Open Tools from Sybase, Inc.

PowerBuilder

Connecting to Your Database

Version 6

PowerBuilder®

AA0525

October 1997

Contents

About This Book

Subject

This book describes how to connect to a database in the PowerBuilder development environment by using an ODBC data source driver or Powersoft database interface. It gives procedures for preparing, defining, establishing, maintaining, and troubleshooting your database connections *on all supported platforms*.

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.

PART 1

Working with Database Connections

This part describes how to set up, define, establish, and manage database connections accessed through the Powersoft ODBC interface or one of the native Powersoft database interfaces.

CHAPTER 1

Understanding Data Connections

Where you are

> (*Optional*) Get an introduction to database connections
Prepare to use the data source or database
Install the ODBC driver or Powersoft database interface
Define the data source or database
Connect to the data source or database
(*Optional*) Set additional connection parameters
(*Optional*) Troubleshoot the data connection

About this chapter

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

Contents

How to find the information you need

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

This manual describes how to connect to your database *in the PowerBuilder development environment*. The basic steps in this process are the same on *all* supported platforms (Windows, Macintosh, and UNIX). In fact, the only difference you'll notice on the various platforms is the list of databases you can access.

The following table gives an overview of the connection procedure and tells where you can find more detailed information about each step:

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

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)
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	For ODBC data sources: Chapter 2, "Using ODBC Data Sources and Drivers" For Powersoft database interfaces: Chapter 3, "Using Powersoft Database Interfaces"
3	Install the ODBC driver or Powersoft database interface	Install the ODBC data source driver (if supported on your platform) or Powersoft database interface required to access your data	For a list of what's supported on your platform: Appendix A, "Supported ODBC Drivers and Powersoft Database Interfaces"

Step	Action	Details	See
4	Define the data source or database interface	Create the required configuration (for a data source accessed through ODBC) or database profile (for a database accessed through a Powersoft database interface)	For ODBC data sources: Chapter 2, "Using ODBC Data Sources and Drivers" For Powersoft database interfaces: Chapter 3, "Using Powersoft Database Interfaces"
5	Connect to the data source or database	Access the data in PowerBuilder	Chapter 4, "Managing Database Connections"
6	(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	For procedures: Chapter 5, "Setting Additional Connection Parameters" For DBParm descriptions: Chapter 7, "DBParm Parameters" For database preference descriptions: Chapter 8, "Database Preferences"
7	(Optional) Troubleshoot the data connection	If necessary, use the Database Trace and ODBC Driver Manager Trace tools to troubleshoot problems with your connection	Chapter 6, "Troubleshooting Your Connection"

Accessing data in PowerBuilder

There are two ways to access data in the PowerBuilder development environment, depending on your operating system platform:

- Through the Powersoft ODBC interface (if your platform supports it)
- Through one of the native Powersoft database interfaces

Powersoft ODBC interface

On platforms that support it, the Powersoft Open Database Connectivity (ODBC) interface can access any ODBC data source for which you have installed an ODBC-compliant driver. Data that you access with an ODBC driver is called an **ODBC data source**.

Once you install the ODBC driver, the interface communicates with the driver through the ODBC Driver Manager to access the data you need. For example, you can access a Sybase SQL Anywhere (formerly Watcom SQL) data source in PowerBuilder by installing the SQL Anywhere ODBC driver.

On Windows and Macintosh On Windows and Macintosh, the Powersoft ODBC interface *is provided* in PowerBuilder.

On UNIX On UNIX, the only ODBC connections supported in PowerBuilder are those using the INTERSOLV ODBC drivers included with the product. You *cannot* use ODBC drivers obtained from other vendors to access data in PowerBuilder on UNIX.

Powersoft database interfaces

A Powersoft database interface is a native (direct) connection to a database. The connection does *not* go through ODBC to access the database.

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

For example, if you have the appropriate Sybase SQL Server System 10 or System 11 server and client software installed, you can access the database by installing the Powersoft System 10 and System 11 database interface.

For more information

For a complete list of the ODBC drivers and Powersoft database interfaces supplied with the Powersoft Enterprise Series products and the databases they access, see Appendix A, "Supported ODBC Drivers and Powersoft Database Interfaces".

ODBC data sources

What is an ODBC data source?

An **ODBC data source** is data that you access with an ODBC-compliant driver. As defined by the Microsoft Open Database Connectivity (ODBC) standard, an ODBC data source consists of the data and its associated DBMS or file manager, operating system, and (if present) network software. The data source stores and manages the data for your application.

Examples of ODBC data sources

You can access an ODBC data source that resides locally on your computer or remotely on a network server. The following are examples of ODBC data sources accessible in PowerBuilder:

- An Excel or dBASE file on your computer
- A Sybase SQL Anywhere relational database installed on a network server
- An IBM Database 2/Common Server (DB2/CS) database running on a network server that you access through TCP/IP

Platform differences

The ability to access ODBC data sources in the PowerBuilder development environment depends on the platform you are using, as follows:

On this platform	Can you access ODBC data sources?
Windows	Yes, with ODBC drivers supplied by Powersoft or by you
Power Macintosh	Yes, with ODBC drivers supplied by Powersoft or by you
UNIX	Yes, only with the INTERSOLV ODBC drivers supplied by Powersoft. You *cannot* access data in PowerBuilder on UNIX with ODBC drivers obtained from other vendors

Accessing ODBC data sources

If your platform supports it, you can access an ODBC data source in PowerBuilder through the Powersoft ODBC interface. PowerBuilder lets you access both of the following:

- A SQL Anywhere data source
- Other ODBC data sources

In order to access an ODBC data source, the driver needs an **ODBC data source definition** containing the information required to connect, such as the data source name and database location. For some ODBC connections, PowerBuilder creates the data source definition for you. For other connections, you must define the ODBC data source yourself by completing the ODBC setup dialog box for your driver. (Defining an ODBC data source is also called **configuring** the data source.)

On UNIX

In PowerBuilder for UNIX, you define an ODBC data source (such as INFORMIX) by creating a database profile for it. Unlike other platforms, you *cannot* define an ODBC data source on UNIX by completing the ODBC setup dialog box for your driver.

FOR INFO For instructions on connecting to an INFORMIX data source in PowerBuilder for UNIX, see "INTERSOLV INFORMIX on UNIX" on page 72.

For more information

For more about when PowerBuilder creates the ODBC data source definition and when you must create it, see "Summary" on page 15.

For more about how PowerBuilder accesses ODBC data sources, see Chapter 2, "Using ODBC Data Sources and Drivers".

SQL Anywhere

Powersoft Demo Database

On the Windows and Macintosh platforms, PowerBuilder includes a standalone SQL Anywhere database called the Powersoft Demo Database. This database is installed by default, unless you clear this option in the setup program. Initially, you access tables in the Powersoft Demo Database when you use the PowerBuilder tutorial or PowerBuilder code examples.

On UNIX

FOR INFO For instructions on creating the Powersoft Demo Database for use with the PowerBuilder tutorial on UNIX, see the *PowerBuilder Installation Guide (UNIX).*

Accessing SQL Anywhere on different platforms

How you access SQL Anywhere in PowerBuilder depends on the platform you are using, as follows:

On this platform	SQL Anywhere access is
Windows	Through the Powersoft ODBC interface with the SQL Anywhere ODBC driver
Power Macintosh	Through the Powersoft ODBC interface with the SQL Anywhere ODBC driver
UNIX	Unsupported (SQL Anywhere is unavailable on the UNIX platform)

On the Windows and Power Macintosh platforms, a SQL Anywhere database is considered an ODBC data source because you access it with the SQL Anywhere ODBC driver.

Creating SQL Anywhere databases

On all platforms except UNIX, you can create your own SQL Anywhere database for use in your application in either of the following ways:

- Use the Database painter in PowerBuilder running on your computer to create a local SQL Anywhere database
- Create the database some other way, such as with PowerBuilder running on another user's computer, or with SQL Anywhere outside PowerBuilder

---

On UNIX

SQL Anywhere is unavailable on the UNIX platform. Therefore, you cannot create a local SQL Anywhere database in PowerBuilder for UNIX.

The method you use to create a SQL Anywhere database determines how you connect to it in PowerBuilder, as follows.

SQL Anywhere databases created on your computer

When you create a local SQL Anywhere database using PowerBuilder on *your computer*, PowerBuilder automatically creates the ODBC data source definition required to access the new database. It also creates a database profile so you can easily connect to this database at any time.

FOR INFO For instructions on creating a local SQL Anywhere database, see the *PowerBuilder User's Guide*.

FOR INFO For an introduction to database profiles, see "Using database profiles" on page 14.

SQL Anywhere databases not created on your computer

When you access a SQL Anywhere database created with PowerBuilder on *another user's computer*, or with SQL Anywhere outside this product, PowerBuilder does *not* automatically create the ODBC data source definition for you. You must define the data source yourself by following these general steps:

1 Install the SQL Anywhere ODBC driver on your computer.

   You can install the driver by default with PowerBuilder, or you can install it later when you need it.

2 Prepare to use the SQL Anywhere data source outside PowerBuilder before you connect.

   FOR INFO For instructions, see "Sybase SQL Anywhere" on page 106.

3 Define the SQL Anywhere data source in PowerBuilder.

   This involves completing the SQL Anywhere ODBC Configuration dialog box.

   FOR INFO For instructions, see "Sybase SQL Anywhere" on page 106.

When you define the data source, PowerBuilder automatically creates the accompanying database profile. You can select this profile at any time to connect to the data source.

Other ODBC data sources

In addition to SQL Anywhere, PowerBuilder on most platforms provides access to many other ODBC data sources through ODBC-compliant drivers.

Platform differences

Whether you can access ODBC data sources *other* than SQL Anywhere in PowerBuilder depends on the platform you are using, as follows:

On this platform	Can you access ODBC data sources other than SQL Anywhere?
Windows	Yes
Power Macintosh	Yes
UNIX	Only those accessible with the INTERSOLV ODBC drivers supplied by Powersoft. On UNIX, you *cannot* access data in PowerBuilder with ODBC drivers obtained from other vendors

What you do

When you access an ODBC data source with a driver *other than* the SQL Anywhere ODBC driver, PowerBuilder does not automatically create the ODBC data source definition for you. You must create it yourself by following these general steps:

1 Install the ODBC driver that accesses the data source on your computer.

FOR INFO For instructions, see "Using ODBC drivers" next.

2 Prepare to use the ODBC data source outside PowerBuilder before you connect.

FOR INFO For instructions, see the section for your data source in Chapter 2, "Using ODBC Data Sources and Drivers".

3 Define the ODBC data source in PowerBuilder.

This involves completing the ODBC setup dialog box for your ODBC driver.

FOR INFO For instructions, see the section for your data source and driver in Chapter 2, "Using ODBC Data Sources and Drivers".

When you define the data source in PowerBuilder, a database profile is automatically created. You can select this profile to connect to the data source.

Using ODBC drivers

How to obtain ODBC drivers

On the Windows and Macintosh platforms, there are two ways that you can obtain ODBC drivers for use with PowerBuilder:

- **From Powersoft (recommended)** Install one or more of the ODBC drivers supplied with PowerBuilder. You can do this when you first install PowerBuilder, or sometime later when you need the driver. Using an ODBC driver supplied by Powersoft ensures that the driver was tested and found to work with PowerBuilder.
- **From another vendor** If you are using PowerBuilder Enterprise, PowerBuilder Professional, or InfoMaker, you can access data with a Level 1 or higher ODBC-compliant driver obtained from a vendor other than Powersoft. In most cases, the driver will work with PowerBuilder. However, Powersoft may not have tested the driver to verify this.

On UNIX

You *cannot* use ODBC drivers obtained from other vendors with PowerBuilder on the UNIX platform.

PowerBuilder Desktop

If you are using PowerBuilder Desktop, you can access data using the ODBC drivers that are shipped with the product. Unlike PowerBuilder Enterprise and PowerBuilder Professional, you *cannot* use an ODBC driver obtained from another vendor with PowerBuilder Desktop.

Using existing Microsoft ODBC drivers If you have already have Version 2.0 or higher 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 INTERSOLV drivers is recommended
PowerBuilder Desktop comes with INTERSOLV ODBC drivers for all of these data sources *except Access*. Since the INTERSOLV drivers have been tested to work with PowerBuilder Desktop, you should use the INTERSOLV drivers whenever possible to access these data sources.

Supported ODBC drivers

Powersoft provides different sets of ODBC drivers on different operating system platforms.

FOR INFO For more information, see Appendix A, "Supported ODBC Drivers and Powersoft Database Interfaces". In addition, check the *Release Notes* for the latest information about supported ODBC drivers on each platform.

Powersoft database interfaces

On *all* supported platforms, PowerBuilder provides a variety of Powersoft database interfaces as a way to access data. A **Powersoft database interface** is a native (direct) connection to a database in PowerBuilder. If your site uses one of these databases, and if you have the required network, database server, and database client software installed, you can access the data by installing the corresponding Powersoft database interface supplied with PowerBuilder. For example, you can access an Oracle database by installing one of the Powersoft Oracle database interfaces.

A Powersoft database interface does *not* go through ODBC to access a database. Therefore, you do not complete an ODBC setup dialog box to define the data source. Instead, you create a database profile in which you specify connection parameters for accessing the database.

FOR INFO For more about how PowerBuilder accesses data through Powersoft database interfaces, see Chapter 3, "Using Powersoft Database Interfaces". In addition, check the *Release Notes* for the latest information about supported database interfaces on each platform.

Using database profiles

What is a database profile?

A **database profile** is a named set of parameters stored in your PowerBuilder initialization file that defines a connection to a particular database in the PowerBuilder development environment. As described in the preceding sections, database profiles are created automatically for some types of data connections and by you for others.

What you can do

Using database profiles is the easiest way for you to manage your 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

Platform differences

The steps for creating and using database profiles in PowerBuilder are the same on all supported platforms. The only difference among platforms is the name and location of the initialization file where your profiles are stored.

Platform	Filename	Location
Windows	PB.INI	PowerBuilder product directory
Macintosh	PowerBuilder Preferences	System Folder:Preferences: Powersoft 6.0 Preferences folder
UNIX	.pb.ini (hidden file)	$HOME directory

For more information

For instructions on using database profiles, see Chapter 4, "Managing Database Connections".

Summary

The type of data connection determines whether PowerBuilder automatically creates the ODBC data source definition and database profile or you do, as summarized in the following table. (ODBC data source definitions apply only when you are using PowerBuilder on the Windows and Power Macintosh platforms.)

For a connection to	ODBC data source definition is created	Database profile is created
SQL Anywhere database created with PowerBuilder on your computer	Automatically by PowerBuilder	Automatically by PowerBuilder
SQL Anywhere database created with PowerBuilder on another user's computer, or with SQL Anywhere outside these products	By you	Automatically by PowerBuilder
ODBC data source using ODBC driver other than SQL Anywhere	By you	Automatically by PowerBuilder
Powersoft database interface	—	By you

Defining the ODBC data source outside PowerBuilder

The preceding table assumes that you are using the Configure ODBC dialog box in PowerBuilder to define the ODBC data source.

If you use a tool such as the ODBC Administrator (on Windows) or ODBC Setup control panel (on Macintosh) to define the data source *outside* PowerBuilder and then want to access the data source from *within* PowerBuilder, you must create the database profile yourself.

FOR INFO For instructions, see "Creating a profile for an existing ODBC data source" on page 317.

What to do next

FOR INFO For instructions on preparing to use and defining an ODBC data source, see Chapter 2, "Using ODBC Data Sources and Drivers".

FOR INFO For instructions on preparing to use and defining a Powersoft database interface, see Chapter 3, "Using Powersoft Database Interfaces".

CHAPTER 2

Using ODBC Data Sources and Drivers

Where you are

(*Optional*) Get an introduction to database connections
> Prepare to use the data source or database
> Install the ODBC driver or Powersoft database interface
> Define the data source or database
Connect to the data source or database
(*Optional*) Set additional connection parameters
(*Optional*) Troubleshoot the data connection

About this chapter

This chapter gives an introduction to the Powersoft ODBC interface. It then describes the following for each supported ODBC data source so you can connect to it in the PowerBuilder development environment:

- Preparing to use the data source (includes installing the ODBC driver)
- Defining the data source

Contents

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 43. This Help provides important details about using the data source.
- Check for a FaxLine document that describes how to connect to your ODBC data source. Updated information about connectivity issues is available from the Powersoft FaxLine system and from Powersoft electronic services on CompuServe, FTP, BBS, and the World Wide Web.

Using the Powersoft ODBC interface

On most platforms, you can access a wide variety of ODBC data sources in PowerBuilder. This section describes what you need to know to use Powersoft ODBC connections to access your data in PowerBuilder.

Supported ODBC drivers and data sources

FOR INFO For a complete list of the ODBC drivers supplied with the Powersoft Enterprise Series products and the data sources they access, see Appendix A, "Supported ODBC Drivers and Powersoft Database Interfaces".

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 on to a DBMS
- SQL syntax based on the X/Open and SQL Access Group (SAG) CAE specification (1992)
- A standard representation for data types
- 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.

Platform differences

The capabilities for accessing ODBC data sources differ on each supported PowerBuilder platform. The following sections describe platform-specific considerations when you are using ODBC with PowerBuilder on Windows, Macintosh, and UNIX. Keep these considerations in mind as you read the rest of this chapter.

For up-to-date information

As connectivity to additional ODBC data sources is supported in future releases of PowerBuilder, updated information will be available electronically from Powersoft services on CompuServe, FTP, BBS, and the World Wide Web. You should also check the *Release Notes* for the latest information about supported ODBC drivers for your version of PowerBuilder.

FOR INFO For information about how to access these services, see the Answer Page in PowerBuilder online Help.

Using ODBC on Windows

This section gives an overview of what you can and cannot do when using the Powersoft ODBC interface in PowerBuilder on the Windows platform.

What you can do

All ODBC connectivity features are supported in PowerBuilder on Windows. *You can*:

- Connect to a SQL Anywhere standalone database (including the Powersoft Demo Database) using the SQL Anywhere ODBC driver and the Powersoft ODBC interface
- Create and delete local SQL Anywhere databases.

  FOR INFO For instructions, see the *PowerBuilder User's Guide.*
- Use Powersoft-supplied INTERSOLV ODBC drivers to access your data

  FOR INFO For a list of the ODBC drivers supplied on Windows, see Appendix A, "Supported ODBC Drivers and Powersoft Database Interfaces".
- In all products *except* PowerBuilder Desktop, use Level 1 or higher ODBC-compliant drivers obtained from vendors other than Powersoft to access your data

  FOR INFO See "Obtaining ODBC drivers" on page 29.
- Use the Configure ODBC dialog box and the ODBC setup dialog box for your driver to define ODBC data sources

  FOR INFO See "About defining ODBC data sources" on page 34.

Using ODBC on Macintosh

The PowerBuilder development environment is available on the Power Macintosh platform, using the PowerPC family of microprocessors and the Apple Code Fragment Manager (CFM) runtime architecture.

This section gives an overview of what you can and cannot do when accessing ODBC data sources on Power Macintosh.

Using ODBC on Power Macintosh

What you can do

In PowerBuilder on the Power Macintosh platform, *you can*:

- Connect to a SQL Anywhere standalone database (including the Powersoft Demo Database) using the SQL Anywhere ODBC driver and the Powersoft ODBC interface
- Create and delete local SQL Anywhere databases

  FOR INFO For instructions, see the *PowerBuilder User's Guide*.
- In all products *except* PowerBuilder Desktop, use Level 1 or higher ODBC-compliant drivers obtained from vendors other than Powersoft to access your data

  FOR INFO See "Obtaining ODBC drivers" on page 29.
- Use the Configure ODBC dialog box and the ODBC setup dialog box for your driver to define ODBC data sources

  FOR INFO For instructions, see "About defining ODBC data sources" on page 34.

What you cannot do

In PowerBuilder on the Power Macintosh platform, *you cannot*:

- Use Powersoft-supplied INTERSOLV ODBC drivers to access your data

  PowerBuilder on Power Macintosh does not supply INTERSOLV ODBC drivers to access data sources. You can, however, use ODBC-compliant drivers obtained from vendors other than Powersoft (including INTERSOLV) to access ODBC data sources in all products *except* PowerBuilder Desktop.

Using ODBC on UNIX

This section gives an overview of what you can and cannot do when accessing ODBC data sources in PowerBuilder on UNIX platform. The information applies equally to the Solaris, HP-UX, and AIX versions of PowerBuilder.

What you can do

In PowerBuilder on UNIX, *you can*:

- Use the Powersoft-supplied INTERSOLV ODBC drivers to access data

  On UNIX, the only ODBC connections supported in PowerBuilder are those using the INTERSOLV ODBC drivers included with the product. You *cannot* use ODBC drivers obtained from other vendors to access data.

What you cannot do

In PowerBuilder for UNIX, *you cannot*:

- Connect to, create, or delete local SQL Anywhere databases

  SQL Anywhere is unavailable on the UNIX platform. Therefore, PowerBuilder for UNIX *does not provide* the SQL Anywhere standalone database engine or the ability to create and delete local SQL Anywhere databases.

  Powersoft Demo Database on UNIX

  To use the PowerBuilder tutorial on UNIX, you or your system administrator must create the Powersoft Demo Database yourself by installing the code examples. (The Powersoft Demo Database is *not* a SQL Anywhere standalone database in PowerBuilder on UNIX.)

  FOR INFO For instructions on creating the Powersoft Demo Database for use with the PowerBuilder tutorial on UNIX, see the *PowerBuilder Installation Guide (UNIX)*.

- Use ODBC-compliant drivers obtained from vendors other than Powersoft to access your data
- Use the Configure ODBC dialog box and the ODBC setup dialog box for your driver to define ODBC data sources

  The Configure ODBC dialog box is unavailable in PowerBuilder on UNIX. If you are accessing a data source with one of the INTERSOLV ODBC drivers that Powersoft supplies, you must instead define the data source by creating a database profile.

  FOR INFO For instructions on defining an INFORMIX data source, see "INTERSOLV INFORMIX on UNIX" on page 72.

