file.

# Index

## A

- accessing databases
  - ODBC data sources 27
  - troubleshooting any connection 158
  - troubleshooting JDBC connections 184
  - troubleshooting ODBC connections 172
- Adaptive Server Anywhere ODBC Configuration dialog box 35
- API conformance levels for ODBC 23
- applications
  - connecting to databases from 150
  - in database interface connections 58
  - in ODBC connections 19
  - setting AutoCommit and Lock 154
  - setting database preferences 151
  - setting DBParm parameters 140, 143
  - tracing any database connection from 162
  - tracing JDBC connections from 186
  - tracing ODBC connections from 174
  - using Preview tab to set connection options 10, 125, 139, 140, 151
  - using Preview tab to set trace options 163, 175, 187
- Auto Commit Mode checkbox in Database Profile Setup dialog box 145
- AutoCommit database preference
  - displayed on Preview tab 151
  - setting in PowerBuilder script 151
- AutoCommit database preference, setting in database profiles 145
- AutoCommit Transaction object property 153

## B

- backquotes, not supported as delimiter 26
- basic procedures
  - defining database interfaces 61
  - editing database profiles 125

- importing and exporting database profiles 130
- preparing databases for use with database interfaces 60
- preparing ODBC data sources 25
- selecting a database profile to connect 121
- setting database preferences 137, 144
- setting DBParm parameters 137, 139
- sharing database profiles 126
- starting JDBC Driver Manager Trace 186
- starting ODBC Driver Manager Trace 174
- steps for connecting 3
- stopping Database Trace 166
- stopping JDBC Driver Manager Trace 189
- stopping ODBC Driver Manager Trace 178
- Browse button, in Adaptive Server Anywhere ODBC Configuration dialog box 36

## C

- case sensitivity, in PBODB90 initialization file 207, 209
- client software
  - DirectConnect 111
  - Informix 61, 65
  - Microsoft SQL Server 70
  - Oracle 75
  - Sybase Adaptive Server Enterprise 89
- columns
  - identity, Sybase Adaptive Server Enterprise 86
  - in extended attribute system tables 134
  - special timestamp, in Sybase Adaptive Server Anywhere 36
  - SQL naming conventions 26
- conformance levels for ODBC drivers
  - API 23
  - recommendations for 22
  - SQL 23
- Connect DB at Startup checkbox in Database Preferences property sheet 148

- Connect DB at Startup database preference 148
  - connect descriptors, Oracle
    - about 77
    - syntax and example 77
  - connect strings, ODBC
    - about 29
    - DSN (data source name) value 29
  - connect strings, Oracle 77
  - connecting to databases
    - about 120, 123
    - and extended attribute system tables creation 132
    - at startup or from a painter 120
    - basic steps for 3
    - by selecting a database profile 121
    - during application execution 150
    - troubleshooting any connection 158
    - troubleshooting JDBC connections 184
    - troubleshooting ODBC connections 172
    - using database profiles 120
    - when connections occur 3
  - ConnectOption DBParm parameter 175
  - ConnectionString DBParm parameter
    - about 29
    - DSN (data source name) value 29
    - in ODBC connections 29
  - conventions x
  - Core API conformance level for ODBC 23
  - Core SQL conformance level for ODBC 23
  - CT-Library client software for DirectConnect 112
  - CT-Library client software for Sybase Adaptive Server Enterprise 85, 89
  - CT-Library client software for Sybase Systems 51
- D**
- data providers, and OLE DB interface 48
  - data providers, obtaining 50
  - database connections for transactional components 195
  - database interfaces
    - about 58
    - connecting to databases 121
    - connection components 58
    - creating database profiles 8, 26
    - defining 61
    - DirectConnect 105
    - editing database profiles 125
    - importing and exporting database profiles 130
    - Informix 61
    - JDBC 40
    - Microsoft SQL Server 68
    - not supported in PowerBuilder Professional and PowerBuilder Desktop 58
    - Oracle 72
    - preparing databases 60
    - sharing database profiles 126
    - Sybase Adaptive Server Enterprise 85
    - troubleshooting 158
  - Database painter, changing SQL terminator character 79
  - database preferences
    - AutoCommit 145, 151
    - AutoCommit and Lock displayed on Preview tab 151
    - how to set 137
    - Keep Connection Open 148
    - Lock 145, 151
    - Read Only 135, 148
    - setting in Database Preferences property sheet 147
    - setting in database profiles 10, 145
    - setting in PowerBuilder scripts 151
    - Shared Database Profiles 127, 148
    - SQL Terminator Character 148
    - Use Extended Attributes 135, 148
    - using ProfileString function to read 154
  - Database Preferences button 127
  - Database Preferences property sheet
    - about 147
    - General property page, values for 127, 148
    - SQL Terminator Character box 79
  - Database Profile button 121
  - Database Profile Setup dialog box
    - about 10
    - Auto Commit Mode checkbox 145
    - character limit for DBParm strings 139
    - editing profiles 125
    - Generate Trace checkbox 166
    - Isolation Level box 145
    - JDBC Driver Manager Trace, stopping 189
    - ODBC Driver Manager Trace, stopping 178
    - Preview tab 10, 125, 139, 140, 151, 163, 175, 187
    - supplying sufficient information to connect 11