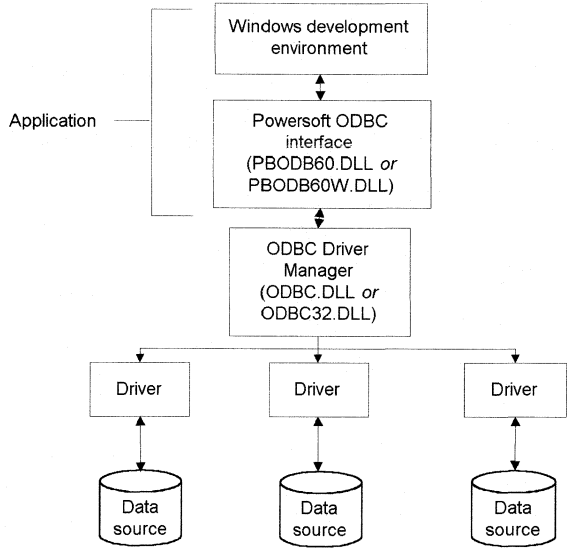
Components of an ODBC connection

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.

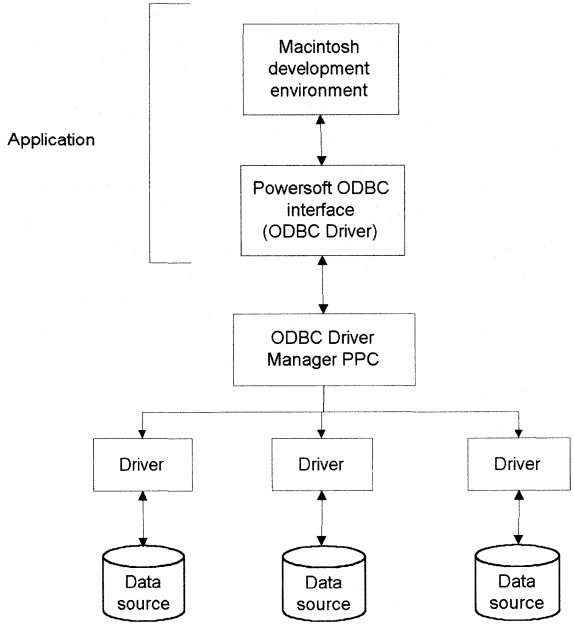
Because ODBC is a standard API, PowerBuilder uses the same interface to access every ODBC data source. As long as a driver is ODBC-compliant, PowerBuilder can access it through the Powersoft ODBC interface to the ODBC Driver Manager.

The following diagrams show the general components of a Powersoft ODBC connection on the Windows and Macintosh platforms. The development environment and the Powersoft ODBC interface work together as the application component.

On Windows



On Macintosh



Component descriptions

The following table gives the provider and a brief description of each ODBC component shown in the diagram:

Component	Provider	What it does
Application	Powersoft	Calls ODBC functions to submit SQL statements, catalog requests, and retrieve results from a data source PowerBuilder uses the same ODBC interface to access all ODBC data sources
ODBC Driver Manager	Microsoft	Installs, loads, and unloads drivers for an application

Component	Provider	What it does
Driver	Driver vendor	Processes ODBC function calls, submits SQL requests to a particular data source, and returns results to an application If necessary, translates an application's request so it conforms to the SQL syntax supported by the backend database FOR INFO See "Types of ODBC drivers" next.
Data source	DBMS or database vendor	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

For more information

For diagrams showing the basic components of *your* ODBC connection, see the section for your driver later in this chapter.

Types of ODBC drivers

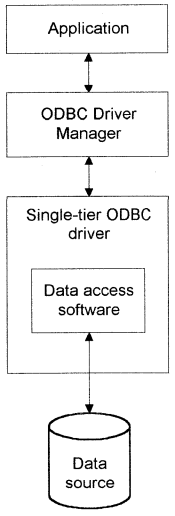
When PowerBuilder is connected to an ODBC data source, you may see messages from the ODBC driver that include the words *single-tier* or *multiple-tier*. These terms refer to the two types of drivers defined by the ODBC standard:

- Single-tier driver
- Multiple-tier driver

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.

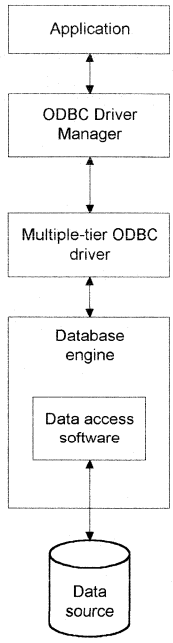
An example of a single-tier ODBC driver is one that accesses Xbase files, such as the INTERSOLV dBASE ODBC driver.



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 SQL Anywhere driver.



Ensuring the proper ODBC driver conformance levels

If you are using PowerBuilder Enterprise or PowerBuilder Professional, you can access data with ODBC drivers obtained from vendors *other* than Powersoft, such as directly from 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 data types

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 X/Open and SAG Call Level Interface (CLI) specification
- **Level 1** Includes all Core API functions as well as several extended functions
- **Level 2** Includes all Core and Level 1 API functions as well as additional extended functions

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

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

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

- Powersoft 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 Powersoft (recommended)** Install one or more of the ODBC drivers shipped with PowerBuilder. You can do this when you first install PowerBuilder, or later. Using an ODBC driver supplied by Powersoft ensures that the driver was tested and found to work with PowerBuilder.
- **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 Powersoft. In most cases, these drivers will work with PowerBuilder. However, Powersoft may not have tested the drivers to verify this.

Accessing ALLBASE/SQL, SQLBase, and XDB through ODBC
The vendors of ALLBASE/SQL, SQLBase, and XDB recommend using ODBC to access their DBMSs. To comply with this recommendation, PowerBuilder no longer provides native database interfaces for accessing ALLBASE/SQL, SQLBase, and XDB. Instead, you should use ODBC-compliant drivers obtained from other vendors to access these DBMSs.

Platform differences

The ability to use ODBC drivers supplied by Powersoft or from another vendor to access data in the development environment depends on the platform you are using, as follows:

Platform	Product includes ODBC driver(s)	Product allows use of other vendors' ODBC drivers
Windows	Yes (INTERSOLV and SQL Anywhere)	Yes
Power Macintosh	Yes (SQL Anywhere)	Yes
UNIX	Yes (INTERSOLV)	No

Using ODBC drivers with PowerBuilder Desktop

Using ODBC drivers that come with Desktop

If you are using PowerBuilder Desktop, you can access data using the ODBC drivers that are shipped with the product. Unlike PowerBuilder Enterprise and PowerBuilder Professional, you *cannot* use an ODBC driver obtained from another vendor with PowerBuilder Desktop.

Using existing Microsoft ODBC drivers

If you already have Version 2.0 or higher 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 INTERSOLV drivers is recommended
PowerBuilder Desktop on the Windows platform comes with INTERSOLV ODBC drivers for all of these data sources *except Access*. Since the INTERSOLV drivers have been tested to work with PowerBuilder Desktop, you should use the INTERSOLV 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 Configure ODBC dialog box	Click the Help button in the Configure ODBC dialog box
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 your INTERSOLV ODBC driver	**On Windows 95 or Windows NT** Click the Help button in the ODBC setup dialog box for your driver

To get help on	Do this
Using SQL Anywhere	See the *Sybase SQL Anywhere User's Guide* (available in the Powersoft Online Books)
Using an ODBC driver obtained from a vendor other than Powersoft	See the vendor's documentation for that driver
Troubleshooting your ODBC connection	Check for a FaxLine document that describes how to connect to your ODBC data source. Updated information about connectivity issues is available from the Powersoft FaxLine system and from Powersoft electronic services on CompuServe, FTP, BBS, and the World Wide Web

About preparing ODBC data sources

The first step in connecting to an ODBC data source is to prepare to use the data source. Preparing the data source ensures that you will be able to connect to it 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 general 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.

In general, table and column names should *avoid* using all lowercase characters, blank spaces, or database-specific reserved words. It is safest to use all uppercase characters when naming tables and columns that you want to access in PowerBuilder.

Backquote character not supported as delimiter

The online Help supplied for the INTERSOLV 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 support the backquote character as a delimiter for table and column names.

To ensure that you can access your tables and columns in PowerBuilder, avoid using nonstandard characters (such as all lowercase, mixed case, blank spaces, or database-specific reserved words) when naming tables and columns.

---

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 Powersoft ODBC interface (installed by default unless you clear this option)

FOR INFO For specific instructions, see "Preparing to use the data source" in the section for your data source driver later in this chapter.

About 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 needs 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 can start PowerBuilder and define the data source. To define a data source, you must complete the ODBC setup dialog box for the driver you are using to access it. For example, to define a Sybase SQL Anywhere data source, you must complete the SQL Anywhere ODBC Configuration dialog box (as described in the SQL Anywhere section "Defining the SQL Anywhere data source on Windows" on page 111.)

The contents of the ODBC setup dialog box varies somewhat for each driver, but most setup dialog boxes ask for the following information:

- Data source name and location
- Optional description
- Other DBMS-specific connection parameters

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.

On Windows and Macintosh You can use the Configure ODBC dialog box in PowerBuilder to access your driver's ODBC setup dialog box to define the data source.

On UNIX The Configure ODBC dialog box is unavailable in PowerBuilder for UNIX. Instead, you must define a data source by creating a database profile. (For instructions on defining an INFORMIX data source, see "INTERSOLV INFORMIX on UNIX" on page 72.)

When you define the data source

For some ODBC data sources, PowerBuilder automatically defines the data source and creates the database profile for you. For example, a SQL Anywhere database on the Windows and Power Macintosh platforms is considered an ODBC data source because you access it with the SQL Anywhere ODBC driver.

When you create a local SQL Anywhere database using PowerBuilder on your computer, PowerBuilder automatically defines the SQL Anywhere data source and creates the database profile.

However, *you* must define the ODBC data source yourself when you want to access either of the following:

- A SQL Anywhere database that you did not create using PowerBuilder on your computer
- An ODBC data source that you are accessing with an ODBC driver other than the SQL Anywhere ODBC driver

---

On Windows and Macintosh You *can* use ODBC-compliant drivers obtained from vendors other than Powersoft to access data in PowerBuilder.

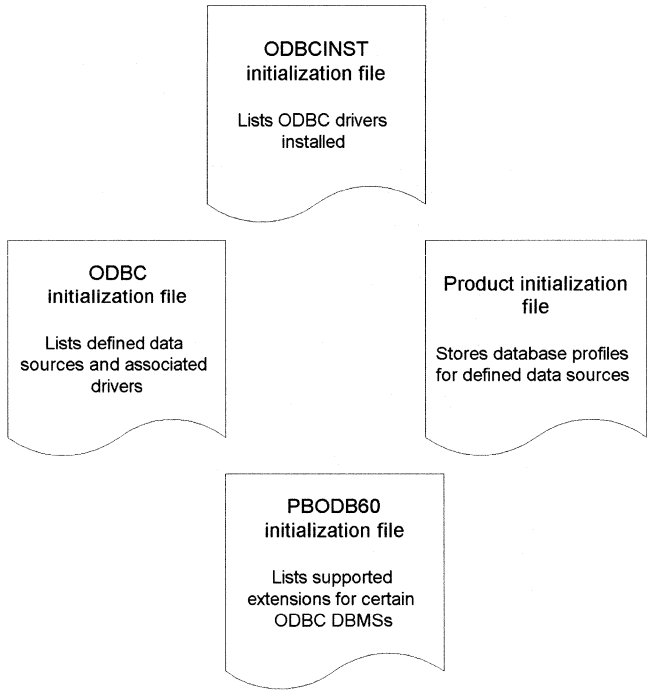
On UNIX You *cannot* use ODBC-compliant drivers obtained from vendors other than Powersoft to access data in PowerBuilder.

How Powersoft products access the data source

This section describes what happens in PowerBuilder when you access an ODBC data source.

Four initialization files

When you access an ODBC data source in PowerBuilder, there are four initialization files on your computer that work with the Powersoft ODBC interface and driver to make the connection. (These files may have different names and locations on different platforms, as described in "Names and locations on different platforms" next.)



Names and locations on different platforms

Depending on the product and platform you are using, these initialization files have different names and locations, as summarized in the following table:

Win 95 and Win NT	Macintosh	UNIX
Registry subkey in: HKEY_LOCAL_MACHINE SOFTWARE ODBC ODBCINST.INI	—	$HOME/ .odbcinst.ini

Win 95 and Win NT	Macintosh	UNIX
Registry subkey in: HKEY_CURRENT_USER SOFTWARE ODBC ODBC.INI	System Folder:Preferences: ODBC Preferences PPC *or* System Folder:Preferences: ODBC Preferences	$HOME/.odbc.ini
PWRS\SHARED\ PBODB60.INI *or* Program Files\Powersoft\ Shared\PBODB60.INI	System Folder:Preferences: Powersoft ODBC 6.0 Preferences	$PBHOME/bin/ pbodb60.ini
PB.INI in PowerBuilder product directory	System Folder: Preferences: Powersoft 6.0 Preferences: PowerBuilder Preferences	$HOME/.pb.ini

ODBCINST initialization file

Contents

When you install an ODBC-compliant driver supplied by Powersoft or another vendor, the ODBCINST initialization file is automatically updated with a description of the driver. This description includes:

- The DBMS or data source associated with the driver
- The drive and directory of the driver and setup DLLs (for some data sources, the driver and setup DLLs are the same)
- Other driver-specific connection parameters

Editing

You should *not* need to edit ODBCINST initialization file directly to modify connection information. If your driver uses the information in the ODBCINST initialization file, the file is automatically updated when you install the driver. This is true whether the driver is supplied by Powersoft or another vendor.

Example

The following portion of an ODBCINST.INI file on Windows shows an entry for the installed Sybase SQL Anywhere 5.0 driver:

```
[ODBC Drivers]
Sybase SQL Anywhere 5.0=Installed
[Sybase SQL Anywhere 5.0]
Driver=c:\SQLANY50\win32\WOD50T.DLL
Setup=c:\SQLANY50\win32\WOD50T.DLL
```

ODBC initialization file

Contents

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 initialization file.

The [ODBC Data Sources] section of the ODBC initialization file lists the name of each defined data source and its associated DBMS. The ODBC initialization file also includes a separate section for each data source. This section 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:

- Name
- Optional description
- DBMS-specific connection parameters

Editing

You should *not* edit the ODBC initialization file directly to modify connection information. Instead, you should use a tool designed to define ODBC data sources and the ODBC initialization file automatically.

In the PowerBuilder development environment on Windows and Macintosh, you can access the appropriate ODBC setup dialog box for your driver from the Configure ODBC dialog box. The driver writes the connection parameters you specify in the setup dialog box to the ODBC initialization file.

Outside PowerBuilder, you should use a tool such as the ODBC Administrator (on Windows) or ODBC Setup control panel (on Macintosh) to configure data sources and make the necessary changes to the ODBC initialization file.

Example

The following portion of an ODBC.INI file on Windows shows a sample definition for a Sybase SQL Anywhere 5.0 data source named Powersoft Demo DB V6:

```
[ODBC Data Sources]
Powersoft Demo DB V6=Sybase SQL Anywhere 5.0
[Powersoft Demo DB V6]
DatabaseFile=c:\program files\powersoft\pb6\demodb\
   psdemodb.db
DatabaseName=PSDEMODB
UID=dba
PWD=sql
Driver=c:\SQLANY50\win32\WOD50T.DLL
Start=c:\SQLANY50\win32\dbeng50.exe -d -c512
```

PBODB60 initialization file

Contents

On the Windows and Macintosh platforms, PowerBuilder uses the PBODB60 initialization file to maintain access to extended functionality in the backend DBMS for which ODBC does not provide an API call. Examples of extended functionality are SQL syntax or DBMS-specific function calls.

Editing

In most cases, you should not need to edit the PBODB60 initialization file. In certain situations, however, you may need to add functions to the PBODB60 initialization file for your backend DBMS.

FOR INFO For instructions, see Appendix B, "Adding Functions to the PBODB60 Initialization File".

Example

The following portion of the PBODB60 initialization file on Windows shows the entry for Sybase SQL Anywhere:

```
;*****************************************
;DBMS Driver/DBMS Settings see comments at end
;of file
;*****************************************
...
[Sybase SQL Anywhere]
PBSyntax='WATCOM50_SYNTAX'
PBDateTime='STANDARD_DATETIME'
PBFunctions='WATCOM_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)'
```

PowerBuilder initialization file

Contents

When you define an ODBC data source in PowerBuilder on the Windows or Macintosh platforms, a database profile is automatically created for that data source containing the connection parameters you specified in the ODBC setup dialog box. You can then select this database profile at any time to connect to the data source.

Database profiles for all data sources are stored in the PowerBuilder initialization file.

Editing

You should *not* need to edit the PowerBuilder initialization file 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 database profile box or complete the Database Preferences property sheet in PowerBuilder to specify other connection parameters stored in the initialization file. (For instructions, see Chapter 5, "Setting Additional Connection Parameters".)

Example

The following portion of the PB.INI file on Windows shows the database profile for the Powersoft Demo DB V6 data source:

```
[Profile Powersoft Demo DB V6]
DBMS=ODBC
Database=Powersoft Demo DB V6
UserId=dba
DatabasePassword=
LogPassword=
ServerName=
LogId=
Lock=
DbParm=ConnectString='DSN=Powersoft Demo DB V6;
   UID=dba;PWD=sql'
Prompt=0
```

Values used with ODBC data sources This initialization file example shows the two most important values in a database profile for an ODBC data source:

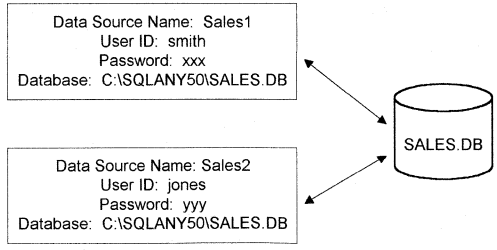
- **DBMS** The DBMS value (ODBC) indicates that you are using the Powersoft ODBC interface to connect to the data source.
- **DBParm** The ConnectString DBParm parameter controls your ODBC data source connection. The connect string *must* specify the DSN (data source name) value, which tells ODBC which data source you want to access. When you select a database profile to connect to a data source, ODBC looks in the ODBC initialization file for a section that corresponds to the data source name in your profile (in this case, Powersoft Demo DB V6). ODBC then uses the information in the specified section of the ODBC initialization file to load the programs required 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.

About defining multiple data sources for the same data

When defining 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 a SQL Anywhere database located in C:\SQLANY50\SALES.DB. Depending on your application, you may want to specify different sets of connection parameters for accessing the database—for example, using 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:\SQLANY50\SALES.DB) but different user IDs and passwords. Defining these data sources automatically creates corresponding database profiles named Sales1 and Sales2. When you connect to the data source using either of these profiles, you are using different connection parameters to access the same data.



Inheriting ODBC data sources from other users

In addition to defining a new ODBC data source in PowerBuilder, you can also access an existing ODBC data source that you inherit from another user. For example, to access a SQL Anywhere database created by someone else at your site, you must:

- Install the required data files for the data source
- Define the ODBC data source on your computer by completing the ODBC setup dialog box for the driver you are using

❖ **To access an inherited ODBC data source:**

1. Make sure the required data files are installed on your computer or on the network.

   For example, if you are accessing a SQL Anywhere data source, you need the database (DB) file and the accompanying transaction log (LOG) file. If you are accessing an Excel data source, you need the Excel workbook or worksheet (XLS) file.

2. Make sure you have prepared the data file for use with PowerBuilder.

   FOR INFO For instructions, see "Preparing to use the data source" in the section for your data source driver later in this chapter.

3. Make sure the ODBC driver required to access the data source is installed on your computer.

   FOR INFO For instructions on installing drivers supplied by Powersoft, see the *PowerBuilder Installation Guide*.

   FOR INFO For more about using an ODBC driver supplied by a vendor other than Powersoft, see "Obtaining ODBC drivers" on page 29.

4. Define the ODBC data source on your computer by completing the ODBC setup dialog box for the driver you are using.

   For example, to define a SQL Anywhere data source, you must complete the SQL Anywhere ODBC Configuration dialog box.

   FOR INFO For instructions, see the section for your data source driver later in this chapter.

What happens

When you complete the ODBC setup dialog box for your driver, PowerBuilder automatically does both of the following:

- Updates the ODBC initialization file on your computer with the information required to connect to the data source
- Creates a database profile so you can easily connect to this data source

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 (applicable on Windows and Macintosh platforms only)
- Using the ODBC driver to access the data source

Help for any ODBC driver on Windows and Macintosh

Use the following procedure to display vendor-supplied Help when you are in the ODBC setup dialog box for *any* ODBC driver supplied with PowerBuilder on the Windows and Macintosh platforms.

❖ **To display Help for any ODBC driver on Windows and Macintosh:**

1 Click the Help button in the ODBC setup dialog box for your driver.

A Help window displays describing the boxes 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.

Completing the ODBC setup dialog box

To define an ODBC data source, you must complete the ODBC setup dialog box for the driver you are using to access the data source, as described in the following procedure.

This procedure is the same in PowerBuilder on the Windows and Macintosh platforms.

On UNIX

The Configure ODBC dialog box is unavailable in PowerBuilder for UNIX. Instead, you must define a data source by creating a database profile.

FOR INFO For instructions on defining an INFORMIX data source, see "INTERSOLV INFORMIX on UNIX" on page 72.

Before you start

Before you define an ODBC data source, make sure you've installed the ODBC driver required to access the data source, and prepared the data source for use in PowerBuilder.

FOR INFO For more about ODBC drivers, see "Using the Powersoft ODBC interface" on page 19.

FOR INFO For instructions on preparing an ODBC data source, see "Preparing to use the data source" in the section for your data source driver.

❖ **To define an ODBC data source:**

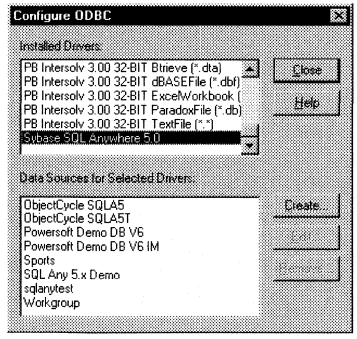
1 Click the Configure ODBC button in the PowerBar.
or
In the Database painter, select File>Configure ODBC from the menu bar.
or
In the Database Profiles dialog box, select the ODBC interface or an ODBC database profile. Click the Config ODBC button, or display the item's popup menu and select Config ODBC.