- Trace File box 173, 185
- Trace JDBC Calls checkbox 185
- Trace ODBC API Calls checkbox 173
- database profiles
  - about 8, 120
  - character limit for DBParm strings 139
  - connect string for ODBC data sources 29
  - creating 8
  - Database Profile Setup dialog box 10
  - Database Profiles dialog box 9
  - DBMS value for ODBC data sources 29
  - editing 125
  - exporting 131
  - importing 130
  - importing and exporting 130
  - JDBC Driver Manager Trace, starting 185
  - JDBC Driver Manager Trace, stopping 189
  - Microsoft SQL Server database interface 71
  - ODBC Driver Manager Trace, stopping 178
  - Oracle database interfaces 77
  - reasons to use 120
  - selecting in Database Profiles dialog box 121
  - server name for Sybase Open Client directory services 96
  - setting database preferences 10
  - setting DBParm parameters 10, 140
  - setting Isolation Level and AutoCommit Mode 145
  - shared 126
  - shared, maintaining 129
  - shared, saving in local PowerBuilder or InfoMaker initialization file 129
  - shared, selecting to connect 128
  - shared, setting up 127
  - suppressing password display 124
  - Sybase Adaptive Server Enterprise database interface 91
  - Sybase DirectConnect interface 114
- Database Profiles dialog box
  - about 9, 121
  - displaying shared profiles 128
- Database section in PowerBuilder and InfoMaker initialization files 123, 168
- Database Trace
  - about 158
  - annotating the log 169
  - deleting or clearing the log 169
  - log file contents 159
  - log file format 160
  - sample output 170
  - specifying a nondefault log file 167
  - starting in PowerBuilder scripts 162
  - stopping in PowerBuilder scripts 166
  - syntax displayed on Preview tab 163
  - viewing the log 169
- databases
  - accessing 27
  - basic steps for connecting 3
  - connecting at startup or from a painter 120
  - connecting with database profiles 120, 121
  - in database interface connections 58
  - logging on for the first time 132
  - when connections occur 3
- datatypes
  - Adaptive Server 86
  - conversion in PowerBuilder scripts 87
  - DirectConnect 110
  - Informix 62
  - Microsoft SQL Server 68
  - Oracle 72
  - special timestamp, Transact-SQL 36
  - SQL Server 68
  - Sybase Adaptive Server Enterprise 86
- DateTime datatype, Informix 62
- DB2/CS, IBM, DB2SYSPB.SQL script, using 114
- DB2/MVS, IBM
  - accessing through DirectConnect 105
  - accessing through Open ServerConnect 105
  - DB2SYSPB.SQL script, using 114
- DB2SYSPB.SQL script 115
- DBMS
  - backend, adding ODBC functions for 204
  - entries in PBODB90 initialization file 204, 208
  - system tables, displaying 132
  - trace keyword, adding to PowerBuilder application script 164
  - trace keyword, displayed on Preview tab 163
  - trace keyword, removing from PowerBuilder application script 166
  - value in database profiles 29