Configure ODBC button
If your PowerBar does not include the Configure ODBC button, use the customize feature to add the button to the PowerBar.

FOR INFO For information about customizing toolbars, see the *PowerBuilder User's Guide*.

The Configure ODBC dialog box displays, listing the following:

- The ODBC drivers installed on your computer. The names of the ODBC drivers come from the ODBCINST initialization file.
- The data sources defined for each selected driver. The names of the data sources come from the ODBC initialization file.



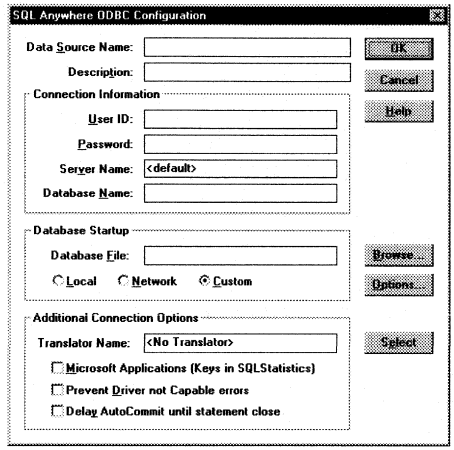
2 Select the ODBC driver for the data source you want to access.

If any data sources are already defined for that driver, their names display in the data source list.

3 Click the Create button to create a new ODBC data source definition.

The ODBC setup dialog box for the selected driver displays. The contents of this dialog box varies, depending on the driver you select and the PowerBuilder platform you are using.

For example, the following dialog box displays on Windows if you select the Sybase SQL Anywhere driver.



4 In the Data Source Name box, type a name for the data source.

The name you specify becomes the name of the database profile for this data source.

Parentheses prohibited in data source names

Due to a Microsoft restriction, parentheses are *not allowed* in ODBC data source names. If you include parentheses in the Data Source Name box, an error message displays indicating that the data source is invalid.

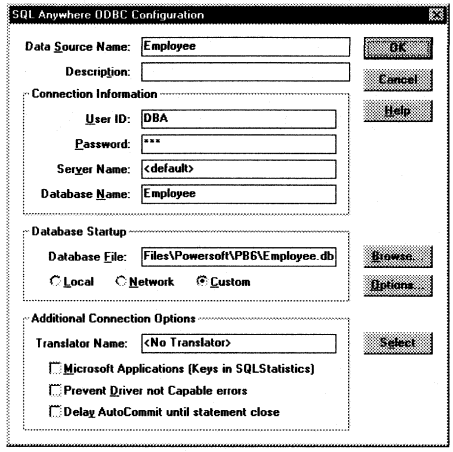
5 Specify the fully qualified database name (such as C:\SQLANY50\EMPLOYEE.DB), or the directory or folder where the database resides (such as C:\SQLANY50).

Do this in one of the following ways, depending on the contents of the ODBC setup dialog box you are completing:

- Type the name in the appropriate textbox. In most dialog boxes, this textbox is labeled Database Directory or Database File.

- If the ODBC setup dialog box contains a Browse or Select button, you can click this button to select the database name or directory from a list of available databases. The name you select displays in the appropriate box in the setup dialog box.

  For example, the following shows a completed SQL Anywhere ODBC Configuration dialog box on the Windows platform for a data source named Employee.



6 Supply values in the remaining boxes as appropriate for your data source connection.

Getting help

If the ODBC setup dialog box for your driver has a Help button, click it for information about completing the dialog box.

FOR INFO For general help on using your driver, see "Getting help with ODBC drivers" on page 30.

7 Click OK.

If PowerBuilder requires additional information to connect to the data source, such as the user ID and password, you will be prompted for it later when you select the database profile to connect to the data source.

FOR INFO For security considerations when supplying the user ID and password, see "Specifying the user ID and password" on page 47.

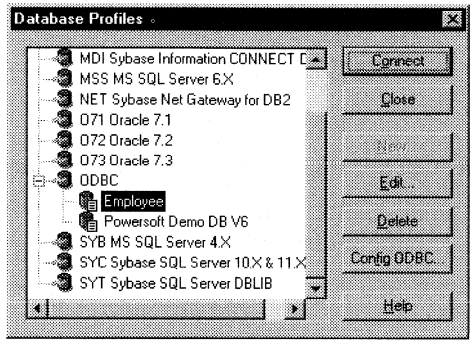
When you complete the ODBC setup dialog box, the Configure ODBC dialog box displays. The data source you defined (Employee) is listed with any other data sources previously defined for the selected driver.

8 Repeat steps 2 through 7 if you want to define another ODBC data source.

9 Click Close when you are finished defining the ODBC data sources you need.

PowerBuilder updates the ODBC initialization file with the data source definition, and creates a database profile for each data source you defined.

For example, after you define a SQL Anywhere data source named Employee, a database profile named Employee displays under the ODBC interface in the Database Profiles dialog box.



FOR INFO For instructions on using database profiles to connect to a data source, see Chapter 4, "Managing Database Connections".

Specifying the user ID and password

User ID

If your driver's ODBC setup dialog box requires a user ID and you omit it, PowerBuilder prompts you for the user ID when you connect to the data source, unless the UID (user ID) value is already in the ConnectString.

If the ODBC driver returns the UID value when you connect, PowerBuilder copies this value to the ConnectString DBParm in your database profile.

Password

A password specified in the ODBC setup dialog box is displayed in two places: the ODBC initialization file and the ConnectString DBParm in the database profile. Therefore, specifying a password in the setup dialog box may be a security risk.

Suppressing display in initialization file

To suppress password display in the ODBC initialization file and the database profile, you can instead specify the password in the dialog box that prompts you for additional connection information when you connect to the data source.

Selecting an ODBC translator

What is an ODBC translator?

The ODBC drivers supplied with PowerBuilder on the Windows platform 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.

On Macintosh Currently, you cannot select a translator in the Sybase SQL Anywhere ODBC Configuration dialog box on the Macintosh.

On UNIX Currently, you cannot specify a translator for the INTERSOLV INFORMIX ODBC driver in PowerBuilder on UNIX.

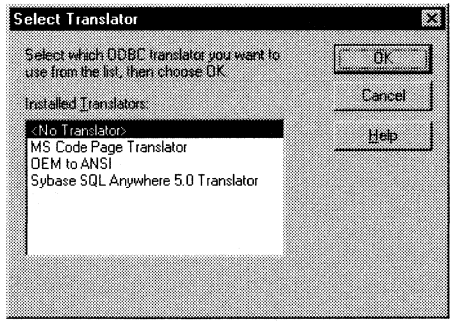
What you do

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

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

FOR INFO The way you display the Select Translator dialog box for Powersoft-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.

The Select Translator dialog box displays. The translators listed in this dialog box are determined by the values in your ODBCINST initialization file.



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.

What to do next

FOR INFO For step-by-step instructions on how to prepare and define your ODBC data source, go to the section for your data source driver.

INTERSOLV Btrieve

This section describes how to prepare and define a Btrieve data source in order to connect to it using the INTERSOLV Btrieve ODBC driver.

Supported versions and platforms for Btrieve

Versions

The INTERSOLV Btrieve ODBC driver supports connection to Btrieve Version 5.x and 6.x data sources.

Platforms

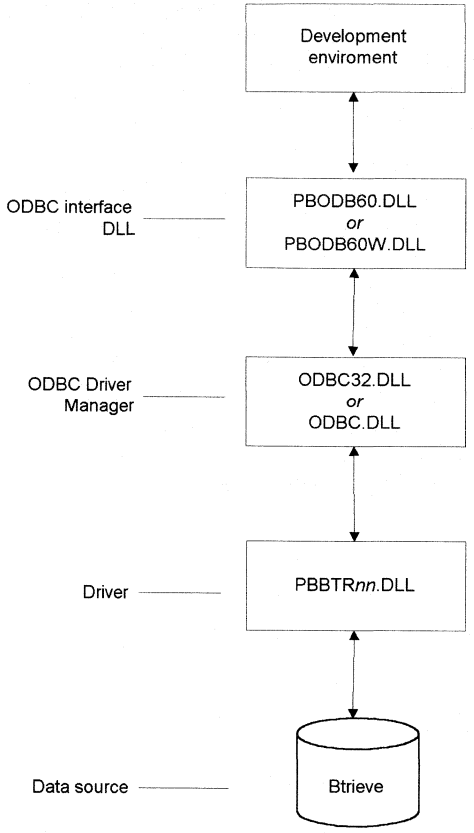
The INTERSOLV Btrieve ODBC driver is available on the following operating system platforms:

- Windows NT
- Windows 95
- Windows 3.x

FOR INFO For more information about supported drivers for your product, see Appendix A, "Supported ODBC Drivers and Powersoft Database Interfaces".

Basic software components for Btrieve

The following diagram shows the basic software components you need to connect to a Btrieve data source in PowerBuilder using the INTERSOLV Btrieve ODBC driver:



Preparing to use the Btrieve data source

Before you define and connect to a Btrieve data source in PowerBuilder, follow these steps to prepare the data source.

❖ **To prepare a Btrieve data source for use with the INTERSOLV Btrieve driver:**

1 In addition to the files shown in "Basic software components for Btrieve" on page 51, make sure the following files are in a directory on your program search path or in the Windows SYSTEM directory:

Btrieve version	Required files	Provider
Btrieve 5.x (single-user version installed locally)	WBTRCALL.DLL (non-network version) WDDLSVCS.DLL	Powersoft or INTERSOLV INTERSOLV
Btrieve 6.x (multiuser version installed on a network server)	WBTRCALL.DLL (network version) WBTRV32.DLL WDDLSVCS.DLL	Btrieve Technologies Btrieve Technologies INTERSOLV

On Windows 95 and Windows NT
When you are using the 32-bit INTERSOLV Btrieve ODBC driver with PowerBuilder on the Windows 95 or Windows NT platforms, *you must use* the multiuser (network) versions of WBTRCALL.DLL and WBTRV32.DLL. These files are supplied by Btrieve Technologies.

2 Make sure you have the necessary Scalable SQL data dictionary files in the same directory as the Btrieve data source you want to access.

Without the proper data dictionary files, you cannot access Btrieve data in PowerBuilder.

FOR INFO For instructions, see "Defining the structure of Btrieve tables" on page 53.

3 Check your WIN.INI file to make sure it contains a Btrieve section.

If you install the INTERSOLV Btrieve ODBC driver supplied with PowerBuilder, the setup program should add the correct Btrieve section in WIN.INI to enable access to Btrieve data sources.

Defining the Btrieve data source

❖ **To define a Btrieve data source for the INTERSOLV Btrieve driver:**

1 Select the INTERSOLV Btrieve driver in the Configure ODBC dialog box.

2 Click the Create button.

The ODBC Btrieve Driver Setup dialog box displays.

3 Click the Help button for information about how to specify values in the Setup dialog box.

4 (Optional) If the Setup dialog box contains an Advanced tab or button, click it to configure additional connection options for the data source.

Click Help for information about how to specify optional settings for the Btrieve driver.

5 (Optional) To define the table structure, click the Define tab in the Setup dialog box or the Define button in the Advanced Driver Setup dialog box.

FOR INFO See "Defining the structure of Btrieve tables" next.

6 (Optional) To select an ODBC translator to translate your data from one character set to another, click the Translate button on the Advanced tab or in the Advanced Driver Setup dialog box.

FOR INFO See "Selecting an ODBC translator" on page 48.

7 Click OK to save the data source definition.

Defining the structure of Btrieve tables

Unlike some other databases, Btrieve does not store column information with the data file itself. Therefore, in order to access an existing Btrieve table in PowerBuilder, you must have the necessary Scalable SQL data dictionary files in the same directory as the tables you want to access.

Scalable SQL data dictionary

The INTERSOLV Btrieve ODBC driver uses the Scalable SQL data dictionary to obtain information about the structure of your Btrieve tables. This includes information about table names, column names, data types, precision, scale, nulls, and indexes.

The data dictionary defines the structure of the Btrieve table, and consists of the following files:

File	Contains information about
FILE.DDF	Each table in the database
FIELD.DDF	Each column in each table
INDEX.DDF	Each index

Using CDB=1 in the ConnectString

If a complete data dictionary does not exist in your Btrieve database directory, you can set a DBParm value in the database profile for this data source that tells the INTERSOLV Btrieve driver to create a new data dictionary if one does not already exist.

PowerBuilder creates a database profile automatically when you define the data source.

❖ **To set the ConnectString DBParm to create a new data dictionary:**

- Type the following in the Driver-Specific Parameters box in the Database Profile Setup - ODBC dialog box:

  CDB=1

FOR INFO For more information, see ConnectString on page 414.

Creating DDF files

The new data dictionary files will contain information about any *new* tables you create. They will not, however, contain information about existing tables you may want to access. To access an existing Btrieve table, you must have the DDF files created for that table.

If you do not have the DDF files for an existing Btrieve table, you can use the INTERSOLV Btrieve driver to create the DDF files. Keep in mind, however, that you must know the exact layout and internal structure of the table in order to create DDF files for it. Creating DDF files does not change your Btrieve file in any way. You can continue to access the file directly through any existing Btrieve application.

The procedure for defining the structure of Btrieve tables depends on the version of the INTERSOLV Btrieve driver you are using.

FOR INFO For instructions, click Help in your driver's Setup dialog box.

What to do next

FOR INFO For instructions on connecting to the data source, see "Connecting to a database" on page 295.

INTERSOLV dBASE

This section describes how to prepare and define a data source in order to connect to it using the INTERSOLV dBASE ODBC driver.

Supported versions and platforms for dBASE

Versions

The INTERSOLV dBASE ODBC driver supports connection to the following data sources:

- Clipper files
- dBASE II, III, IV, and V files
- FoxBASE tables and indexes
- FoxPro files 1.x, 2.x, and 3.0 tables and indexes

FoxPro 3.0 DBC data sources not supported

The online Help for the INTERSOLV dBASE ODBC driver indicates that you can access FoxPro 3.0 database container (DBC) data sources.

However, PowerBuilder does not provide the INTERSOLV FoxPro DBC driver required to access FoxPro 3.0 DBC data sources.

Platforms

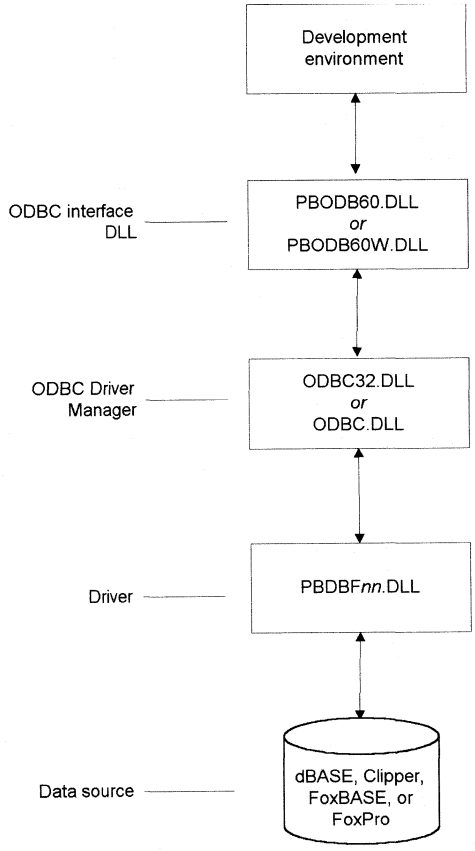
The INTERSOLV dBASE ODBC driver is available on the following operating system platforms:

- Windows NT
- Windows 95
- Windows 3.x

FOR INFO For more information about supported drivers for your product, see Appendix A, "Supported ODBC Drivers and Powersoft Database Interfaces".

Basic software components for dBASE

The following diagram shows the basic software components you need to connect to a dBASE data source in PowerBuilder using the INTERSOLV dBASE ODBC driver:



Preparing to use the dBASE data source

Before you define and connect to a dBASE data source in PowerBuilder, follow these steps to prepare the data source.

dBASE product not required
The INTERSOLV dBASE driver executes SQL statements directly on dBASE and dBASE-compatible files. Therefore, you do *not* need the dBASE product installed to access these files with the INTERSOLV dBASE driver.

❖ **To prepare a dBASE data source for use with the INTERSOLV dBASE driver:**

- In order to update a dBASE or Clipper table, PowerBuilder requires that the table have a unique index associated with it. Therefore, make sure a unique index is associated with the data source, as follows:
  - **dBASE II, dBASE III, and Clipper** Indexes for dBASE II and III files have an NDX extension; indexes for Clipper files have an NTX extension. If a dBASE II, III, or Clipper file does not have a unique index associated with it, click the Define button in the Setup dialog box or the Advanced Driver Setup dialog box to assign one. (For information, see "Associating dBASE and Clipper files with indexes" on page 58.) You must do this to enable the INTERSOLV driver to use and maintain the index.
  - **dBASE IV, dBASE V, and FoxPro** Indexes for dBASE IV files have an MDX extension; indexes for FoxPro files have a CDX extension. The INTERSOLV driver automatically associates dBASE IV and V files with the proper index. However, you cannot mark indexes with an MDX or CDX extension as unique. Instead, you can use the procedure for assigning an index to mark a tag within the index as unique. (For information, see "Associating dBASE and Clipper files with indexes" on page 58.)

Defining the dBASE data source

❖ **To define a dBASE data source for the INTERSOLV dBASE driver:**

1. Select the INTERSOLV dBASE driver in the Configure ODBC dialog box.
2. Click the Create button.

   The ODBC dBASE Driver Setup dialog box displays.
3. Click the Help button for information about how to specify values in the Setup dialog box.

4 (Optional) If the Setup dialog box contains an Advanced tab or button, click it to configure additional connection options for the data source.

Click Help for information about how to specify optional settings for the dBASE driver.

5 (Optional) To associate dBASE or Clipper files with indexes, click the Define tab in the Setup dialog box or the Define button in the Advanced Driver Setup dialog box.

FOR INFO See "Associating dBASE and Clipper files with indexes" next.

6 (Optional) To select an ODBC translator to translate your data from one character set to another, click the Translate button on the Advanced tab or in the Advanced Driver Setup dialog box.

FOR INFO See "Selecting an ODBC translator" on page 48.

7 Click OK to save the data source definition.

Associating dBASE and Clipper files with indexes

In order to update a dBASE or Clipper table, PowerBuilder requires that the table have a unique index associated with it, as follows:

- **dBASE II, dBASE III, and Clipper** The INTERSOLV dBASE driver does not automatically associate an index with dBASE II, dBASE III, or Clipper tables. Therefore, to access one of these tables that does not have a unique index, you *must* associate an index with it.
- **dBASE IV** The INTERSOLV dBASE driver automatically associates dBASE IV tables (which use an MDX index format) and dBASE V tables with the proper index. However, if the index you are using has an MDX or CDX (FoxPro) extension, you cannot mark it as unique. Therefore, although the following procedure for associating an index is *not required for dBASE IV and V tables*, you can use it as a way to mark tags within the index as unique.

The procedure for associating dBASE or Clipper files with indexes depends on the version of the INTERSOLV dBASE driver you are using.

FOR INFO For instructions, click Help in your driver's setup dialog box.

What to do next

FOR INFO For instructions on connecting to the data source, see "Connecting to a database" on page 295.

INTERSOLV Excel 4

This section describes how to prepare and define an Excel data source in order to connect to it using the INTERSOLV Excel 4 ODBC driver.

Supported versions and platforms for Excel 4

Versions

The INTERSOLV Excel 4 ODBC driver supports connection to Excel 2, 3, and 4 XLS files.

Excel 5 Workbook files

To access Excel 5 Workbook files in PowerBuilder, you must use the INTERSOLV Excel Workbook driver.

FOR INFO For instructions, see "INTERSOLV Excel Workbook" on page 65.

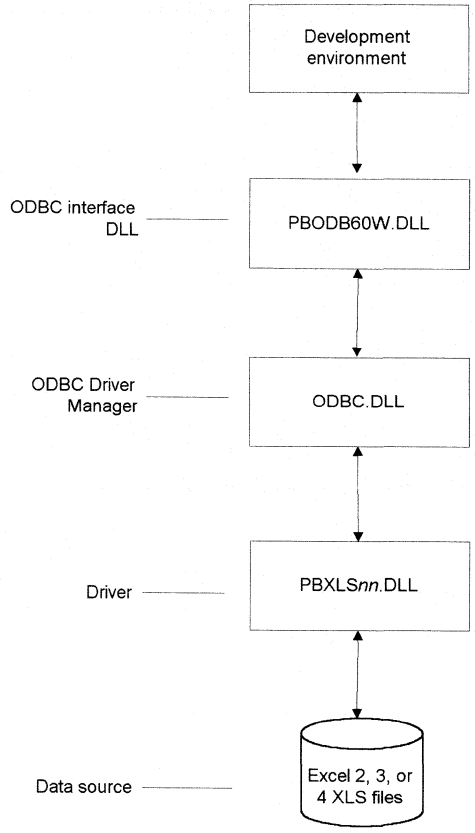
Platforms

The INTERSOLV Excel 4 ODBC driver is available on the Windows 3.x operating system platform only.

FOR INFO for more information about supported drivers for your product, see Appendix A, "Supported ODBC Drivers and Powersoft Database Interfaces".

Basic software components for Excel 4

The following diagram shows the basic software components you need to connect to an Excel data source in PowerBuilder using the INTERSOLV Excel 4 ODBC driver:



Preparing to use the Excel 4 data source

Excel database ranges

In order for PowerBuilder to access data from an Excel 2, 3, or 4 worksheet, you must first define a database range for the worksheet. In an Excel worksheet, a **database range** is a rectangular range of cells defined as a database. Only those Excel worksheets with a defined database range will display in the list of tables for the data source in PowerBuilder.

If your Excel 2, 3, or 4 worksheet already contains a database range, then you do *not* need the Excel product installed to access the data source in PowerBuilder.

What you do

If the Excel worksheet you want to access does not contain a database range, then you *must* follow these steps to prepare the data source before you define and connect to it in PowerBuilder.

❖ **To prepare an Excel data source for use with the INTERSOLV Excel 4 driver:**

1 Start Excel on your computer.

2 Open the Excel worksheet you want to use as a data source.

3 Select the range of cells you want to include in the database range for this worksheet.

The first row of cells you select contains the column (field) names. Subsequent rows you select contain data for the columns.