- DBMS identifier
    - DIR 105
    - IN9 61
    - MSS 68
    - O84 72
    - O90 72
    - SYC 85
  - DBParm parameters
    - character limit for strings in database profiles 139
    - ConnectOption 175
    - ConnectString 29
    - displayed on Preview tab 10, 125, 139, 140
    - for Sybase Open Client directory services 98
    - for Sybase Open Client security services 93
    - how to set 137
    - in ODBC connections 29
    - PBCatalogOwner 116
    - setting in database profiles 10, 140
    - setting in PowerBuilder scripts 140
    - using ProfileString function to read 143
  - DBParm Transaction object property 142
  - DBTraceFile entry in PowerBuilder and InfoMaker
    - initialization files 168
  - DECLARE PROCEDURE statement 78
  - defining database interfaces
    - about 61
    - DirectConnect 114
    - editing database profiles 125
    - importing and exporting database profiles 130
    - Microsoft SQL Server 71
    - Oracle 77
    - sharing database profiles 126
    - Sybase Adaptive Server Enterprise 91
  - defining ODBC data sources
    - about 26
    - creating configurations and database profiles 26
    - editing database profiles 125
    - multiple data sources 29
    - sharing database profiles 126
    - Sybase Adaptive Server Anywhere 35
  - DIR DBMS identifier 105
  - DirectConnect interface see Sybase DirectConnect interface
  - directory services, Sybase Open Client see Sybase Open Client directory services
  - display formats, in extended attribute system tables 134
  - DIT base for Sybase Open Client directory services 96
  - DLL files
    - in database interface connections 58
    - in JDBC connections 41
    - in ODBC connections 19
    - ODBC.DLL 19
    - ODBC32.DLL 19
    - PBODB90.DLL 19
  - DSN (data source name) value, in ODBC connect strings 29
- ## E
- EAS Demo DB 19
  - edit styles, in extended attribute system tables 134
  - editing
    - database profiles 125
    - PBODB90 initialization file 203
    - shared database profiles 129
  - exporting a database profile 131
  - extended attribute system tables
    - about 114, 131
    - contents 134
    - controlling creation with Use Extended Attributes
      - database preference 135
    - controlling permissions 136
    - controlling updates with Read Only database
      - preference 135
    - creating in DB2 databases 114
    - displaying 132
    - ensuring proper creation 132
    - PBOwner in DB2SYSPB.SQL script 116
  - Extended SQL conformance level for ODBC 23
- ## F
- FreeDBLibraries 6
  - functions, ODBC
    - adding to existing section in PBODB90 initialization file 204
    - adding to new section in PBODB90 initialization file 208



**G**

- General property page in Database Preferences
  - property sheet 127, 148
- Generate Trace checkbox in Database Profile Setup
  - dialog box 166
- granting permissions on extended attribute system
  - tables 136

**H**

- help 25
  - data source 17
  - Database Trace, using 158
  - FaxLines 57
  - for ODBC drivers 25, 30
  - Java website 39
  - JDBC Driver Manager Trace, using 184
  - JDBC website 39, 47
  - Microsoft Universal Data Access 47
  - ODBC Driver Manager Trace, using 172
  - online Help, using 17, 25
- heterogeneous cross-database joins 99

**I**

- IBM database interface, DB2SYSPB.SQL script, using 114
- identity columns and datatype
  - Sybase Adaptive Server Enterprise 86
- importing a database profile 130
- IN9 DBMS identifier 61
- InfoMaker initialization file
  - setting Shared Database Profiles database
    - preference 127
- Informix client software 61, 65
- Informix IN9 database interface
  - client software required 65
  - connection components 64
  - databases supported 61
  - datatypes supported 62
  - installing 66
  - preparing the database 64
  - verifying the connection 66

## initialization files

- DBMS\_PROFILES section 129
- in ODBC connections 27
- locating when sharing database profiles 126
- ODBC 28
- ODBCINST 27
- PBODB90, adding functions to 203
- reading DBMS value from 165, 167
- reading DBParm values from 143, 177, 179, 188, 190
- specifying nondefault Database Trace log 167
- storing connection parameters 123
- suppressing password display 124
- installing
  - Java virtual machines 42
  - ODBC drivers 24
- INTERSOLV ODBC drivers
  - backquote delimiters not supported 26
  - displaying Help 25, 30
  - translators, selecting 30
- interval datatype, Informix 63
- Isolation Level box in Database Profile Setup dialog box 145
- isolation levels and lock values, setting in database
  - profiles 145
- ISQL, using to install stored procedures 103