If any empty cells appear in the database range in place of column names, these cells will display as columns when you open the worksheet as a table in PowerBuilder. However, they will have names such as Field1 or Field2 assigned instead of actual column names.

4 Select Data>Set Database from the Excel menu bar to define the current worksheet selection as a database range.

FOR INFO For instructions on using the Set Database command, see your Excel documentation.

5 Save your changes to the Excel worksheet.

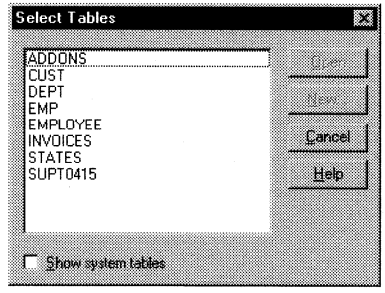
Example

In the following Excel worksheet, the database range is highlighted area including cells A1 through D10. The cells in this range will display when you open this worksheet as a table in PowerBuilder.

	A	B	C	D	E
1	state_id	state_name	state_capital	country	
2	AB	Alberta	Edmonton	CAN	
3	BC	British Columbia	Victoria	CAN	
4	MB	Manitoba	Winnipeg	CAN	
5	NB	New Brunswick	Fredericton	CAN	
6	NF	Newfoundland	St. John's	CAN	
7	NT	Northwest Territories	Yellowknife	CAN	
8	NS	Nova Scotia	Halifax	CAN	
9	ON	Ontario	Toronto	CAN	
10	PE	Prince Edward Island	Charlottetown	CAN	
11	PQ	Québec	Québec	CAN	
12	SK	Saskatchewan	Regina	CAN	
13	YT	Yukon Territory	Whitehorse	CAN	
14	AL	Alabama	Montgomery	USA	
15	AK	Alaska	Juneau	USA	
16	AZ	Arizona	Phoenix	USA	
17	AR	Arkansas	Little Rock	USA	
18	CA	California	Sacramento	USA	
19	CO	Colorado	Denver	USA	
20	CT	Connecticut	Hartford	USA	

When you connect to the Excel data source containing this worksheet in PowerBuilder:

- The name of the worksheet (without the XLS extension) displays on the table list for this data source. For example, the Excel file STATES.XLS is a table named STATES in the Select Tables dialog box.



- The rows and columns you included in the database range for this worksheet display when you view the data in the table.

Defining the Excel 4 data source

❖ **To define an Excel data source for the INTERSOLV Excel 4 driver:**

1 Select the INTERSOLV Excel 4 (ExcelFile) driver in the Configure ODBC dialog box.

2 Click the Create button.

The ODBC Excel Driver Setup dialog box displays.

3 Click the Help button for information about how to specify values in the setup dialog box.

4 (Optional) If the Setup dialog box contains an Advanced tab or button, click it to configure additional connection options for the data source.

Click Help for information about how to specify optional settings for the Excel 4 driver.

5 (Optional) To select an ODBC translator to translate your data from one character set to another, click the Translate button on the Advanced tab or in the Advanced Driver Setup dialog box.

FOR INFO See "Selecting an ODBC translator" on page 48.

6 Click OK to save the data source definition.

What to do next

FOR INFO For instructions on connecting to the data source, see "Connecting to a database" on page 295.

INTERSOLV Excel Workbook

This section describes how to prepare and define an Excel data source in order to connect to it using the INTERSOLV Excel Workbook ODBC driver (formerly called the Excel 5 driver).

Supported versions and platforms for Excel Workbook

Versions

The INTERSOLV Excel Workbook ODBC driver supports connection to Excel 5 XLS Workbook files.

Excel 2, 3, and 4 files

To access Excel Version 2, 3, or 4 files, you must use the INTERSOLV Excel 4 ODBC driver.

FOR INFO For instructions, see "INTERSOLV Excel 4" on page 60.

Platforms

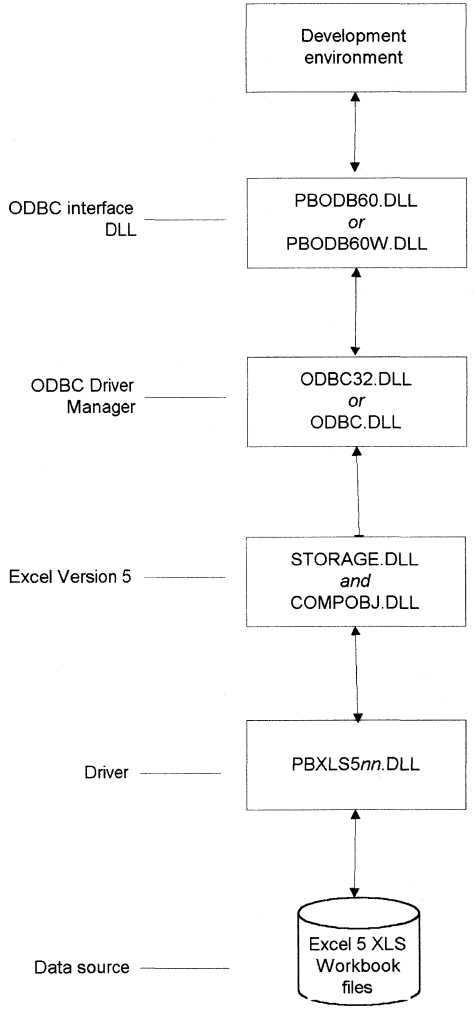
The INTERSOLV Excel Workbook ODBC driver is available on the following operating system platforms:

- Windows NT
- Windows 95
- Windows 3.x

FOR INFO For more information about supported drivers for your product, see Appendix A, "Supported ODBC Drivers and Powersoft Database Interfaces".

Basic software components for Excel Workbook

The following diagram shows the basic software components you need to connect to an Excel data source in PowerBuilder using the INTERSOLV Excel Workbook ODBC driver:



Preparing to use the Excel Workbook data source

Excel Workbook databases

In order for PowerBuilder to access an Excel Workbook data source, the Excel file must be a database. An **Excel Workbook database** is an Excel 5 XLS workbook file that contains one or more named lists. An **Excel Workbook list** is a labeled series of worksheet rows that contain similar information.

When you use an Excel Workbook as a database, each list is analogous to a database table, the list rows correspond to database records, and the list columns correspond to database fields. When you connect to an Excel Workbook database in PowerBuilder, the list names display in the Select Tables list in the Database painter.

What you do

Before you define and connect to an Excel Workbook data source in PowerBuilder, follow these steps to prepare the data source.

❖ **To prepare an Excel data source for use with the INTERSOLV Excel Workbook driver:**

1 Make sure the Excel product is installed on your computer. You need the Excel product to access an Excel Workbook data source because:

- The INTERSOLV Excel Workbook driver requires two DLLs (COMPOBJ.DLL and STORAGE.DLL) to access an Excel data source. These DLLs are installed with the Excel product.
- The INTERSOLV Excel Workbook driver can only read data from (not write data to) Excel 5 XLS files. Therefore, you need the Excel product if you want to manipulate your Excel databases.

2 Make sure your Excel Workbook file contains one or more named lists. To ensure that you can access the lists as tables in PowerBuilder, observe the following guidelines when naming the lists:

- Give the list a workbook-level name, not a worksheet-level name.
- Make sure the list name follows standard SQL naming conventions, as described in "About preparing ODBC data sources" on page 32. It is best to use all uppercase characters when naming the list.

- The INTERSOLV Excel Workbook driver recognizes one list named *Database* per XLS file (in addition to other named lists in that file). However, you should consider giving your lists more meaningful and unique names to prevent naming conflicts with lists in other workbook databases.

FOR INFO For instructions on defining lists in an Excel Workbook and using them as databases, see your Excel documentation.

FOR INFO For illustrations showing how to set up and access an Excel Workbook file in PowerBuilder, see the section "Example" next.

3 Make sure the first row in each Excel Workbook list contains column names that identify the fields in each record.

Column names in an Excel Workbook file, like lists, must follow standard SQL naming conventions, as described in "About preparing ODBC data sources" on page 32.

Example

Setting up the database in Excel Assume that you have created an Excel Workbook file named Q2STOCK.XLS that you want to use as an ODBC data source in PowerBuilder.

Microsoft Excel - Q2STOCK.XLS

File Edit View Insert Format Tools Data Window Help

A1 ITEMNO

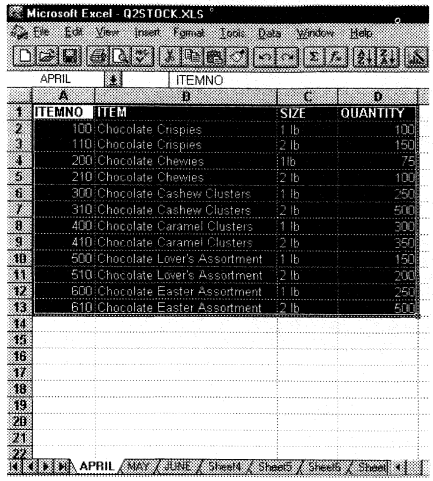
	A	B	C	D
1	ITEMNO	ITEM	SIZE	QUANTITY
2	100	Chocolate Crispies	1 lb	100
3	110	Chocolate Crispies	2 lb	150
4	200	Chocolate Chewies	1lb	75
5	210	Chocolate Chewies	2 lb	100
6	300	Chocolate Cashew Clusters	1 lb	250
7	310	Chocolate Cashew Clusters	2 lb	500
8	400	Chocolate Caramel Clusters	1 lb	300
9	410	Chocolate Caramel Clusters	2 lb	350
10	500	Chocolate Lover's Assortment	1 lb	150
11	510	Chocolate Lover's Assortment	2 lb	200
12	600	Chocolate Easter Assortment	1 lb	250
13	610	Chocolate Easter Assortment	2 lb	500
14				
15				
16				
17				
18				
19				
20				
21				
22				

APRIL MAY JUNE Sheet4 Sheet5 Sheet6 Sheet

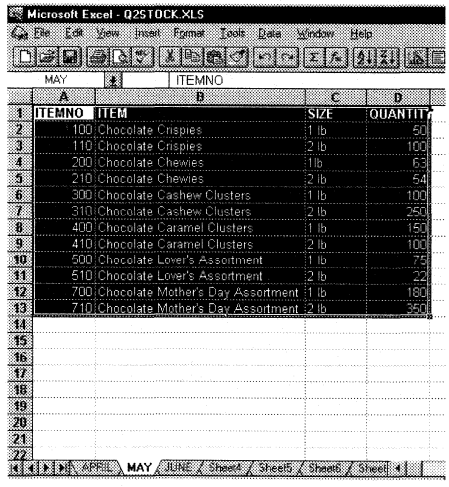
You must define named lists for the workbook file in Excel so you can access the data in PowerBuilder. Each Excel 5 list will display as a table in PowerBuilder.

Q2STOCK.XLS contains three named lists, one for each worksheet in the workbook. The list and column names follow standard SQL naming conventions to ensure that you can access the data in PowerBuilder.

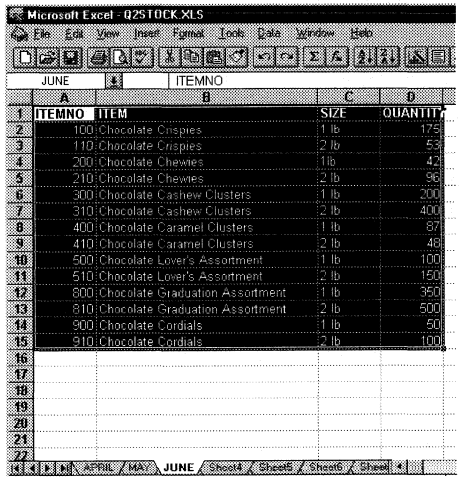
The APRIL list consists of all rows and columns in the APRIL worksheet.



The MAY list consists of all rows and columns in the MAY worksheet.

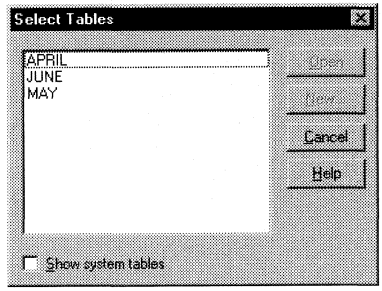


The JUNE list consists of all rows and columns in the JUNE worksheet.



Accessing the database in PowerBuilder When you connect to the Q2STOCK.XLS database in PowerBuilder:

- The APRIL, MAY, and JUNE lists you defined in Excel display as tables in the Select Tables dialog box in the Database painter.



- The rows and columns you included in each list display when you view the data in a table.

Defining the Excel 5 data source

❖ **To define an Excel data source for the INTERSOLV Excel 5 driver:**

1 Select the INTERSOLV Excel 5 (Excel5Workbook) driver in the Configure ODBC dialog box.

2 Click the Create button.

The ODBC Excel 5 Driver Setup dialog box displays.

3 Click the Help button for information about how to specify values in the setup dialog box.

4 (Optional) If the Setup dialog box contains an Advanced tab or button, click it to configure additional connection options for the data source.

Click Help for information about how to specify optional settings for the Excel Workbook driver.

5 (Optional) To select an ODBC translator to translate your data from one character set to another, click the Translate button on the Advanced tab or in the Advanced Driver Setup dialog box.

FOR INFO See "Selecting an ODBC translator" on page 48.

6 Click OK to save the data source definition.

What to do next

FOR INFO For instructions on connecting to the data source, see "Connecting to a database" on page 295.

INTERSOLV INFORMIX on UNIX

This section describes how to prepare and define an INFORMIX data source to connect to it in PowerBuilder for UNIX using the INTERSOLV INFORMIX ODBC driver.

On UNIX
On UNIX, the only ODBC connections supported in PowerBuilder are those using the INTERSOLV ODBC drivers included with the product. You *cannot* use ODBC drivers obtained from other vendors to access data in PowerBuilder for UNIX.

Supported versions and platforms for INFORMIX

Versions

The INTERSOLV INFORMIX ODBC driver supports connection to the following INFORMIX database servers:

- INFORMIX-SE Versions 5.x, 6.x, and 7.x
- INFORMIX-OnLine Version 5.x, 6.x, and 7.x

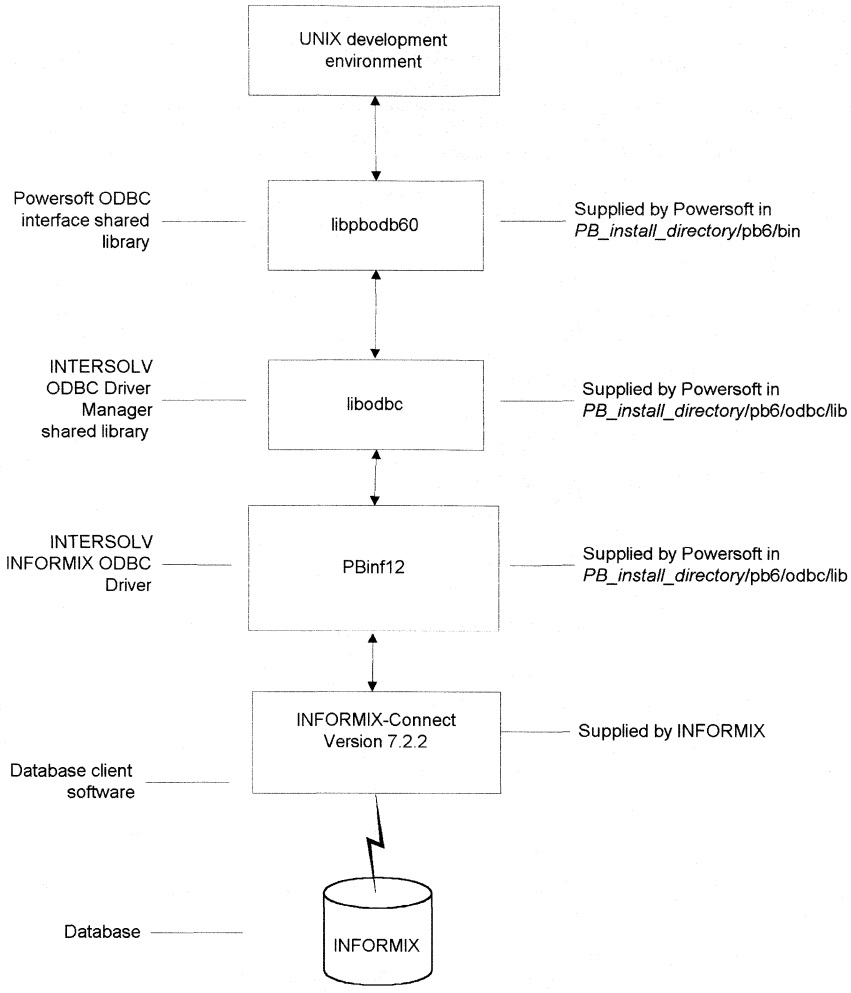
Platforms

The INTERSOLV INFORMIX ODBC driver is available on the Solaris, HP-UX, and AIX UNIX platforms.

FOR INFO For more information about supported drivers for your product, see Appendix A, "Supported ODBC Drivers and Powersoft Database Interfaces".

Basic software components for INFORMIX

The following diagram shows the basic software components you need to connect to an INFORMIX data source in PowerBuilder for UNIX using the INTERSOLV INFORMIX ODBC driver.



Preparing to use the INFORMIX data source

Before you define and connect to an INFORMIX data source in PowerBuilder for UNIX, follow these steps to prepare to use the data source.

Overview of basic steps for INFORMIX ODBC connection

Preparing an INFORMIX data source for use with PowerBuilder for UNIX involves the following basic steps.

❖ **To prepare an INFORMIX data source for use with the INTERSOLV INFORMIX ODBC driver:**

1 Install the Powersoft ODBC interface.

2 Install and configure the required database server and network software.

3 Define required environment variables and verify shared libraries.

4 Install and configure the required INFORMIX client software.

5 Verify that you can access the INFORMIX database outside PowerBuilder for UNIX.

Step 1: install the Powersoft ODBC interface

❖ **To install the Powersoft ODBC interface with PowerBuilder for UNIX:**

♦ Select the Powersoft ODBC interface when prompted to do so by the PowerBuilder for UNIX installation program.

The INTERSOLV INFORMIX ODBC driver and required ODBC shared libraries and initialization files are installed when you select the Powersoft ODBC interface.

FOR INFO For instructions, see the *PowerBuilder Installation Guide (UNIX).*

Step 2: install and configure database server and network software for INFORMIX

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

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

You must obtain the database server software from INFORMIX.

FOR INFO For installation instructions, see your INFORMIX documentation.

2 Make sure the required TCP/IP network software is running on your workstation and is properly configured so you can connect to the database server at your site.

FOR INFO For TCP/IP installation and configuration instructions, see your network administrator.

Step 3: define environment variables and verify shared libraries for INFORMIX

❖ **To define environment variables and verify shared libraries:**

1 Make sure the PBHOME environment variable is defined in your shell initialization file.

The PBHOME environment variable points to the directory where PowerBuilder for UNIX is installed (for example, /export/home/pb6).

Here is a sample PBHOME definition for a C shell initialization file:

```
setenv PBHOME /export/home/pb6
```

2 Define the ODBCDIR environment variable in your shell initialization file.

The ODBCDIR environment variable points to the PowerBuilder for UNIX ODBC directory that includes the INTERSOLV INFORMIX driver and related files. The location is *PowerBuilder_install_directory*/pb6/odbc.

For example, here is the ODBCDIR definition for a C shell initialization file:

```
setenv ODBCDIR ${PBHOME}/odbc
```

3 Make sure the following objects and shared libraries are in your $ODBCDIR/lib directory. (The file extension depends on your UNIX platform.)

libodbc
libodbcinst
libPBbas12
libPBflt12
PBinf12
libPBult12

4 Append $ODBCDIR/lib to the library path environment variable in your shell initialization file.

For example, on Solaris, here is the LD_LIBRARY_PATH definition for a C shell initialization file:

```
setenv LD_LIBRARY_PATH ${ODBCDIR}/lib:
   ${LD_LIBRARY_PATH}
```

5 Use the **which ldd** command to make sure ldd is in your command path.

Typically, the ldd command is in the /usr/bin directory.

6 Issue an ldd command on libpbodb60 (the Powersoft ODBC interface shared library) to make sure it has links to the appropriate database shared libraries. For example, on Solaris, type:

```
ldd $PBHOME/bin/libpbodb60.so
```

If your $PBHOME directory is /export/home/pb6, you should see output similar to the following:

```
libodbc.so          => /export/home/pb6/odbc/lib/libodbc.so
libC.so.5           => /export/home/pb6/windu/lib.sol2/libC.so.5
libPBult12.so       => /export/home/pb6/odbc/lib/libPBult12.so
libPBbas12.so       => /export/home/pb6/odbc/lib/libPBbas12.so
libc.so.1           => /usr/lib/libc.so.1
libintl.so.1        => /usr/lib/libintl.so.1
libdl.so.1          => /usr/lib/libdl.so.1
libw.so.1           => /usr/lib/libw.so.1
libodbcinst.so.1    => /export/home/pb6/odbc/lib/libodbcinst.so.1
```

7 Copy the .odbcinst.ini and .odbc.ini files from the $ODBCDIR directory (*PowerBuilder_install_directory*/pb6/odbc) to each user's $HOME directory. For example:

```
cp $ODBCDIR/.odbcinst.ini $HOME
cp $ODBCDIR/.odbc.ini $HOME
```

Step 4: install and configure the required INFORMIX client software

❖ **To install and configure the INFORMIX client software:**

1 Install the required INFORMIX client software on each client machine on which PowerBuilder for UNIX is installed.

To use the INFORMIX ODBC driver in PowerBuilder on UNIX, you must install INFORMIX-Connect Version 7.2.2 or higher.

You must obtain the INFORMIX-Connect client software from INFORMIX.

FOR INFO For installation instructions, see your INFORMIX documentation.

2 Define the required environment variables in your shell initialization file.

FOR INFO For complete information about these environment variables, see your INFORMIX documentation.