**J**

- Java virtual machines, installing 42
- Java Web site 39
- JDBC connections, troubleshooting 184
- JDBC database interface, troubleshooting 184
- JDBC Driver Manager Trace
  - about 184
  - availability on different platforms 184
  - performance considerations 184
  - specifying a nondefault log file 185
  - starting in database profiles 185
  - starting in PowerBuilder scripts 186
  - stopping in database profiles 189
  - stopping in PowerBuilder scripts 190
  - syntax displayed on Preview tab 187
  - viewing the log 191
- JDBC drivers, troubleshooting 184

JDBC interface  
  about 39  
  components 41  
  data types supported 43  
  database server configuration 43  
  DLL files required 41  
  drivers 42  
  Java classes package 42  
  Java virtual machines 42  
  PBJDB90.DLL 43  
  registry entries 42  
  specifying connection parameters 45  
  using 41  
JDBC Web site 39  
JDBC.LOG file  
  about 184  
  leaving open 191  
  performance considerations 184  
  using nondefault log file instead 185  
  viewing 191

## K

Keep Connection Open checkbox in Database Preferences  
  property sheet 148  
Keep Connection Open database preference 148

## L

large object, as output parameter in Oracle stored procedure 82  
Level 1 API conformance level for ODBC 23  
Lock database preference  
  displayed on Preview tab 151  
  setting in database profiles 145  
  setting in PowerBuilder script 151  
Lock Transaction object property 153  
lock values and isolation levels, setting in database profiles 145  
LOG files  
  JDBC.LOG 184, 191  
  PBTRACE.LOG 158, 168  
  specifying nondefault for Database Trace 167

  specifying nondefault for JDBC Driver Manager  
    Trace 185  
  specifying nondefault for ODBC Driver Manager  
    Trace 173  
  SQL.LOG 172, 180  
logging on to databases for the first time 132

## M

maintaining shared database profiles 129  
Microsoft Data Link, using with OLE DB interface 54  
Microsoft SQL Server database interface  
  client software required 70  
  connection components 69  
  datatypes supported 68  
  defining 71  
  installing 71  
  preparing the database 69  
  verifying the connection 71  
  versions supported 68  
Microsoft Universal Data Access website 47  
Minimum SQL conformance level for ODBC 23  
MSS DBMS identifier 68  
MTS, database connections for transactional  
  components 199  
multiple ODBC data sources, defining 29  
multiple-tier ODBC drivers 21

## N

naming conventions, tables and columns 26

## O

O73 DBMS identifier 72  
O84 DBMS identifier 72  
O84 Oracle 8.0.4 Driver 72  
O90 DBMS identifier 72  
O90 Oracle 9i Driver 72  
ODBC 25

- ODBC (Open Database Connectivity)
  - about 18
  - components 19
  - defining data sources 26
  - defining multiple data sources 29
  - driver conformance levels 22
  - drivers from other vendors, using 24
  - ODBC initialization file 28
  - ODBCINST initialization file 27
  - preparing data sources 25
  - translators, selecting for drivers 30
- ODBC connect strings
  - about 29
  - DSN (data source name) value 29
- ODBC data sources
  - accessing 27
  - creating configurations and database profiles 26
  - defining 26
  - defining multiple 29
  - editing database profiles 125
  - in ODBC connections 19
  - in ODBC initialization file 28
  - in ODBCINST initialization file 27
  - PBODB90 initialization file 203
  - preparing 25
  - sharing database profiles 126
  - Sybase Adaptive Server Anywhere 32
  - translators, selecting for drivers 30
  - troubleshooting 158, 172
- ODBC Driver Manager 19
- ODBC Driver Manager Trace
  - about 172
  - performance considerations 172
  - sample output 180
  - setting with ConnectOption DBParm 175
  - specifying a nondefault log file 173
  - starting in database profiles 173
  - starting in PowerBuilder scripts 174
  - stopping in database profiles 178
  - stopping in PowerBuilder scripts 179
  - syntax displayed on Preview tab 175
  - viewing the log 180
- ODBC drivers
  - about 18
  - and ODBC initialization file 28
  - and ODBCINST initialization file 27
  - API conformance levels 23
  - conformance levels, recommendations for 22
  - displaying Help 17, 25, 30
  - from other vendors 24
  - in ODBC connections 19
  - installing 24
  - multiple-tier 21
  - PBODB90 initialization file 203
  - single-tier 21
  - SQL conformance levels 23
  - Sybase Adaptive Server Anywhere 32
  - translators, selecting 30
  - troubleshooting 158, 172
  - using 19
  - with PowerBuilder Desktop 24
- ODBC functions
  - adding to existing section in PBODB90 initialization file 204
  - adding to new section in PBODB90 initialization file 208
- ODBC initialization file
  - about 28
  - and PBODB90 initialization file 208
- ODBC interface
  - about 18
  - connecting to data sources 121
  - ConnectOption DBParm, using 175
  - DLL files required 19
  - initialization files required 27
  - ODBC initialization file 28
  - ODBCINST initialization file 27
  - troubleshooting 158, 172
  - using 19
- ODBC interface, PBODB90 initialization file 203
- ODBC.DLL file 19
- ODBC32.DLL file 19
- ODBCINST initialization file 27
- OLE DB interface 48
  - components 49
  - data provider 48
  - getting help 47
  - installing data providers 52
  - installing Microsoft Data Access Components 52
  - object interfaces supported 48
  - obtaining data providers, from other vendors 50
  - obtaining data providers, from Sybase 50