Environment variable	Description
INFORMIXDIR	The directory where the INFORMIX client software is installed
INFORMIXSERVER	The name of the INFORMIX database server as defined in the DBSERVERNAME setting in one of the following files on your server machine: ♦ **For Versions 5.x servers** Use the DBSERVERNAME setting in $INFORMIXDIR/etc/tbconfig file ♦ **For Version 6.x and 7.x servers** Use the DBSERVERNAME setting in $INFORMIXDIR/etc/onconfig file

Here are sample INFORMIXDIR and INFORMIXSERVER definitions for a C shell initialization file:

```
setenv INFORMIXDIR /db/informix
setenv INFORMIXSERVER informix_db
```

FOR INFO For information about accessing an INFORMIX database on a *different* server than the one defined as $INFORMIXSERVER, see "Accessing a different INFORMIX database server" on page 83.

3 Append $INFORMIXDIR/lib to the library path environment variable in your shell initialization file.

For example, on Solaris, here is the LD_LIBRARY_PATH definition for a C shell initialization file:

```
setenv LD_LIBRARY_PATH ${INFORMIXDIR}/lib:
${LD_LIBRARY_PATH}
```

4 Make sure the etc/sqlhosts file on your INFORMIX database server is properly configured for your environment.

In the following sample etc/sqlhosts entry, *informix_db* is the database server name, *ontlitcp* is the network protocol, *informix* is the name of the host machine running the database server, and *infx701* is the services entry name.

```
informix_db     ontlitcp     informix     infx701
```

FOR INFO For more information about setting up the etc/sqlhosts file, see your INFORMIX documentation.

5 Copy the etc/sqlhosts file from your INFORMIX database server to the $INFORMIXDIR/etc directory on the client machine.

For example, if your INFORMIX database server is located on a machine named *informix*, type:

```
cp /net/informix/etc/sqlhosts
   $INFORMIXDIR/etc/sqlhosts
```

6 Add an INFORMIX listener port in the /etc/services file on the machine from which you are running PowerBuilder for UNIX.

The /etc/services entry consists of a service name (*infx701*) and a listener port (*1525/tcp*). For example:

```
infx701     1525/tcp     #INFORMIX listener port
```

The service name *must match* the services entry name in the $INFORMIXDIR/etc/sqlhosts file and the entry in the /etc/services file on the database server. The listener port *must match* the entry in the /etc/services file on the database server.

7 Make sure the host name of your client machine is in the /etc/hosts file on the database server.

8 Make sure the .rhosts file or /etc/hosts.equiv file are properly set up so that your client machine is "trusted" by the INFORMIX database server.

FOR INFO See the .rhosts or hosts.equiv man page on your UNIX system.

Step 5: verify the INFORMIX connection outside PowerBuilder for UNIX

You can use the INFORMIX-ISQL tool to make sure that you can access the INFORMIX database outside PowerBuilder for UNIX. You must obtain INFORMIX-ISQL from INFORMIX.

INFORMIX-ISQL version
Make sure the version of INFORMIX-ISQL you are using is earlier than or the same as the version of INFORMIX on the database server you are accessing.

❖ **To verify the INFORMIX connection using INFORMIX-ISQL:**

1 Log in to your system as the Informix user.

2 Make sure the following environment variables are defined in your shell initialization file:

- INFORMIXDIR
- INFORMIXSERVER
- Library path environment variable (such as LD_LIBRARY_PATH on Solaris)
- PATH

Here are sample environment variable definitions for a C shell initialization file on Solaris:

```
setenv INFORMIXDIR /db/informix
setenv INFORMIXSERVER informix_db
setenv LD_LIBRARY_PATH ${INFORMIXDIR}/lib:
   ${LD_LIBRARY_PATH}
setenv PATH ${PATH}:${INFORMIXDIR}/bin
```

3 Type the following command as the INFORMIX user to start INFORMIX-ISQL:

```
isql
```

4 Verify that INFORMIX-ISQL displays list of the databases on your INFORMIX server.

This confirms that you can access the database through the INFORMIX-Connect client software.

5 Exit the INFORMIX-ISQL session.

Defining the INFORMIX data source on UNIX

Unlike the Windows and Macintosh platforms, there is no ODBC Administrator on the UNIX platform. Therefore, the Configure ODBC dialog box, which is the graphical interface to the ODBC Administrator, is unavailable in PowerBuilder for UNIX.

Since the Configure ODBC dialog box is unavailable, you must define an INFORMIX ODBC data source in PowerBuilder for UNIX by creating a database profile for it.

❖ **To create a database profile for an INFORMIX data source:**

1 Click the Database Profile button in the PowerBar.
or
In the Database painter, select File>Connect>Setup from the menu bar.

The Database Profiles dialog box displays, listing your installed Powersoft database interfaces and database profiles.

2 Select the ODBC interface and click New.
or
Display the popup menu for the ODBC interface and select New.

The Database Profile Setup-ODBC dialog box displays with the Connection tab visible.

3 Type a meaningful name for your profile in the Profile Name box.

4 Select the **pbinf** data source from the Data Source dropdown listbox.

You *must* select pbinf because it is the name of the INFORMIX data source defined in the .odbc.ini file in your $HOME directory.

5 Type the user name required to connect to the INFORMIX database server in the User ID box.

To properly create the Powersoft repository tables, the first person to connect to the database using PowerBuilder on *any* platform must have sufficient authority to create tables and grant permissions to PUBLIC. For INFORMIX databases, this is typically the *Informix* user account.

6 Type the password required to connect to the INFORMIX database server in the Password box.

FOR INFO Click Help for more information about security considerations when specifying the password.

7 Specify the following values in the Driver-Specific Parameters box, separating the parameters with semicolons:

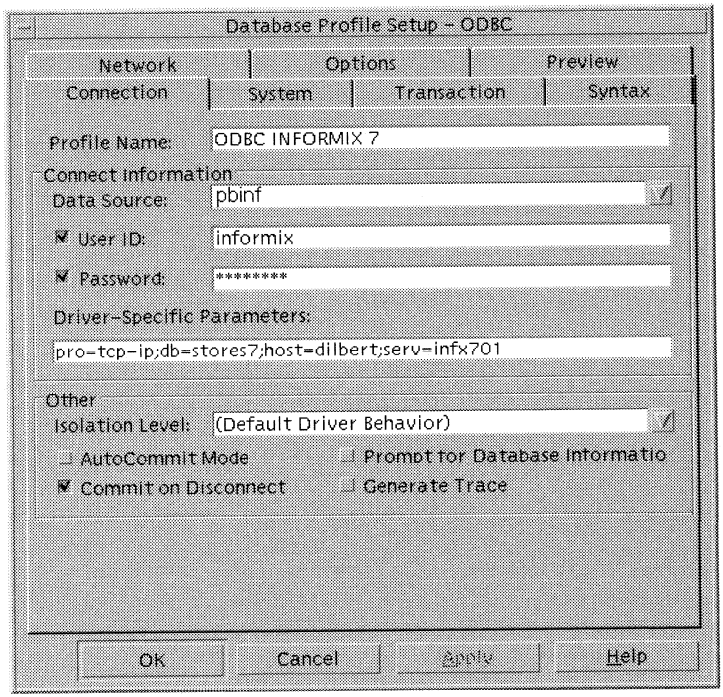
PRO=tcp-ip;**DB**=*database*;**HOST**=*host*;**SERV**=*service*

Parameter	Value
PRO=tcp-ip	The PRO (Protocol) value identifies the network protocol used to access the INFORMIX database server. The only value that PowerBuilder supports is **tcp-ip**
database	Replace this with the name of the INFORMIX database you want to access. Use the format *database@server* (for example, **stores7@informix_db**) or *database* (for example, **stores7**)
host	Replace this with the name of the INFORMIX database server. This is the same as the $INFORMIXSERVER environment variable FOR INFO For information about accessing an INFORMIX database on a *different* server than the one defined as $INFORMIXSERVER, see "Accessing a different INFORMIX database server" on page 83
service	Replace this with the name of the service as it appears in the /etc/services file and $INFORMIXDIR/etc/sqlhosts file on the database server and client machines

For example:

PRO=tcp-ip;DB=stores7;HOST=sales;SERV=infx701

A completed Connection tab might look like the following when you are accessing an INFORMIX data source:



8 Click Apply on the Connection tab.

The values you specified in steps 4 through 7 display in the ConnectString DBParm parameter. (You can click the Preview tab to see the ConnectString syntax that PowerBuilder generates.)

For example, here is the ConnectString syntax for an INFORMIX 7 data source:

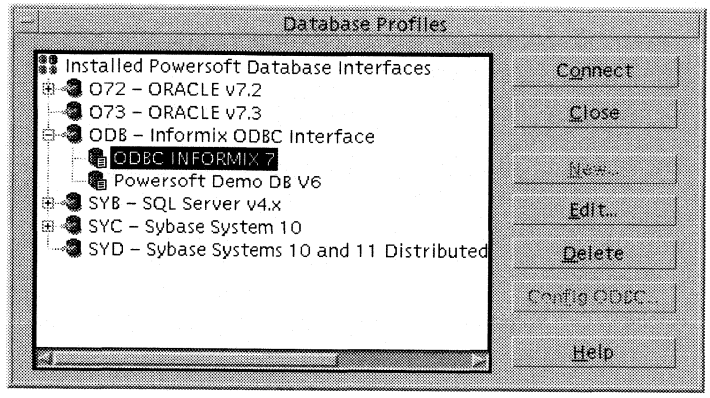
```
SQLCA.DBParm="ConnectString='DSN=pbinf;
   UID=informix;PWD=informix;PRO=tcp-ip;
   DB=stores7;HOST=sales;SERV=infx701'"
```

9 (Optional) On the other tabbed pages, supply values for any additional connection options (DBParm parameters and SQLCA properties) that you want to set.

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

10 Click OK.

The Database Profiles dialog box displays, with the new profile name highlighted under the ODBC interface. PowerBuilder saves your profile values in the .pb.ini file.



11 With your profile name selected, click Connect in the Database Profiles dialog box.

PowerBuilder connects to the specified INFORMIX database.

Accessing a different INFORMIX database server

Databases on the same database server

In the PowerBuilder for UNIX development environment, you can switch connections to a different INFORMIX database running on the *same database server* as the one defined in your $INFORMIXSERVER environment variable.

For example, suppose your $INFORMIXSERVER environment variable points to a database server named Sales, and Sales is running two databases: Sales1 and Sales2. You can freely switch connections between Sales1 and Sales2 in the PowerBuilder development environment.

Databases on different database servers

To switch connections to an INFORMIX database running on a *different database server* than the one defined in your $INFORMIXSERVER environment variable, you must exit PowerBuilder, redefine $INFORMIXSERVER, and then restart PowerBuilder and connect to the new database.

For example, suppose your $INFORMIXSERVER environment variable points to the Sales database server and PowerBuilder is currently connected to the Sales2 database. To switch connections in PowerBuilder to an INFORMIX database named Employee running on a server named Personnel, you must follow these steps:

❖ **To access an INFORMIX database on a different database server in PowerBuilder:**

1 Exit PowerBuilder.

2 Redefine the $INFORMIXSERVER environment variable in your shell initialization file to point to the INFORMIX database server you want to access.

FOR INFO For instructions and an example, see "Step 4: install and configure the required INFORMIX client software" on page 77.

3 Restart PowerBuilder.

4 Connect to the database running on the database server you just defined in the $INFORMIXSERVER environment variable.

What to do next

FOR INFO For instructions on connecting to the data source, see "Connecting to a database" on page 295.

INTERSOLV Paradox 4

This section describes how to prepare and define a Paradox data source in order to connect to it using the INTERSOLV Paradox 4 ODBC driver.

Supported versions and platforms for Paradox 4

Versions

The INTERSOLV Paradox 4 ODBC driver supports connection to Paradox Version 3.x and 4.x data sources.

Paradox Version 5.x files

To access Paradox Version 5.x files in PowerBuilder, you *must* use the INTERSOLV Paradox 5 ODBC driver.

FOR INFO For instructions, see "INTERSOLV Paradox 5" on page 90.

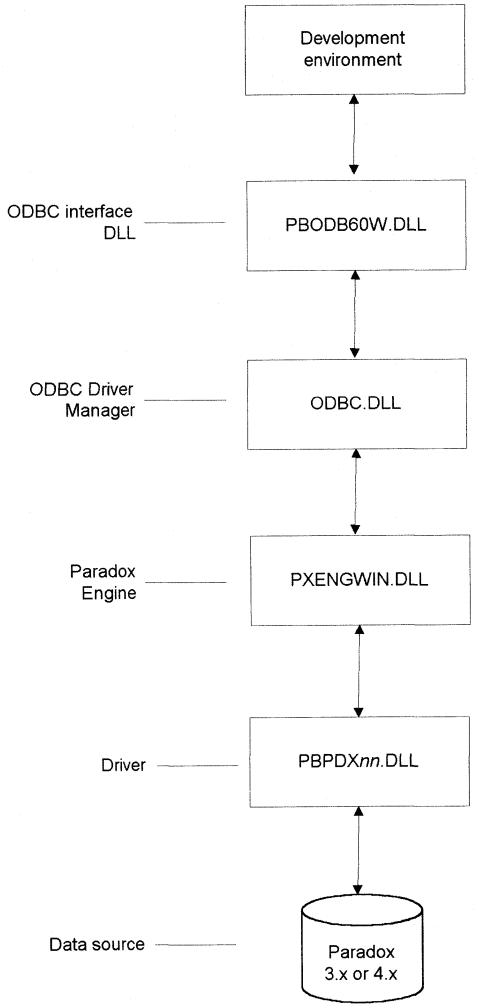
Platforms

The INTERSOLV Paradox 4 ODBC driver is available on the Windows 3.x operating system platform only.

FOR INFO For more information about supported drivers for your product, see Appendix A, "Supported ODBC Drivers and Powersoft Database Interfaces".

Basic software components for Paradox 4

The following diagram shows the basic software components you need to connect to a Paradox data source in PowerBuilder using the INTERSOLV Paradox 4 ODBC driver.



Paradox Engine (PXENGWIN.DLL)
To access a Paradox data source with the INTERSOLV Paradox 4 ODBC driver, you must first install the Windows 3.x version of the Paradox Engine (PXENGWIN.DLL) on your computer or network server. Without this DLL, you will be unable to use the driver to open Paradox tables in PowerBuilder.

Powersoft does not supply the Paradox Engine with its products. You must purchase the Paradox Engine Version 3.0 or higher from Borland.

Preparing to use the Paradox 4 data source

Before you define and connect to a Paradox data source in PowerBuilder, follow these steps to prepare the data source.

❖ **To prepare a Paradox data source for use with the INTERSOLV Paradox 4 driver:**

1 Install the Paradox Engine (PXENGWIN.DLL) on your computer or network server. Make sure PXENGWIN.DLL is in a directory on your program search path or in the Windows SYSTEM directory.

If you do not install the Paradox Engine, you will be unable to define the Paradox 4 data source.

2 Make sure the column names in your data source follow standard SQL naming conventions, as described in "About preparing ODBC data sources" on page 32.

For example, if a column name contains a Paradox reserved word, an error message may display when you try to access this table in PowerBuilder. If you see an error message, rename the column and reconnect.

FOR INFO For more about Paradox reserved words, see your Paradox documentation.

3 Make sure the DOS SHARE program is installed on your computer.

4 Issue the MS-DOS SHARE command in either of the following ways:

- Type the command from an MS-DOS prompt.
- Include the command line in your AUTOEXEC.BAT file.

5 Start Windows on your computer.

Preparing to use a shared Paradox 4 data source

In addition to accessing a local Paradox data source, you can also access shared Paradox data sources from PowerBuilder. Examples of shared Paradox data sources are:

- Paradox tables in a shared network directory
- Paradox tables on your computer that are shared among many applications

❖ **To access a shared Paradox data source:**

1 Set up a shared directory on your network and configure it to give appropriate access to all users.

FOR INFO For instructions, see your system or network administrator.

2 Use the Paradox Engine configuration utility (PXENGCFG.EXE) to specify the proper file-sharing characteristics for the tables accessed by Paradox.

FOR INFO For instructions, see your Paradox documentation.

Defining the Paradox 4 data source

❖ **To define a Paradox data source for the INTERSOLV Paradox 4 driver:**

1 Select the INTERSOLV Paradox 4 (ParadoxFile) driver in the Configure ODBC dialog box.

2 Click the Create button.

The ODBC Paradox Driver Setup dialog box displays.

3 Click the Help button for information about how to specify values in the Setup dialog box.

4 (Optional) If the Setup dialog box contains an Advanced tab or button, click it to configure additional connection options for the data source.

Click Help for information about how to specify optional settings for the Paradox 4 driver.

5 (Optional) To select an ODBC translator to translate your data from one character set to another, click the Translate button on the Advanced tab or in the Advanced Driver Setup dialog box.

FOR INFO See "Selecting an ODBC translator" on page 48.

6 Click OK to save the data source definition.

About creating an index for Paradox 4 tables

When you create the primary (unique) index for a Paradox table accessed with the INTERSOLV Paradox 4 ODBC driver, you must name the index Primary. You can then create other nonprimary (duplicate) indexes for the table as needed.

FOR INFO For instructions on creating indexes, see the *InfoMaker User's Guide*.

What to do next

FOR INFO For instructions on connecting to the data source, see "Connecting to a database" on page 295.

INTERSOLV Paradox 5

This section describes how to prepare and define a Paradox data source in order to connect to it using the INTERSOLV Paradox 5 ODBC driver.

Supported versions and platforms for Paradox 5

Versions

The INTERSOLV Paradox 5 ODBC driver supports connection to Paradox Version 3.x, 4.x, and 5.x data sources.

Platforms

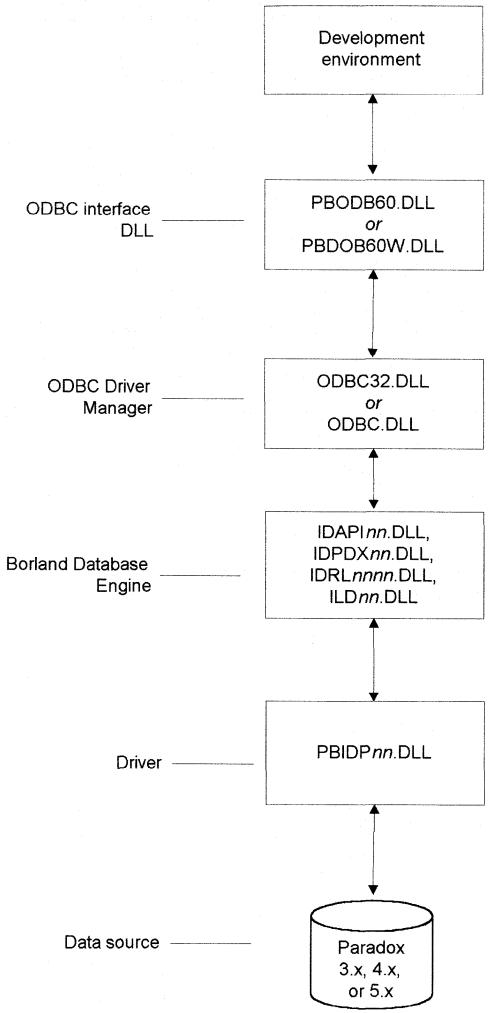
The INTERSOLV Paradox 5 ODBC driver is available on the following operating system platforms:

- Windows NT
- Windows 95

FOR INFO For more information about supported drivers for your product, see Appendix A, "Supported ODBC Drivers and Powersoft Database Interfaces".

Basic software components for Paradox 5

The following diagram shows the basic software components you need to connect to a Paradox data source in PowerBuilder using the INTERSOLV Paradox 5 ODBC driver.



Borland Database Engine

To access a Paradox data source with the INTERSOLV Paradox 5 ODBC driver, you must first install the Borland Database Engine, which conforms to the IDAPI programming interface. Without the Borland Database Engine, you will be unable to use this driver to open Paradox tables in PowerBuilder.

Powersoft does not supply the Borland Database Engine with its products. If you have installed the most recent version of Paradox for Windows or dBASE for Windows, then you should already have the Borland Database Engine on your computer.

Preparing to use the Paradox 5 data source

Before you define and connect to a Paradox data source in PowerBuilder, follow these steps to prepare the data source.

❖ **To prepare a Paradox data source for use with the INTERSOLV Paradox 5 driver:**

1 Install the Borland Database Engine on your computer or network server. Make sure you have the following files (where *nn* or *nnnn* represents the most recent version number):

- IDAPI*nn*.DLL
- IDPDX*nn*.DLL
- IDRL*nnnn*.DLL
- ILD*nn*.DLL

If you do not install the Borland database engine, you will be unable to define the Paradox 5 data source.

2 Make sure the files in the Borland Database Engine are in your program search path or Windows SYSTEM directory.

3 Make sure the PARADOX.NET file and IDAPI configuration file are correctly configured. (These files come with the Paradox software.)

FOR INFO For instructions, see your Paradox documentation.

4 Make sure the column names in your data source follow standard SQL naming conventions, as described in "About preparing ODBC data sources" on page 32.

For example, if a column name contains a Paradox reserved word, an error message may display when you try to access this table in PowerBuilder. If you see an error message, rename the column and reconnect.

FOR INFO For more about Paradox reserved words, see your Paradox documentation.

Preparing to use a shared Paradox 5 data source

In addition to accessing a local Paradox data source, you can also access shared Paradox data sources in PowerBuilder. Examples of shared Paradox data sources are:

- Paradox tables in a shared network directory
- Paradox tables on your computer that are shared among many applications

❖ **To access a shared Paradox data source:**

1 Set up a shared directory on your network and configure it to give appropriate access to all users.

FOR INFO For instructions, see your system or network administrator.

2 Make sure you specify the following connection parameters when you define the data source:

- **Database Directory** Specifies the directory containing the Paradox tables you want to access.
- **Network Directory** If the database directory you specify is a shared network directory, Paradox also requires a Network Directory (NetDir) setting. This setting specifies the directory containing the PARADOX.NET file for the database you want to access.

3 Make sure every user who accesses the same shared directory of Paradox tables has a Network Directory setting that points to the same PARADOX.NET file.

If your connect string does not specify a Network Directory value, Paradox uses the NetDir setting specified in the Paradox section of the IDAPI configuration file. Therefore, make sure the IDAPI configuration file includes the correct NetDir setting.

FOR INFO For instructions, see your Paradox documentation.

Defining the Paradox 5 data source

❖ **To define a Paradox data source for the INTERSOLV Paradox 5 driver:**

1 Select the INTERSOLV Paradox driver in the Configure ODBC dialog box.

2 Click the Create button.

The ODBC Paradox 5 Driver Setup dialog box displays.

3 Click the Help button for information about how to specify values in the Setup dialog box.

4 (Optional) If the Setup dialog box contains an Advanced tab or button, click it to configure additional connection options for the data source.

Click Help for information about how to specify optional settings for the Paradox 5 driver.

5 (Optional) To select an ODBC translator to translate your data from one character set to another, click the Translate button on the Advanced tab or in the Advanced Setup dialog box.

FOR INFO See "Selecting an ODBC translator" on page 48.

6 Click OK to save the data source definition.

About creating an index for Paradox 5 tables

When you create the primary (unique) index for a Paradox table accessed with the INTERSOLV Paradox 5 ODBC driver, you must name the index Primary. You can then create other nonprimary (duplicate) indexes for the table as needed.

FOR INFO For instructions on creating indexes, see the *PowerBuilder User's Guide*.

What to do next

FOR INFO For instructions on connecting to the data source, see "Connecting to a database" on page 295.

INTERSOLV Scalable SQL

This section describes how to prepare and define a Scalable SQL data source in order to connect to it using the INTERSOLV Scalable SQL ODBC driver.

Scalable SQL was formerly NetWare SQL
The INTERSOLV Scalable SQL ODBC driver was formerly called the INTERSOLV NetWare SQL ODBC driver.

Supported versions and platforms for Scalable SQL

Versions

The INTERSOLV Scalable SQL ODBC driver supports connection to Scalable SQL Version 3.x databases.

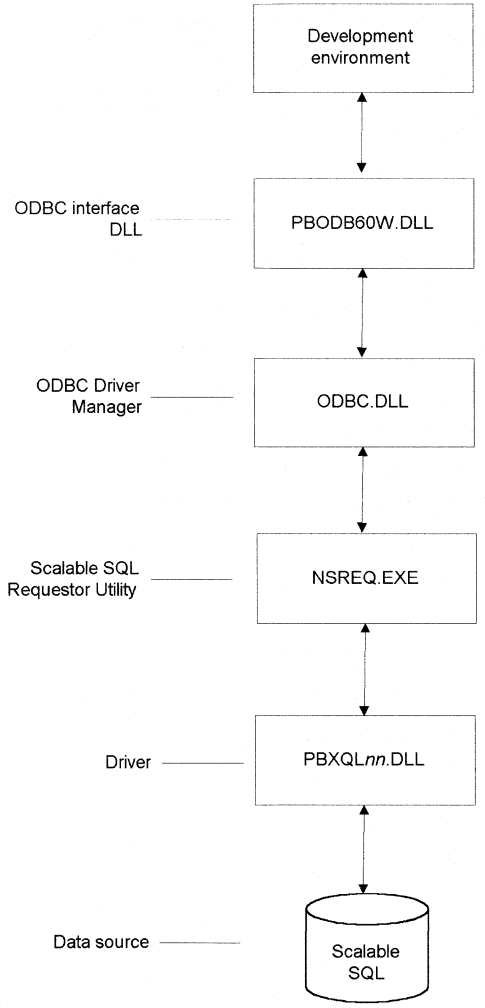
Platforms

The INTERSOLV Scalable SQL ODBC driver is available on the Windows 3.x operating system platform only.

FOR INFO For more information about supported drivers for your product, see Appendix A, "Supported ODBC Drivers and Powersoft Database Interfaces".

Basic software components for Scalable SQL

The following diagram shows the basic software components you need to connect to a Scalable SQL data source in PowerBuilder using the INTERSOLV Scalable SQL ODBC driver.



Scalable SQL Requestor utility (NSREQ.EXE)
To access a Scalable SQL data source from PowerBuilder, you must first install and start the Scalable SQL Requestor Utility (NSREQ.EXE) on your computer or network server before running Windows and starting PowerBuilder.

Powersoft *does not supply* the Scalable SQL Requestor utility with its products. You must purchase the Scalable SQL Requestor software from Novell, which includes the Scalable SQL Requestor utility.

Preparing to use the Scalable SQL data source

Before you define and connect to a Scalable SQL data source in PowerBuilder, follow these steps to prepare the data source.

❖ **To prepare a Scalable SQL data source for use with the INTERSOLV Scalable SQL driver:**

1 Install the Scalable SQL Requestor utility (NSREQ.EXE) on your computer or network sever.

2 Start the Scalable SQL Requestor utility with a minimum data size setting of 4096 (the default).

For example, to start NSREQ.EXE with a 32 K data size setting, type the following at a DOS prompt:

```
NSREQ /D:32767
```

3 Make sure the file WXQLCALL.DLL (the Scalable SQL client driver) is installed in a directory on your path or in the Windows SYSTEM directory.

4 Start Windows on your computer.

Defining the Scalable SQL data source

❖ **To define a Scalable SQL data source for the INTERSOLV Scalable SQL driver:**

1 Select the INTERSOLV Scalable SQL driver in the Configure ODBC dialog box.

2 Click the Create button.

The ODBC Scalable SQL Driver Setup dialog box displays.

3 Click the Help button for information about how to specify values in the Setup dialog box.

4 (Optional) If the Setup dialog box contains an Advanced tab or button, click it to configure additional connection options for the data source.

Click Help for information about how to specify optional settings for the Scalable SQL driver.

5 (Optional) To select an ODBC translator to translate your data from one character set to another, click the Translate button on the Advanced tab or in the Advanced Driver Setup dialog box.

FOR INFO See "Selecting an ODBC translator" on page 48.

6 Click OK to save the data source definition.

What to do next

FOR INFO For instructions on connecting to the data source, see "Connecting to a database" on page 295.

INTERSOLV Text

This section describes how to prepare and define a text data source in order to connect to it using the INTERSOLV Text ODBC driver.

Supported versions and platforms for text files

Versions

The INTERSOLV Text ODBC driver supports connection to any ASCII text file.

Platforms

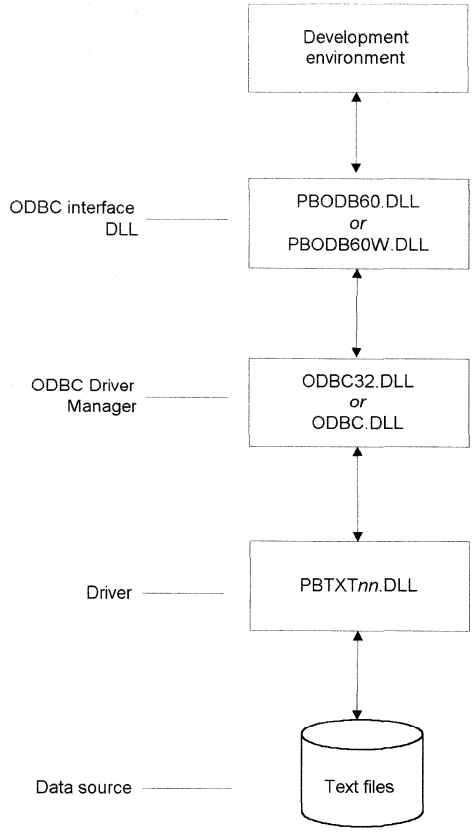
The INTERSOLV Text ODBC driver is available on the following operating system platforms:

- Windows NT
- Windows 95

FOR INFO For more information about supported drivers for your product, see Appendix A, "Supported ODBC Drivers and Powersoft Database Interfaces".

Basic software components for text files

The components you need to connect to a text data source in PowerBuilder using the INTERSOLV Text ODBC driver.



Preparing to use the text data source

Structure of a text database

The INTERSOLV Text ODBC driver supports connection to a text database. A **text database** is a directory (or folder) that contains one or more text files formatted as tables. Because the data source name references a particular directory, you can use a single data source connection to access any text tables in this directory.

For example, suppose you have a directory named C:\SALES that contains three text tables: CUSTOMER.TXT, PRODUCT.TXT, and VENDOR.TXT. You then define an ODBC text data source named Sales, which references the C:\SALES directory. When you connect to the Sales data source in PowerBuilder, you can access data from all three tables.

What you do

Before you define and connect to a text data source in PowerBuilder, follow these steps to prepare the data source.

❖ **To prepare a text data source for use with the INTERSOLV Text driver:**

1 Make sure all text files that you want to access in PowerBuilder through a single data source connection are in the same directory or folder on your computer.

To access text files in a directory *other* than the one referenced by your data source definition, do either of the following:

- Define a new ODBC data source for the INTERSOLV Text driver that references this directory.
- Move all text files to the directory referenced by your data source definition.

2 Make sure you've defined the structure of the text tables you want to access.

Because not all text files have the same structure, the INTERSOLV Text driver allows you to define the structure of a text table so the driver can access it.

FOR INFO See "Defining the structure of text tables" on page 104.

Removing the file extension is not required

You need *not* rename the text file to remove file extensions such as ASC, CSV, TAB, and TXT before you can access it as a table in PowerBuilder with the INTERSOLV Text driver.

The name that displays for this table in PowerBuilder is the one you specify in the Define Table dialog box (see "Defining the structure of text tables" on page 104).

Defining the text data source

❖ **To define a text data source for the INTERSOLV Text driver:**

1 Select the INTERSOLV Text (TextFile) driver in the Configure ODBC dialog box.

2 Click the Create button.

The ODBC Text Driver Setup dialog box displays.

3 Click the Help button for information about how to specify values in the Setup dialog box.

4 (Optional) If the Setup dialog box contains an Advanced tab or button, click it to configure additional connection options for the data source.

Click Help for information about how to specify optional settings for the Text driver.

5 (Optional) To define the table structure, click the Define button on the Advanced tab or in the Advanced Driver Setup dialog box.

FOR INFO See "Defining the structure of text tables" next.

6 (Optional) To select an ODBC translator to translate your data from one character set to another, click the Translate button on the Advanced tab or in the Advanced Driver Setup dialog box.

FOR INFO See "Selecting an ODBC translator" on page 48.

7 Click OK to save the data source definition.

Defining the structure of text tables

In order to access tables in a text database, the INTERSOLV Text driver needs information about the structure of each table in the database. As part of defining a text data source, you can use the Define Table dialog box to define the structure of each table you want to access.

The INTERSOLV Text driver stores the definitions for all tables in the database in a file named QETXT.INI in the data source directory. There is a separate QETXT.INI file for each text database.

❖ **To define the structure of a text table with the INTERSOLV Text driver:**

1 Click the Define button on the Advanced tab or in the Advanced Driver Setup dialog box.

The Define File dialog box displays.

2 Select the text table to define.

When you use text data sources, a directory or folder represents the database and each file in the directory represents a table in the database.

3 Click Open or OK.

The Define Table dialog box displays. The Database Name box displays the text database directory and the File box displays the text table you selected.

4 Click the Help button for information about how to specify values in the Define Table dialog box.

Table name

The name you specify in the Table box will display as the name for this text table in the Select Tables dialog box in PowerBuilder.

Make sure that the table name you specify has no extension. (If you omit the table name, the default name is the filename without its extension.)

For example, the default table name for the STATES.TXT file is STATES.

---

5 Click OK to save the table definition.

The Define File dialog box displays.

6 If necessary, repeat steps 2 through 5 to define the structure of other tables in the text database.

7 Click Cancel in the Define File dialog box when you are finished defining the structure of all tables in the text database.

You are returned to the Advanced tab or Advanced Driver Setup dialog box.

What to do next

FOR INFO For instructions on connecting to the data source, see "Connecting to a database" on page 295.

Sybase SQL Anywhere

This section describes how to prepare and define a Sybase SQL Anywhere data source in order to connect to it using the SQL Anywhere ODBC driver on the Windows and Macintosh platforms. (The SQL Anywhere ODBC driver was formerly called the Watcom SQL ODBC driver.)

Accessing remote SQL Anywhere databases

The SQL Anywhere network server and SQL Anywhere Client, which you must install to access remote SQL Anywhere databases, are *not included* with the SQL Anywhere standalone engine that comes with PowerBuilder.

FOR INFO See "Accessing SQL Anywhere remote databases" on page 132.

FOR INFO For information about using Sybase SQL Anywhere, see the *Sybase SQL Anywhere User's Guide* available in the Powersoft Online Books.

Supported versions and platforms for SQL Anywhere

Versions

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

- PowerBuilder running on your Windows or Macintosh computer
- SQL Anywhere Version 5.x
- Watcom SQL Version 4.x

Platforms

The SQL Anywhere ODBC driver is available on the following operating system platforms:

- Windows NT
- Windows 95
- Macintosh

FOR INFO For more information about supported drivers for your product, see Appendix A, "Supported ODBC Drivers and Powersoft Database Interfaces".

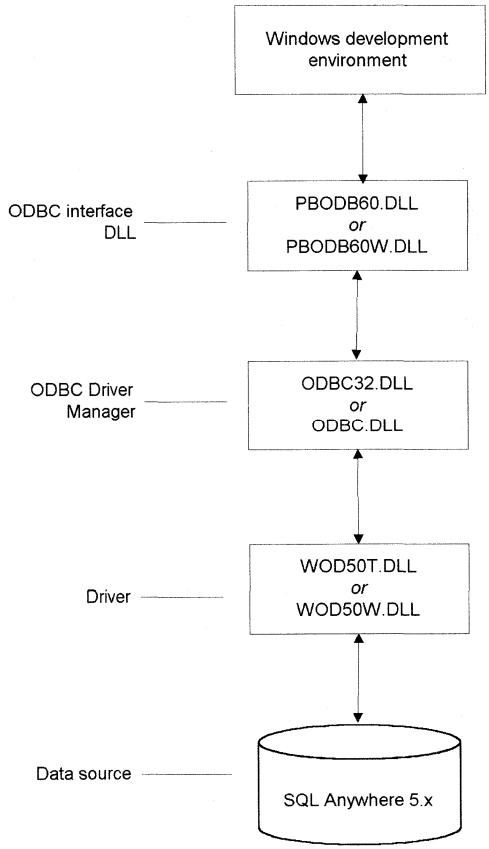
On UNIX

SQL Anywhere is unavailable on the UNIX platform. Therefore, PowerBuilder for UNIX *does not provide* the SQL Anywhere standalone database engine or the ability to create and delete local SQL Anywhere databases.

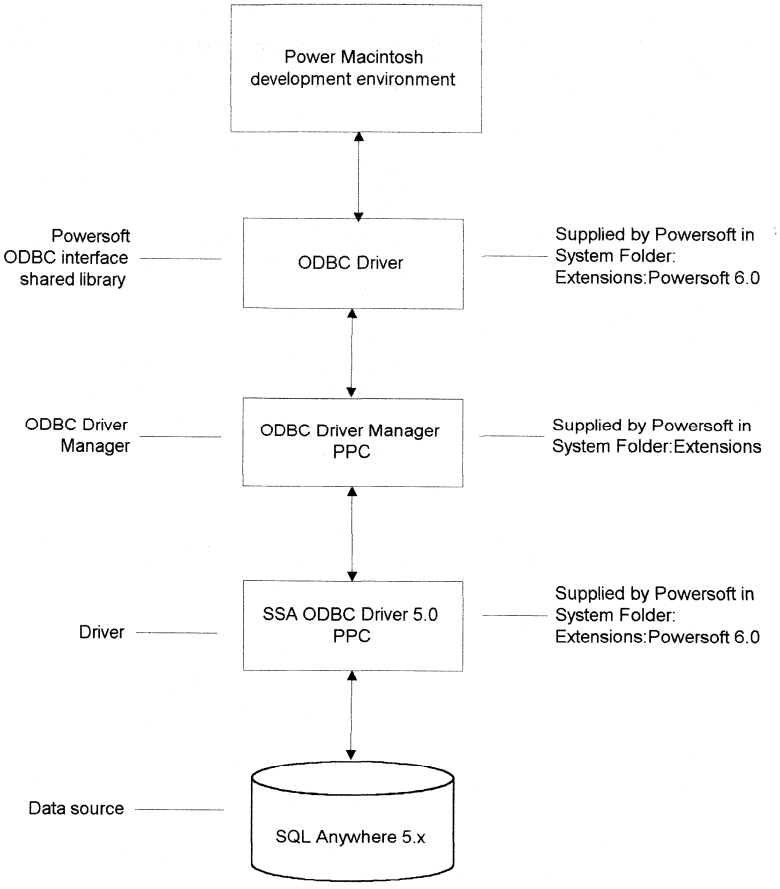
Basic software components for SQL Anywhere

The following diagrams show the basic software components you need on the Windows and Power Macintosh platforms to connect to a SQL Anywhere data source in PowerBuilder.

Accessing SQL Anywhere on Windows



Accessing SQL Anywhere on Power Macintosh



Preparing to use the SQL Anywhere data source on Windows and Macintosh

Before you define and connect to a SQL Anywhere data source in PowerBuilder on the Windows or Macintosh platforms, follow these steps to prepare the data source.

❖ **To prepare a SQL Anywhere data source:**

1 Make sure the database file for the SQL Anywhere data source already exists. You can either:

- **Create a new database** You can create a new SQL Anywhere database by:
  - Selecting File>Create Database in the Database painter when running PowerBuilder on your computer.

    This method creates a local SQL Anywhere database on your computer, and also creates the data source definition and database profile for this connection. (For instructions, see the *PowerBuilder User's Guide*.)
  - Creating the database some other way, such as with PowerBuilder running on another user's computer, or by using SQL Anywhere outside PowerBuilder. (For more about using SQL Anywhere to create a database, see the *Sybase SQL Anywhere User's Guide*.)
- **Move an existing database between platforms** You can move a SQL Anywhere database from the Windows platform to the Macintosh platform, or vice versa. SQL Anywhere databases are binary compatible and run on either platform. After you move a database from Windows to the Macintosh, drag the Windows DB file and drop it on the SQL Anywhere icon. SQL Anywhere converts the file and changes the Windows icon to the appropriate Macintosh SQL Anywhere icon.

2 Make sure you have the LOG file associated with the SQL Anywhere database so you can fully recover the database if it becomes corrupted.

If the LOG file for the SQL Anywhere database does not exist, the SQL Anywhere database engine will create 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.

What to do next

FOR INFO If you are defining a SQL Anywhere data source on the *Windows* platform, go to "Defining the SQL Anywhere data source on Windows" next.

FOR INFO If you are defining a SQL Anywhere data source on the *Macintosh* platform, go to "Defining the SQL Anywhere data source on Macintosh" on page 117.

Defining the SQL Anywhere data source on Windows

When you create a local SQL Anywhere database in the Database painter, PowerBuilder automatically creates the data source definition and database profile for you.

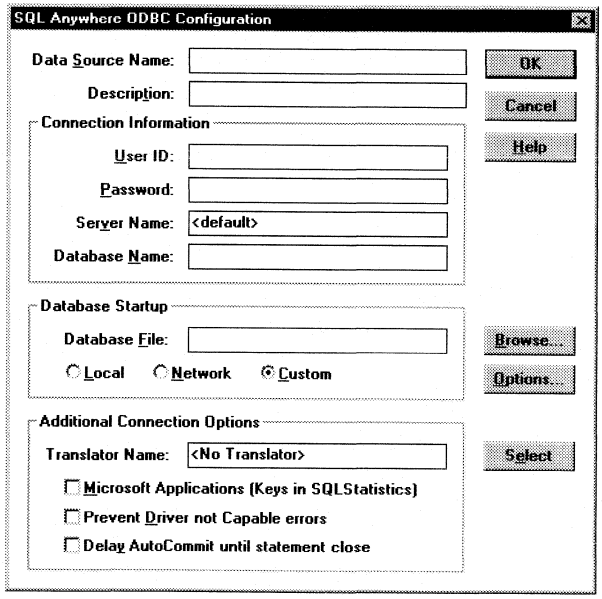
Therefore, you need only use the following procedure to define a SQL Anywhere data source when you want to access a SQL Anywhere database not created using PowerBuilder on your computer.

❖ **To define a SQL Anywhere data source for the SQL Anywhere driver on Windows:**

1 Select the SQL Anywhere driver in the Configure ODBC dialog box.

2 Click the Create button.

The SQL Anywhere ODBC Configuration dialog box displays:



3 Click the Help button for information about how to specify values 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.

FOR INFO For instructions on specifying database startup options, see "Specifying SQL Anywhere database startup options on Windows" on page 112.

FOR INFO For information about differences in the way you complete this dialog box for local and remote connections, see "Accessing local and remote SQL Anywhere databases" on page 131.

4 (Optional) If you need to select an ODBC translator to translate your data from one character set to another, click the Select button.

FOR INFO See "Selecting an ODBC translator" on page 48.

5 Click OK to save the data source definition.

Specifying SQL Anywhere database startup options on Windows

Local, Network, and Custom startup options

The SQL Anywhere ODBC driver uses the settings specified in the Database Startup box to start the database engine or SQL Anywhere client when it cannot find a running database engine or client using the Connection Information settings.

❖ **To specify database startup options for the SQL Anywhere driver on Windows:**

- In the SQL Anywhere ODBC Configuration dialog box, click one of the following radio buttons in the Database Startup box to start the SQL Anywhere database engine or client when the named engine or client is not already running:

Click this button	To
Local	Start the 32-bit standalone database engine with the default command **dbeng50** (on Windows 95 and Windows NT), and start the database specified in the Database File box. If you click Local, you cannot edit the command to add nondefault startup commands or database switches Select Local if you are using the SQL Anywhere 32-bit standalone database engine supplied with PowerBuilder and want to start the engine with the default settings FOR INFO For more about SQL Anywhere local connections, see "Accessing local and remote SQL Anywhere databases" on page 131 **Using Local** In most cases, selecting the Local startup option should meet your needs when you are accessing a local SQL Anywhere database in the PowerBuilder development environment.
Network	Start the SQL Anywhere Client with the default command **dbclient** (on Windows 95 and Windows NT). If you click Network, you cannot edit the command to add nondefault startup commands or database switches Select Network if you are using the SQL Anywhere Client to access the SQL Anywhere network server specified in the Server Name box and want to start the client with the default settings FOR INFO For more about SQL Anywhere remote (network) connections, see "Accessing local and remote SQL Anywhere databases" on page 131
Custom	Specify nondefault startup options for the database engine or client in the Startup Options dialog box. Click Custom and then click the Options button to display the Startup Options dialog box Select Custom and complete the Startup Options dialog box if you want to start the database engine or client with nondefault startup commands or database switches FOR INFO For instructions on completing the Startup Options dialog box, see "How to specify nondefault startup options" on page 114

Using nondefault startup options

In most cases, you should not have to change the default commands that start the SQL Anywhere database engine or client in PowerBuilder. As described in the preceding table, these default commands are **dbeng50** (associated with the Local radio button) and **dbclient** (associated with the Network radio button).