- PBOLE90.DLL 51
  - specifying connection parameters 53
  - using Data Link 54
- Open Client software, Sybase 51, 89, 112
- Oracle database interfaces
  - client software required 75
  - connect strings or descriptors, specifying 77
  - connection components 74
  - datatypes supported 72
  - defining 77
  - installing 76
  - preparing the database 74
  - using Oracle stored procedures 78
  - verifying the connection 76
  - versions supported 72
- Oracle SQL\*Net client software 75
- Oracle stored procedure
  - using LOB output parameter 82

## P

- passwords, suppressing display 124
- PBCatalogOwner DBParm parameter, and
  - DB2SYSPB.SQL script 116
- pbcatcol table 134
- pbcatedt table 134
- pbcatfmt table 134
- pbcattbl table 134
- pbcatvld table 134
- PBIN990.DLL file 64
- PBJDB90.DLL 43
- PBMSS90.DLL file 69
- PBO8490.DLL file 72
- PBO9090.DLL file 72
- PBODB90 initialization file
  - about 203
  - adding functions to existing section 204
  - adding functions to new section 208
  - case sensitivity 207, 209
  - finding DBMS section names in ODBC initialization file 208
  - special timestamp column support 38
- PBODB90.DLL file 19
- PBOLE90.DLL file 51
- PBSYC.SQL script
  - about 100
  - compared to PBSYC2.SQL script 103
  - finding 100
  - running with ISQL 103
  - running with WISQL 104
  - when to run 100
- PBSYC2.SQL script
  - about 102
  - compared to PBSYC.SQL script 103
  - finding 100
  - running with ISQL 103
  - running with WISQL 104
  - when to run 102
- PBTRACE.LOG file
  - about 158
  - annotating 169
  - contents 159
  - deleting or clearing 169
  - format 160
  - leaving open 169
  - sample output 170
  - using nondefault log file instead 167
  - viewing 169
- permissions, granting on system tables 136
- PowerBuilder Desktop
  - database interfaces, not supported 58
  - ODBC drivers, using 24
- PowerBuilder Enterprise, ODBC drivers, using 24
- PowerBuilder initialization file
  - about 125
  - locating when sharing database profiles 126
  - saving shared database profiles locally 129
  - setting Shared Database Profiles database preference 127
  - specifying nondefault Database Trace log 167
  - suppressing password display 124
- PowerBuilder Professional
  - database interfaces not supported 58
  - ODBC drivers, using 24
- PowerScript syntax, on Preview tab 12
- preparing databases for use with database interfaces
  - about 60
  - DirectConnect 111
  - Informix IN9 64
  - Microsoft SQL Server 69

- Oracle 74
- Sybase Adaptive Server Enterprise 88
- preparing databases for use with Sybase Adaptive Server Enterprise 51
- preparing ODBC data sources
  - about 25
  - Sybase Adaptive Server Anywhere 34
- Preview tab
  - about 10, 12, 125, 139, 140
  - copying AutoCommit and Lock properties 151
  - copying Database Trace syntax 163
  - copying DBParm parameters 139, 140
  - copying DBParm properties 10, 125
  - copying JDBC Driver Manager Trace syntax 187
  - copying ODBC Driver Manager Trace syntax 175
- PRINT statements in SQL Server stored procedures 99
- procedures, basic
  - defining database interfaces 61
  - editing database profiles 125
  - importing and exporting database profiles 130
  - preparing databases for use with database interfaces 60
  - preparing ODBC data sources 25
  - selecting a database profile to connect 121
  - setting database preferences 137, 144
  - setting DBParm parameters 137, 139
  - sharing database profiles 126
  - steps for connecting 3
  - stopping Database Trace 166
  - stopping JDBC Driver Manager Trace 189
  - stopping ODBC Driver Manager Trace 178
- profiles, database see database profiles
- ProfileString function
  - setting AutoCommit and Lock in scripts 154
  - setting DBParm parameters in scripts 143
  - starting Database Trace in scripts 165
  - starting JDBC Driver Manager Trace in scripts 188
  - starting ODBC Driver Manager Trace in scripts 177
  - stopping Database Trace in scripts 167
  - stopping JDBC Driver Manager Trace in scripts 190

- stopping ODBC Driver Manager Trace in scripts 179
- Prompt for Database Information checkbox 124

## R

- Read Only check box in Database Preferences property sheet 148
- Read Only database preference 135, 148
- registry, Windows
  - ODBC initialization file 28
  - ODBCINST initialization file 27
- result sets, using Oracle stored procedures 79
- RPCFUNC keyword 78

## S

- scripts, PowerBuilder
  - datatype conversions 87
  - setting database preferences 151
  - setting DBParm values 140
  - starting Database Trace 162
  - starting JDBC Driver Manager Trace 186
  - starting ODBC Driver Manager Trace 174
  - using Preview tab to set connection options 10, 125, 139, 140, 151
  - using Preview tab to set trace options 163, 175, 187
  - using ProfileString function to read 143, 154
- security services, Sybase Open Client see Sybase Open Client security services
- Select Tables dialog box, Show system tables checkbox 132
- Select Translator dialog box 30
- semicolons, as default SQL terminator character 79
- SERVER directory files
  - for creating repository in DB2 databases 115
  - for installing stored procedures in Adaptive Server Enterprise databases 100
- server name, specifying for Sybase Open Client directory services 96

- shared database profiles
  - maintaining 129
  - saving in local PowerBuilder or InfoMaker initialization file 129
  - selecting in Database Profiles dialog box 128
  - setting Shared Database Profiles database preference 127
  - setting up 126, 127
- Shared Database Profiles box in Database Preferences
  - property sheet 127, 148
- Shared Database Profiles database preference 148
- Show system tables checkbox 132
- single-tier ODBC drivers 21
- sp\_pb90table stored procedure
  - in PBSYC.SQL script 101
  - in PBSYC2.SQL script 103
- SQL conformance levels for ODBC 23
- SQL files
  - DB2SYSPB.SQL 115
  - PBSYC.SQL 100
  - PBSYC2.SQL 102
- SQL naming conventions for tables and columns 26
- SQL Terminator Character database preference 148
- SQL terminator character, changing in Database painter 79, 148
- SQL\*Net client software, Oracle 75
- SQL.LOG file
  - about 172
  - leaving open 180
  - performance considerations 172
  - sample output 180
  - using nondefault log file instead 173
  - viewing 180
- SQL\_OPT\_TRACE parameter in ConnectOption DBParm
  - about 175
  - adding to PowerBuilder application script 176
  - changing to SQL\_OPT\_TRACE\_OFF in PowerBuilder application script 179
- SQL\_OPT\_TRACEFILE parameter in ConnectOption DBParm
  - about 175
  - adding to PowerBuilder application script 176
- SQLCA Transaction object
  - setting AutoCommit property 153
  - setting ConnectOption DBParm 176
  - setting DBParm property 142
  - setting Lock property 153
  - setting TraceFile DBParm 188
  - trace keyword in DBMS property 164, 166
- starting
  - Database Trace in PowerBuilder application 162
  - JDBC Driver Manager Trace in development environment 185
  - JDBC Driver Manager Trace in PowerBuilder application 186
  - ODBC Driver Manager Trace in PowerBuilder application 174
- stopping
  - Database Trace in PowerBuilder application 166
  - JDBC Driver Manager Trace in development environment 189
  - JDBC Driver Manager Trace in PowerBuilder application 190
  - ODBC Driver Manager Trace in development environment 178
  - ODBC Driver Manager Trace in PowerBuilder application 179
- stored procedures
  - about 100
  - created by PBSYC.SQL script 101
  - created by PBSYC2.SQL script 103
  - installing in Adaptive Server Enterprise databases 99
  - ISQL, using to install 103
  - not required for Microsoft SQL Server database interface 99
  - running scripts 103
  - where to find scripts 100
  - WISQL, using to install 104
- stored procedures, Oracle
  - about 78
  - changing SQL terminator character 79, 148
  - creating DataWindows and reports 81
  - with result sets, examples 80
  - with result sets, using 79
- stored procedures, SQL Server, using PRINT statements 99
- Sybase Adaptive Server Anywhere
  - accessing remote databases 32
  - adding functions to PBODB90 initialization file 204
  - connection components 33