Sometimes, however, you may need to start the database engine or client with nondefault commands, database switches, and options. Some possible reasons for this are:

- **Setting engine switches** You want to start the database engine with a command that includes one or more engine switches, or with a nondefault startup command.

  For example, you can use the **-n** engine switch to specify a name for the database engine (for example, **dbeng50 -n myengine**).

- **Setting database switches** You want to set one or more database switches.

  For example, you can use the **-n** database switch to specify the name of the database running on a particular engine (for example, **-n mydatabase**).

- **Setting Autostop Database** You want to turn off the Autostop Database option.

  By default, Autostop Database is on. This option automatically stops the database engine when the last database is shut down, or automatically stops the SQL Anywhere Client when there are no more open database connections to it. You may want to change this default behavior by turning off Autostop Database.

- **Setting other startup options** You want to set other nondefault startup options for the database engine or client.

  Other options that you can set in the Startup Options dialog box specify the agent you want to connect to when you are running both a SQL Anywhere database engine and client on the same machine, the initial isolation level that you want all connections to use, and when you want SQL Anywhere to get updated result set descriptions.

How to specify nondefault startup options

To specify nondefault commands, database switches, and options for a SQL Anywhere database engine or client, you must complete the Startup Options dialog box.

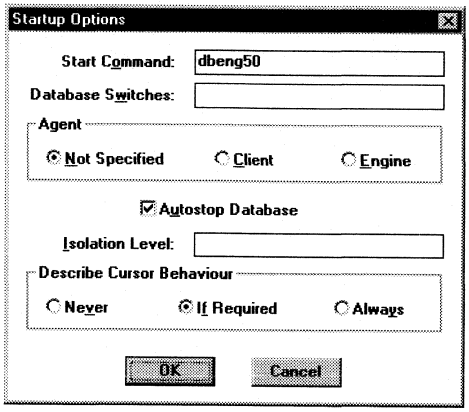
❖ **To specify nondefault startup options for the SQL Anywhere driver on Windows:**

1 In the SQL Anywhere ODBC Configuration dialog box, click the Custom radio button.

The Options button becomes enabled.

2 Click the Options button.

The Startup Options dialog box displays:



3 Specify values as follows:

Box	Value
Start Command	The command you want to use to start the SQL Anywhere database engine or client if it is not already running. For example, if you are using PowerBuilder on Windows 95 or Windows NT, you can type the command **dbeng50 -n myengine** to start the database engine with the name myengine FOR INFO For more about commands that start the database engine, see the *Sybase SQL Anywhere User's Guide*
Database Switches	The switch or switches you want to use to specify database-specific options. For example, you can type **-n mydatabase** to specify mydatabase as the name of your database file FOR INFO For more about database switches, see the *Sybase SQL Anywhere User's Guide*

Box	Value
Agent	Specifies whether you want SQL Anywhere to connect to only the database engine or only the client if you are running both a database engine and client on the same machine (Default = Not Specified)
Autostop Database	Determines whether the SQL Anywhere database engine or client stops automatically when you disconnect from the last database (Default = Yes) Clear the Autostop Database checkbox if you do *not* want the database engine or client to stop automatically when you disconnect from the last database Leave the Autostop Database checkbox selected if you want the database engine or client to stop automatically when you disconnect from the last database
Isolation Level	Specifies the initial isolation level that you want to use for all connections. Setting an isolation level here is useful if you want to override the default setting when you first connect to the database (Default = 0) FOR INFO For information about the isolation levels that SQL Anywhere supports and the values you can supply in this box, see the *SQL Anywhere User's Guide*

Box	Value
Describe Cursor Behaviour	Specifies when SQL Anywhere tries to get an updated result set description of an ANSI SQL or Transact-SQL statement that you have prepared. Values are: ♦ **Never** SQL Anywhere never tries to get an updated result set description. You may want to use this setting if you are working with ANSI SQL statements, because SQL Anywhere can get an accurate result set description of ANSI SQL statements at prepare time ♦ **If Required** (Default) SQL Anywhere tries to get an updated result set description only if required. Typically, this is required only for Transact-SQL statements, because SQL Anywhere cannot always get an accurate result set description of Transact-SQL statements at prepare time ♦ **Always** SQL Anywhere always tries to get an updated result set description

4 Click OK.

You are returned to the SQL Anywhere ODBC Configuration dialog box.

Defining the SQL Anywhere data source on Macintosh

Same as Windows

The procedure for defining a SQL Anywhere data source in PowerBuilder is the same on the Windows and Macintosh platforms. On Macintosh, as on Windows, PowerBuilder automatically defines the data source and creates the database profile for you when you create a local SQL Anywhere database in the Database painter.

Therefore, you need only use the following procedure to define a SQL Anywhere data source when you want to access a SQL Anywhere database *not created* using PowerBuilder for Macintosh on your computer.

Different from Windows

Although the procedure for defining a SQL Anywhere ODBC data source is the same on Windows and Macintosh, the information you supply in the SQL Anywhere ODBC Configuration dialog box differs on the Macintosh platform.

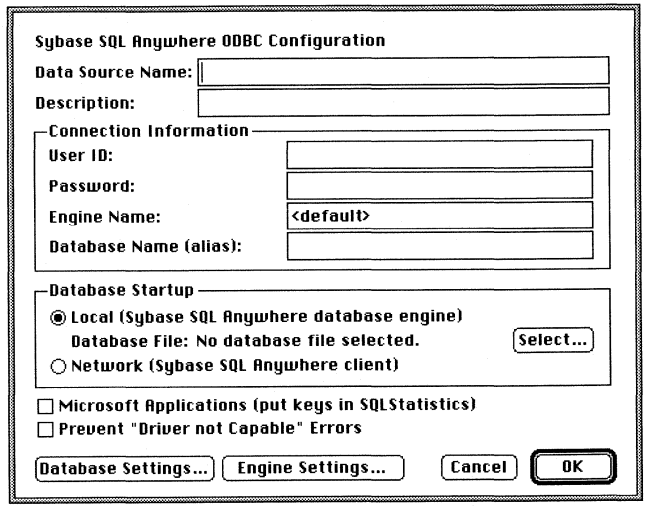
What you do

❖ **To define a SQL Anywhere data source for the SQL Anywhere driver on Macintosh:**

1 Select the SSA ODBC Driver 5.0 PPC (on Power Macintosh) in the Configure ODBC dialog box.

2 Click the Create button.

The Sybase SQL Anywhere ODBC Configuration dialog box displ:ys.



3 Specify values as follows:

Field	Value
Data Source Name	A short name to identify the data source. This becomes the name of the database profile created for this data source Spaces are permitted, but parentheses are prohibited
Description	(Optional) A description of the data source

Field	Value
User ID	(Optional) The user name required to connect to the data source If you omit the user ID, PowerBuilder prompts you for it when you connect to the data source, unless the UID (user ID) value is already in the ConnectString. (If the ODBC driver returns the UID value when you connect, PowerBuilder copies this value to the ConnectString DBParm in your database profile)
Password	(Optional) The password for the specified user ID Because the actual password is stored in the ODBC Preferences file, specifying a password in the SQL Anywhere ODBC Configuration dialog box may be a security risk. Instead, you can specify the password when PowerBuilder prompts you for it during the database connection. This way, the password does not appear in the ODBC Preferences file
Engine Name	The name of a SQL Anywhere database engine or the name of a SQL Anywhere client connected to a SQL Anywhere network server If you omit the engine name, the driver uses the default local database engine
Database Name (alias)	(Optional) If you specify an alias for the database, the database engine or server looks for a database with that alias when the engine or server is started If you do not specify an alias, then a default alias is created from the database filename without the db extension. For example, the default alias for SALES.DB is SALES

Field	Value
Database Startup	The driver uses the Database Startup settings when it cannot find a running SQL Anywhere database or client using the Connection Information settings Click one of the following radio buttons to start the SQL Anywhere database engine or client when the named database engine or client is not already running: ◆ **Local** Starts a single-user database engine on the current computer, and starts the database file specified in the Database File field. Select Local if you are using the SQL Anywhere standalone database engine and want to start the engine with default settings. To specify nondefault settings *for the database*, select Local and click the Database Settings button (described later in this table). To specify nondefault settings *for the engine*, select Local and click the Engine Settings button (also described later in this table) ◆ **Network** Starts a SQL Anywhere client that connects to the multiuser server specified in the Engine Name box. Select Network if you are using the SQL Anywhere Client for the Macintosh to access a SQL Anywhere network server. To specify nondefault settings *for the client*, select Network and click the Client Settings button (described later in this table) **Using Local** In most cases, selecting the Local startup option should meet your needs when you are accessing a local SQL Anywhere database in the PowerBuilder development environment.

Field	Value
Database File	Displays the name of a database file (for example, SALES.DB). To specify a database filename, click the Select button, browse to and select the desired file, and click Open. The filename displays in the Database File field The Database File field is visible (available) when Local is selected in the Database Startup group box and dimmed (unavailable) when Network is selected
Select	If you select Local in the Database Startup group box, click Select to specify the name of the database file you want to access. The selected database name displays in the Database File field The Select button is visible (available) when Local is selected in the Database Startup group box and dimmed (unavailable) when Network is selected
Microsoft Applications (put keys in SQLStatistics)	This option is not applicable for use with PowerBuilder The ODBC standard states that the SQLStatistics function should not return primary and foreign keys. However, some applications assume that SQLStatistics does return primary and foreign keys. Selecting this checkbox makes the SQL Anywhere driver mimic the required behavior so these applications work properly
Prevent Driver not Capable Errors	This option is not applicable for use with PowerBuilder The SQL Anywhere driver returns a Driver Not Capable error because it does not support qualifiers. Some ODBC applications do not handle this error properly. Selecting this checkbox disables the error so these applications work properly
Database Settings	Click to display the Database Settings dialog box FOR INFO For instructions, see "Setting database options for SQL Anywhere on Macintosh" on page 122

Field	Value
Engine Settings	If you select Local in the Database Startup group box, click Engine Settings to specify nondefault settings for the engine in the Sybase SQL Anywhere Engine Settings dialog box FOR INFO For instructions, see "Setting database options for SQL Anywhere on Macintosh" on page 122 The Engine Settings button changes to Client Settings if you select Network in the Database Startup group box
Client Settings	If you select Network in the Database Startup group box, click Client Settings to specify nondefault settings for the client in the Sybase SQL Anywhere Client Settings dialog box FOR INFO For instructions, see "Setting database options for SQL Anywhere on Macintosh" on page 122 The Client Settings button changes to Engine Settings if you select Local in the Database Startup group box

Setting database options for SQL Anywhere on Macintosh

When to do this

Use the following procedure if you want to set nondefault options for a particular database. This procedure applies when you select *either Local or Network* in the Database Startup group box.

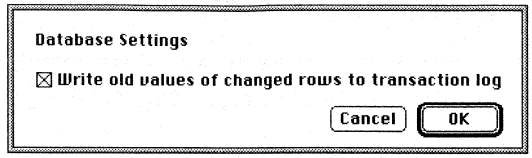
What you do

❖ **To set database options for a SQL Anywhere database on Macintosh:**

1 Click the Database Settings button in the Sybase SQL Anywhere ODBC Configuration dialog box.

The Database Settings dialog box displays.

2 Select the checkbox to write old values of changed rows to the transaction log.



By default, SQL Anywhere records only enough information in the transaction log to uniquely identify each row. This information includes primary key values or values from a not null, unique index.

Selecting this checkbox writes old values of all the columns to the transaction log whenever a row is updated.

This option applies only to the *specified database*. To write old values of changed rows to the transaction log for *all databases* run with this engine, set this option in the Sybase SQL Anywhere Engine Settings dialog box.

FOR INFO For instructions, see "Setting engine startup options for SQL Anywhere on Macintosh" next.

3 Click OK.

You are returned to the Sybase SQL Anywhere ODBC Configuration dialog box.

Setting engine startup options for SQL Anywhere on Macintosh

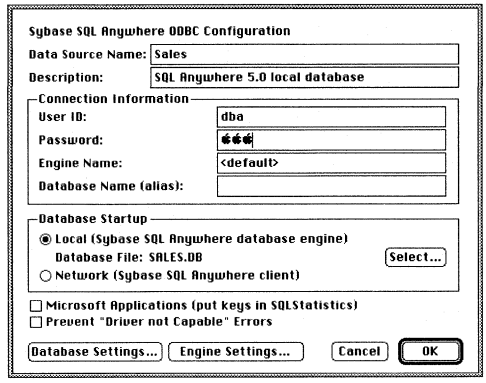
When to do this

Use the following procedure for local SQL Anywhere connections (*Local* selected in Database Startup group box) to set nondefault startup options for the SQL Anywhere database engine.

What you do

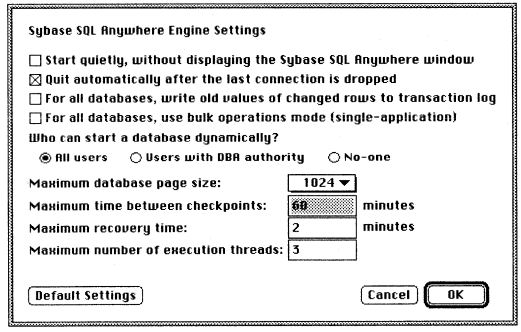
❖ **To set engine startup options for local SQL Anywhere connections on Macintosh:**

1 In the Sybase SQL Anywhere ODBC Configuration dialog box, make sure *Local* is selected in the Database Startup group box.



2 Click the Engine Settings button.

The Sybase SQL Anywhere Engine Settings dialog box displays:



3 Specify values as follows:

Field	Value
Start quietly	Prevents messages from being sent to the database engine window and starts the engine without displaying the window (Default = off)
Quit automatically after last connection	Applications can cause the database engine to start and stop databases. Selecting this checkbox causes the database engine to shut down when the last connection is dropped (Default = on)

Field	Value
Write old values to transaction log	By default, SQL Anywhere records only enough information in the transaction log to uniquely identify each row. This information includes primary key values or values from a not null, unique index. Selecting this checkbox writes old values of all the columns to the transaction log whenever a row is updated (Default = off) This option applies to *all databases* run with this engine. To write old values of changed rows to the transaction log for a *particular database*, set this option in the Database Settings dialog box FOR INFO See "Setting database options for SQL Anywhere on Macintosh" on page 122
Use bulk operations mode	In bulk operations mode, the database engine allows one connection by only one application. It does not keep a rollback log or a transaction log. Bulk operations mode is useful when you load large quantities of data into the database (Default = off)
Who can start a database dynamically?	Click one of the following radio buttons to set the permission level required to start another database dynamically on the current engine: ◆ **All users** (Default) All users can start a database dynamically ◆ **Users with DBA authority** Only users with database administrator (DBA) authority can start a database dynamically ◆ **No-one** No users are allowed to start a database dynamically A user ID must have the specified permission level in order to request that the database engine load the database file

Field	Value
Maximum database page size	Select one of the following values to set the maximum database page size in bytes: ◆ **512** ◆ **1024** (Default) ◆ **2048** ◆ **4096** When you create a database, SQL Anywhere specifies an internal page size. When the database engine is first started, it uses the largest page size of all the databases specified. Unless you first set the Maximum database page size option, you will be unable to load a database later with a larger page size
Maximum time between checkpoints	Specifies the maximum time in minutes that the database runs without performing a checkpoint. When a database engine is running with multiple databases, it uses the checkpoint time specified by the first database started unless you set this option (Default = 60 minutes)
Maximum recovery time	Specifies the maximum time in minutes that the database engine takes to recover from a system failure. This option together with the Maximum time between checkpoints option determines when checkpoints are performed (Default = 2 minutes)
Maximum number of execution threads	Specifies the maximum number of execution threads used in the database engine when it is running with multiple users. When a database engine is running with multiple databases, it uses the thread count specified by the first database started unless you set this option (Default = 3 execution threads)
Default Settings	Resets all options in the dialog box to their default values

4 Click OK.

You are returned to the Sybase SQL Anywhere ODBC Configuration dialog box.

Setting client startup options for SQL Anywhere on Macintosh

When to do this

Use the following procedure for remote SQL Anywhere connections (*Network* is selected in the Database Startup group box) to set nondefault startup options for the SQL Anywhere Client.

Accessing remote SQL Anywhere databases

You must have the necessary SQL Anywhere network server and client software installed to access a remote SQL Anywhere database in PowerBuilder.

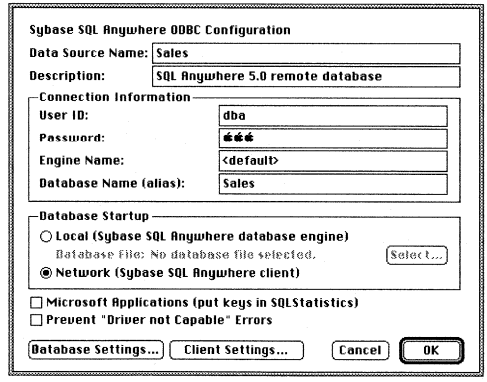
FOR INFO For instructions, see "Accessing SQL Anywhere remote databases" on page 132.

What you do

❖ **To set client startup options for remote SQL Anywhere connections on Macintosh:**

1 In the Sybase SQL Anywhere ODBC Configuration dialog box, make sure *Network* is selected in the Database Startup group box.

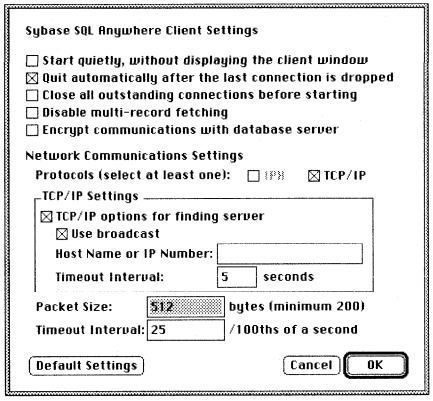
The Engine Settings button changes to Client Settings when you select Network.



2 Click the Client Settings button.

The Sybase SQL Anywhere Client Settings dialog box displays.

By default, the TCP/IP options for finding server checkbox is cleared and the Use broadcast, Host Name, and Timeout Interval options are dimmed. To make these options visible, the following screen shows the TCP/IP options checkbox selected.



3 Specify values as follows:

Field	Value
Start quietly	Prevents messages from being sent to the SQL Anywhere client window and starts the client without displaying the window (Default = off)
Quit automatically after last connection	Selecting this checkbox causes the SQL Anywhere client to shut down when the last connection is dropped (Default = on)
Close all outstanding connections before starting	If you shut down or restart your computer while there are connections to the server, one or more of these connections may be left open. Selecting this checkbox closes these connections (Default = off)
Disable multi-record fetching	By default, when a server receives a simple fetch request it fills one network packet with several rows so that subsequent sequential fetches do not require network traffic. This is called **blocking** of fetches. Selecting this checkbox prevents multiple-row fetching by specifying that the server fetch only one row per network request (Default = off)

Field	Value
Encrypt communications	By default, SQL Anywhere does not encrypt packets sent across the network. If you are concerned about network security, select this checkbox to encrypt these packets. Encryption marginally affects database performance (Default = off)
Protocols	Specifies which of the following network protocols you are using to connect to a SQL Anywhere network server: ♦ **IPX** ♦ **TCP/IP** The software determines whether IPX or TCP/IP (MacTCP) is present on your system and selects the appropriate checkbox In order to use a SQL Anywhere network server with a particular network protocol, the network software for that protocol must be installed and running on your computer FOR INFO For information about the network software you need to install, see the *SQL Anywhere Network Guide* that comes with the SQL Anywhere network server
TCP/IP options for finding server	Selecting this checkbox allows you to set options for finding the database server on a TCP/IP network (Default = off) These options control how a SQL Anywhere network server finds the database server (by issuing a broadcast message, or by using the host name or IP address) and how long it waits for a response (timeout interval) when connecting to the server

Field	Value
Use broadcast	Specifies whether a SQL Anywhere network server issues a general broadcast message to find the database server specified in the Sybase SQL Anywhere ODBC Configuration dialog box (Default = on) If you select the Use broadcast checkbox, the SQL Anywhere network server issues a broadcast message to find the server. If you do not supply a host name or IP address (as described next), the network server must issue a broadcast and therefore does not let you clear the Use broadcast checkbox If you clear the Use broadcast checkbox, the SQL Anywhere network server does not issue a broadcast message. In this situation, you *must* select the TCP/IP options checkbox and supply the database server's host name or IP address so the SQL Anywhere network server can find it
Host name or IP Number	Specifies the name or IP address of the host machine on which the database resides (Default = off) If the Use Broadcast checkbox is cleared, you *must* supply the database server's host name or IP address in this box so the SQL Anywhere network server can find it
Timeout Interval (TCP/IP)	Specifies the length of time in seconds that the SQL Anywhere client waits for a response when connecting to a database server on a TCP/IP network (Default = 5 seconds) You may want to increase this value if you are having difficulty establishing communications on your TCP/IP network
Packet Size	Specifies the maximum size in bytes of network packets. A packet for a SQL Anywhere client can be smaller than a packet for a SQL Anywhere server, but not bigger The SQL Anywhere client and server mediate their packet sizes. If the client's packet size is bigger than the server's, the client detects this from the server and uses only the smaller portion of the packet The minimum allowable packet size is 200 bytes (Default = 512 bytes)

Field	Value
Timeout Interval	Specifies the time in hundredths of a second before the client issues a request if the server has not responded. In most cases, the default value should meet your needs (Default = 25 hundredths of a second)
Default Settings	Resets all options in the dialog box to their default values

4 Click OK.

You are returned to the Sybase SQL Anywhere ODBC Configuration dialog box.

Accessing local and remote SQL Anywhere databases

This section describes differences in the way you complete the SQL Anywhere ODBC Configuration dialog box when connecting to local and remote SQL Anywhere databases in PowerBuilder.

These differences apply on *both* the Windows and Macintosh platforms.

What SQL Anywhere databases can you access?