- creating configurations and database profiles 26
  - defining the data source 35
  - LOG files 34
  - network server, not included 32
  - platforms supported 32
  - preparing to use 34
  - special timestamp columns 36
  - startup options, specifying 36
  - using 19
  - versions supported 32
  - Sybase Adaptive Server Enterprise
    - using SYJ database interface for EAServer 85
  - Sybase Adaptive Server Enterprise database interface
    - client software required 51, 89
    - creating a DW based on a heterogeneous cross-database join 99
    - datatypes supported 86
    - defining 91
    - directory services, using 94
    - identity columns 86
    - installing 90
    - installing stored procedures 99
    - platforms supported 85
    - preparing the database 51, 88
    - security services, using 92
    - verifying the connection 90
    - versions supported 85
  - Sybase DirectConnect interface
    - client software required 111
    - data types supported 110
    - DB2SYSPB.SQL script, using 114
    - defining 114
    - platforms supported 110
    - preparing the database 111
    - using DirectConnect middleware 106
    - using Open ServerConnect middleware 106
    - verifying the connection 113
    - versions supported 110
  - Sybase EAServer
    - database connections for transactional components 196
  - Sybase Open Client directory services
    - about 94
    - DBParm parameters 98
    - requirements for using 95
    - specifying the server name 96
    - third-party directory service providers 94
  - Sybase Open Client security services
    - about 92
    - DBParm parameters, login authentication 93
    - DBParm parameters, per-packet security 94
    - requirements for using 92
    - third-party security mechanisms 92
  - Sybase Open Client software 51
    - about 89, 112
  - Sybase SQL Anywhere *see* Sybase Adaptive Server Anywhere
  - Sybase, getting help from 25
  - sybssystemprocs database, Sybase Adaptive Server Enterprise 103, 104
  - SYC DBMS identifier 85
  - SYSIBM, prohibited as DB2 table owner 116
  - system tables, displaying 132
- T**
- tables
    - extended attribute, creating in DB2 databases 114
    - in extended attributes 134
    - PBOwner in DB2SYSPB.SQL script 116
    - SQL naming conventions 26
    - system, displaying 132
  - technical document, Sybase, getting help from 17
  - technical documents, Sybase, getting help from 57
  - time datatype, Informix 63
  - timestamp, Transact-SQL special 36
  - Trace File box in Database Profile Setup dialog box 173, 185
  - Trace JDBC Calls checkbox in Database Profile Setup dialog box 185
  - trace keyword
    - adding to PowerBuilder application script 164
    - displayed on Preview tab 163
    - removing from PowerBuilder application script 166
  - Trace ODBC API Calls checkbox in Database Profile Setup dialog box 173

## *Index*

- tracing database connections
  - about 157
  - Database Trace 158
  - JDBC Driver Manager Trace 184
  - ODBC Driver Manager Trace 172
  - sample output, Database Trace 170
  - sample output, ODBC Driver Manager Trace 180
- Transaction object, SQLCA
  - setting AutoCommit property 153
  - setting ConnectOption DBParm 176
  - setting DBParm property 142
  - setting Lock property 153
  - setting TraceFile DBParm 188
  - trace keyword in DBMS property 164, 166
- Transact-SQL special timestamp in Sybase Adaptive Server
  - Anywhere 36
- translators, ODBC 30
- troubleshooting database connections
  - about 157
  - Database Trace 158
  - JDBC Driver Manager Trace 184
  - ODBC Driver Manager Trace 172
- typographical conventions x

## **U**

- Unicode
  - support 7
- UNIX, using SYJ database interface for Adaptive Server
  - 85
- Use Extended Attributes checkbox in Database Preferences
  - property sheet 148
- Use Extended Attributes database preference 135, 148

## **V**

- validation rules, in extended attribute system tables 134

## **W**

- WISQL, for installing stored procedures 104