The SQL Anywhere ODBC driver supports connection to SQL Anywhere databases that reside in either of two places:

- **Locally on your computer** You can create a local SQL Anywhere database using PowerBuilder or SQL Anywhere and access it in PowerBuilder.
- **Remotely on a SQL Anywhere network server** If you have the SQL Anywhere network server and client software installed, you can create a remote SQL Anywhere database and access it in PowerBuilder.

Accessing SQL Anywhere local databases

Follow these general steps to access a local SQL Anywhere database in PowerBuilder.

❖ **To access a local SQL Anywhere database:**

1 Make sure you've prepared to use the SQL Anywhere data source so you can access it in PowerBuilder.

FOR INFO See "Preparing to use the SQL Anywhere data source on Windows and Macintosh" on page 109.

2 In PowerBuilder, complete the SQL Anywhere ODBC Configuration dialog box to define the data source.

In general, you complete the SQL Anywhere ODBC Configuration dialog box the same way for local and remote (network) connections. However, you supply different values for some boxes if you are connecting to a local SQL Anywhere database on your computer, as follows:

Field on Windows	Field on Macintosh	Value for local connection
Server Name	Engine Name	Name of a SQL Anywhere database engine
Database Name	Database Name (alias)	Name of a SQL Anywhere database already running on a SQL Anywhere database engine
Database Startup	—	Local *or* Custom
—	Database Startup	Local

Accessing SQL Anywhere remote databases

Follow these general steps to access a remote SQL Anywhere database in PowerBuilder.

❖ **To access a remote SQL Anywhere database:**

1 Purchase and install the SQL Anywhere network server for your operating system platform.

The SQL Anywhere network server includes the SQL Anywhere Client program required to access a remote SQL Anywhere database server.

Not included with PowerBuilder

The SQL Anywhere network server and SQL Anywhere Client are *not included* with the SQL Anywhere standalone engine that comes with PowerBuilder on the Windows and Macintosh platforms.

2 Make sure the required network software for your operating system platform is installed on your computer.

In order to use the SQL Anywhere network server with a particular network protocol, the network software for that protocol must be installed and running on your computer.

FOR INFO For information about the network software you need to install, see the *SQL Anywhere Network Guide* that comes with the SQL Anywhere network server.

3 Make sure you've prepared to use the SQL Anywhere data source so you can access it in PowerBuilder.

FOR INFO See "Preparing to use the SQL Anywhere data source on Windows and Macintosh" on page 109.

4 In PowerBuilder, complete the SQL Anywhere ODBC Configuration dialog box to define the data source.

In general, you complete the SQL Anywhere ODBC Configuration dialog box the same way for local and remote (network) connections. However, you supply different values for some boxes if you are connecting to a remote SQL Anywhere database on a network server, as follows:

Field on Windows	Field on Macintosh	Value for remote connection
Server Name	Engine Name	Name of a SQL Anywhere network server
Database name	Database Name (alias)	Name of a SQL Anywhere database already running on a SQL Anywhere network server
Database Startup	—	Network *or* Custom
—	Database Startup	Network

Support for Transact-SQL special timestamp columns

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

Creating special timestamp columns

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

1. Give the name *timestamp* to any column having a timestamp data type that you want treated as a Transact-SQL special timestamp column. Do this in one of the following ways:
   - **In the Table, DataWindow, or Data Pipeline painter** Select *timestamp* as the column name. (For instructions, see the *PowerBuilder 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 Table, DataWindow, or Data Pipeline painter** Select *timestamp* as the default value for the column. (For instructions, see the *PowerBuilder 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 dropdown listbox 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 INFO For instructions, see the *PowerBuilder User's Guide*.

CREATE TABLE example

CREATE TABLE example The following CREATE TABLE statement defines a SQL Anywhere table named *timesheet* containing three columns: employee_ID (integer data type), hours (decimal data type), and timestamp (timestamp data type 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 SQL 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 SQL Anywhere section of the PBODB60 initialization file to change the value of SQLSrvrTSName from 'Yes' to 'No'.

  FOR INFO For information about the PBODB60 initialization file, see Appendix B, "Adding Functions to the PBODB60 Initialization File".

On Macintosh

On the Macintosh platform, the PBODB60 initialization file is called Powersoft ODBC 6.0 Preferences and is located in System Folder:Preferences.

What to do next

FOR INFO For instructions on connecting to the data source, see "Connecting to a database" on page 295.

CHAPTER 3

Using Powersoft Database Interfaces

Where you are

(*Optional*) Get an introduction to database connections
> Prepare to use the data source or database
> Install the ODBC driver or Powersoft database interface
> Define the data source or database interface
Connect to the data source or database
(*Optional*) Set additional connection parameters
(*Optional*) Troubleshoot the data connection

About this chapter

This chapter introduces Powersoft database interfaces. It then describes the following for each supported interface so you can use it to access your data in the PowerBuilder development environment:

- Preparing to use the database (includes installing the Powersoft database interface)
- Defining the database interface

Contents

For more information

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

- Check for a FaxLine document that describes how to connect to your database. Updated information about connectivity issues is available from the Powersoft FaxLine system and from Powersoft electronic services on CompuServe, FTP, BBS, and the World Wide Web.
- Ask your network or system administrator for assistance when installing and setting up the database server and client software at your site.

About Powersoft database interfaces

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

Supported Powersoft database interfaces

FOR INFO For a complete list of the supported Powersoft database interfaces, see Appendix A, "Supported ODBC Drivers and Powersoft Database Interfaces".

The Powersoft database interfaces are *not* supported in PowerBuilder Professional and PowerBuilder Desktop. However, you can upgrade to PowerBuilder Enterprise in order to use the Powersoft database interfaces.

What is a Powersoft database interface?

A **Powersoft database interface** is a native (direct) connection to your data in PowerBuilder.

Each Powersoft database interface uses its own interface DLL (on Windows) or shared library (on Macintosh and UNIX) to communicate with a specified database through a vendor-specific database API. For example, on Windows NT and Windows 95, the Sybase System 10 and System 11 interface uses a DLL named PBSYC60.DLL to access the database, while the Oracle 7.3 database interface accesses the database through PBO7360.DLL.

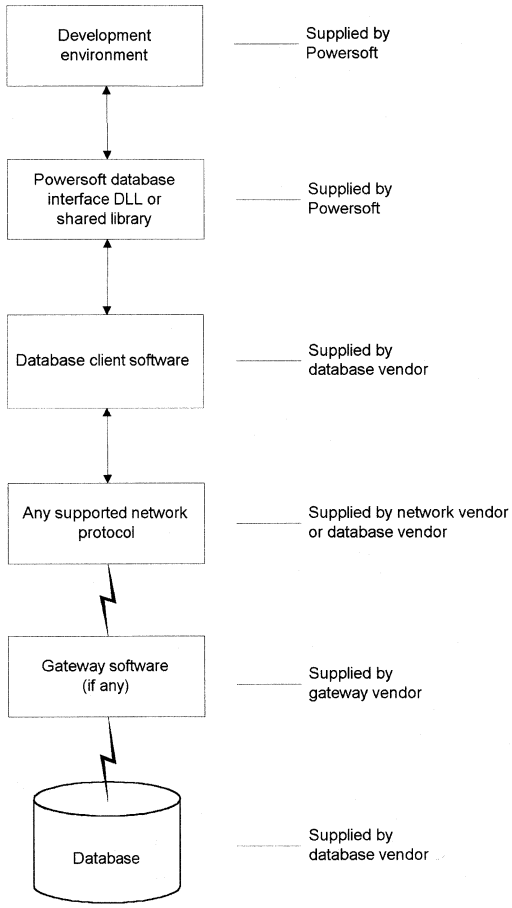
In contrast, ODBC connections use a standard API to communicate with the database. Therefore, PowerBuilder can use a single Powersoft interface DLL or shared library to communicate with the ODBC Driver Manager and corresponding driver to access any ODBC data source.

A Powersoft database interface does *not* go through an ODBC driver to access the database. Therefore, you do not complete a driver-specific ODBC setup dialog box to define it. Instead, you create a database profile in which you specify the information that PowerBuilder needs to connect to the database.

Components of a database interface connection

When you use a Powersoft 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.

The following diagram shows the basic components of a Powersoft database interface connection on *any* platform.



FOR INFO For diagrams showing the specific components of your connection, see "Basic software components" in the section in this chapter for your Powersoft database interface.

Using a Powersoft database interface

❖ **To use a Powersoft database interface to access a database:**

1 Prepare the database for use with PowerBuilder.

FOR INFO For instructions, see "Preparing to use the database" in the section in this chapter for your Powersoft database interface.

2 Install the Powersoft database interface that accesses this database.

FOR INFO For instructions, see the *PowerBuilder Installation Guide*.

3 Define the database interface to specify connection parameters.

This involves completing the Database Profile Setup dialog box for your interface.

FOR INFO For instructions, see "About defining Powersoft database interfaces" on page 143.

Platform differences

You follow the same basic steps on all PowerBuilder platforms to use a Powersoft database interface to access your data. However, PowerBuilder provides different Powersoft database interfaces on different operating system platforms.

FOR INFO For a complete list of the supported Powersoft database interfaces, see Appendix A, "Supported ODBC Drivers and Powersoft Database Interfaces".

About preparing to use the database

The first step in connecting to a database through a native Powersoft 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, define the database interface, and connect to it. The requirements differ for each database but in general, preparing a database involves the following 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 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 INFO For specific instructions to use with your database, see "Preparing to use the database" in the section in this chapter for your Powersoft database interface.

About defining Powersoft database interfaces

After you prepare to use the database as described in the preceding section, you start PowerBuilder and define the database interface. To define a database interface, you must complete the Database Profile Setup dialog box for that interface to create a database profile.

Getting help

There are several ways that you can get help while creating a database profile for a Powersoft database interface.

To get help on	Do this
Database Profiles dialog box	Click the Help button
The Database Profile Setup dialog box for your interface	Click the Help button
Your Powersoft database interface	See the Database Interfaces Help Check whether there is a FaxLine document that describes how to connect to your database. Updated information about connectivity issues is available from the Powersoft FaxLine system and from Powersoft electronic services on CompuServe, FTP, BBS, and the World Wide Web

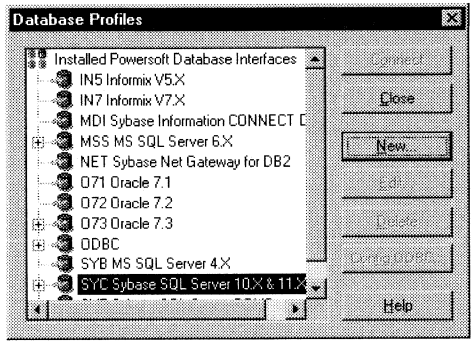
About creating database profiles

When you complete the Database Profile Setup dialog box for your interface, PowerBuilder automatically creates a database profile. A **database profile** is a named set of parameters stored in your PowerBuilder initialization file that defines a connection to a particular database in the development environment. You can select this database profile at any time to connect to the database with the connection parameters you specified.

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.

Database Profiles dialog box

The Database Profiles dialog box uses an easy-to-navigate tree control format to display your installed Powersoft database interfaces and defined database profiles. You can create, edit, and delete database profiles from this dialog box. In addition, when the ODBC interface or one of its profiles is selected, you can access the Configure ODBC dialog box to create, edit, and delete ODBC data source definitions.

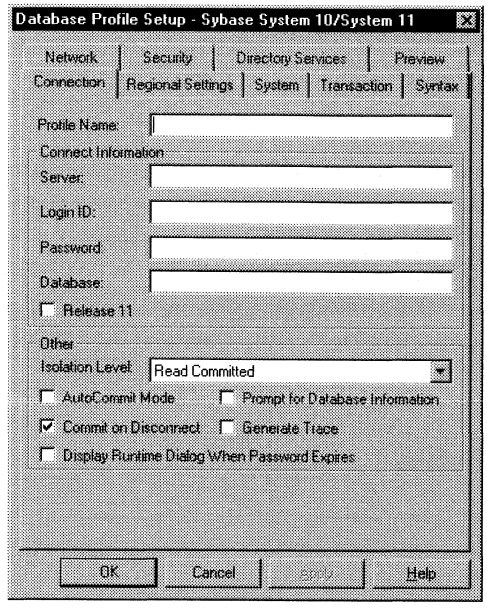


When you run the PowerBuilder Setup program, it updates the Vendors list in the [Database] section of your PowerBuilder initialization file with the interfaces you install. The Database Profiles dialog box displays the same interfaces that appear in the Vendors list.

FOR INFO For detailed instructions on using the Database Profiles dialog box to connect to a database and manage your profiles, see Chapter 4, "Managing Database Connections".

Database Profile Setup dialog box

Each Powersoft database interface has its own Database Profile Setup dialog box where you can set interface-specific connection parameters. For example, if you select the SYC interface and click New in the Database Profiles dialog box, the Database Profile Setup - Sybase System 10/System 11 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 checkboxes, dropdown listboxes, and textboxes. 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. The Preview tab is *not* available in the Database Profile Setup dialog box in InfoMaker.

Supplying sufficient information in the Database Profile Setup dialog box

For some Powersoft 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 should supply values for all applicable boxes in the Database Profile Setup dialog box.

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

Creating a database profile

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

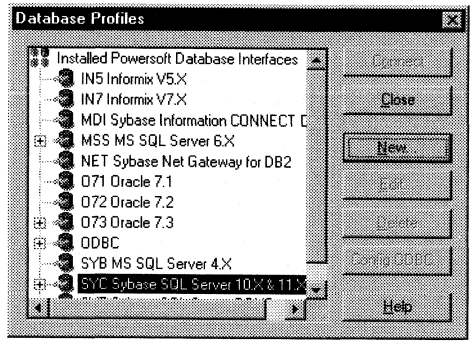
Same steps on all platforms
The following procedure for creating a database profile is the same in PowerBuilder on *all* supported platforms.

❖ **To create a database profile for a Powersoft database interface:**

1 Click the Database Profile button in the PowerBar.
or
In the Database painter, select File>Connect>Setup from the menu bar.

The Database Profiles dialog box displays, listing your installed Powersoft database interfaces. Those interfaces preceded by a plus (+) sign have one or more database profiles already defined. 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.

Where the interface list comes from
When you run the Setup program, it updates the Vendors list in the [Database] section of your PowerBuilder initialization file with the interfaces you install. The Database Profiles dialog box displays the same interfaces that appear in the Vendors list.



2 Select an interface name and click New.
or
Display the popup menu for an interface and select New.

The Database Profile Setup dialog box for the selected interface displays. For example, if you select the SYC interface, the Database Profile Setup - Sybase System 10/System 11 dialog box displays.

Client software and interface must be installed

To display the Database Profile Setup dialog box for your interface when you click New, the required client software and Powersoft database interface must be properly installed and configured.

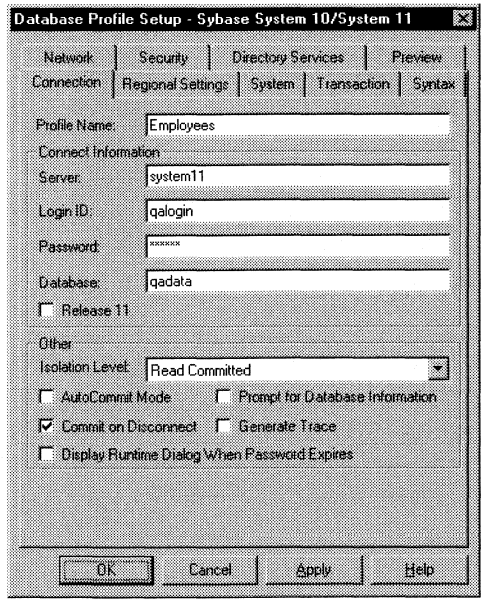
FOR INFO For specific instructions for your database interface, see "Preparing to use the database" in the section in this chapter for your interface.

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

FOR INFO 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 Powersoft database interface (including ODBC), PowerBuilder generates the correct DBMS connection syntax automatically.



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

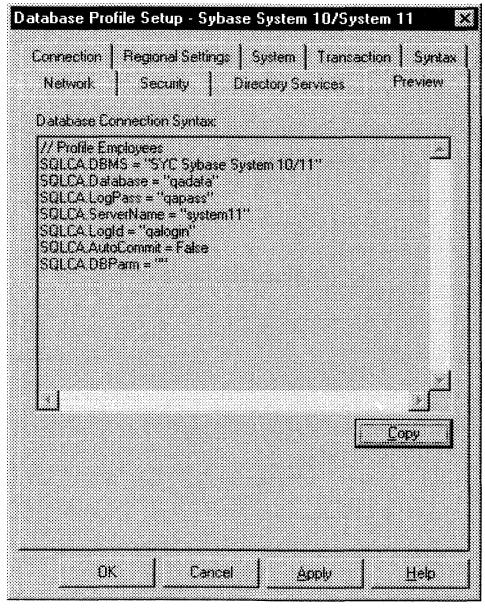
FOR INFO For information about the additional connection parameters for your interface and the values you should supply, click Help.

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.

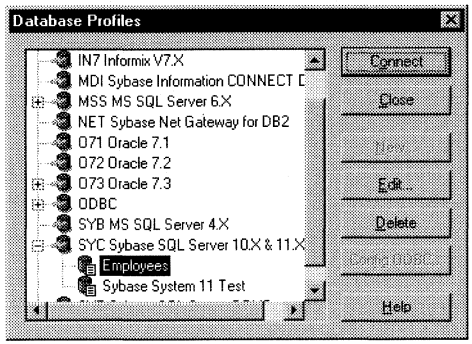
FOR INFO For instructions on using the Preview tab to help you connect in a PowerBuilder application, see the section on using transaction objects in *Application Techniques*.

Since PowerScript connection syntax does not apply to InfoMaker applications, the Preview tab is *not* available in the Database Profile Setup dialog box in InfoMaker.



6 Click OK to save your changes and close the Database Profile Setup dialog box. (To save your changes on a particular tab *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 your PowerBuilder initialization file.



What happens when you connect

PowerBuilder initialization file

When you create a database profile, the values are stored in the PowerBuilder initialization file. Depending on the platform you are using, the initialization file has different names and locations, as follows:

Win 95 and Win NT	Macintosh	UNIX
PB.INI in PowerBuilder product directory	System Folder: Preferences: Powersoft 6.0 Preferences: PowerBuilder Preferences	$HOME/.pb.ini

Sample profile

For example, the Sybase System 11 database profile named Employees in the preceding procedure creates the following entry in the PowerBuilder initialization file on Windows:

```
[Profile Employees]
DBMS=SYC Sybase System 10/11
Database=qadata
UserId=
DatabasePassword=
LogPassword=qapass
ServerName=system11
LogId=qalogin
```

```
Lock=
DbParm=
Prompt=0
AutoCommit=0
```

When you connect to the database by selecting the Employees profile, PowerBuilder copies the profile values for Employees to the [Database] section of the initialization file to indicate that this is the current connection.

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 PowerBuilder initialization file (see the "Sample profile" on page 150), the actual password *does* display in the DatabasePassword or LogPassword field. If the initialization file resides on a network at your site, displaying the password in the initialization file may be a security risk.

Suppressing display in initialization file

To suppress password display in the PowerBuilder initialization file, do the following when you create a database profile.

❖ **To suppress password display in the PowerBuilder initialization file:**

1 Select the Prompt for Database Information checkbox on the Connection tab in the Database Profile Setup dialog box.

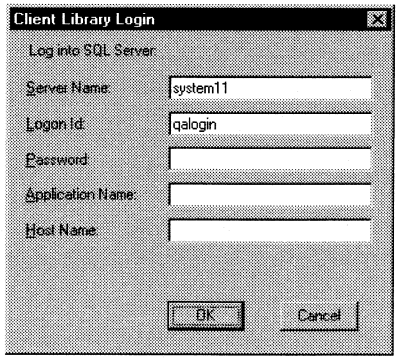
This tells PowerBuilder to prompt you for any missing information when you select this profile to connect to the database.

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 initialization file entry for this profile.

For example, if you do not supply a password in the Database Profile Setup - Sybase System 10/System 11 dialog box when creating a database profile, the Client Library Login dialog box displays to prompt you for the missing information.



Specifying the password in response to a dialog box prompt instead of in the Database Profile Setup dialog box creates the following entry in the PowerBuilder initialization file. Notice that the LogPassword value does not appear.

```
[Profile EmployeesNoPassword]
DBMS=SYC Sybase System 10/11
Database=qadata
UserId=
DatabasePassword=
LogPassword=
ServerName=system11
LogId=qalogin
Lock=
DbParm=
Prompt=1
AutoCommit=0
```

What to do next

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

IBM databases

The IBM database interface is available on the Windows 3.x operating system platform to access the following DB2 databases:

- Database 2/Common Server (DB2/CS) family of databases Version 1.x and 2.1 (includes such databases as DB2/2 and DB2/6000)
- Database 2 for MVS (DB2/MVS) Version 2 Release 3 or higher

The IBM interface uses a Powersoft DLL named PBIBM60W.DLL to access the database.

Deploying applications in PowerBuilder

The IBM database interface is available only on the Windows 3.x (16-bit Windows) platform. Therefore, you can use it for deployment purposes in PowerBuilder as part of the Powersoft Deployment Kit for 16-bit Windows.

You *cannot*, however, develop PowerBuilder applications using the IBM interface because on Windows, the PowerBuilder development environment is available only on the Windows NT and Windows 95 (32-bit Windows) platforms.

Developing and deploying applications in InfoMaker

Unlike PowerBuilder, the InfoMaker development environment is available on the Windows 3.x (16-bit Windows) platform. Therefore, you *can* use the IBM database interface to develop applications in InfoMaker for 16-bit Windows, and to deploy applications as part of the Powersoft Deployment Kit for 16-bit Windows.

FOR INFO For information about using the IBM database interface in the InfoMaker development environment, see *Connecting to Your Database (for InfoMaker)*.

Accessing IBM databases through the Powersoft ODBC interface

As an alternative to using the native IBM database interface, you can also access DB2 databases through the Powersoft ODBC interface. Accessing DB2 databases through ODBC provides you with support for additional features and operating system platforms not available when you use the native IBM interface.

FOR INFO For information on using the Powersoft ODBC interface, see Chapter 2, "Using ODBC Data Sources and Drivers".

INFORMIX

This section describes how to use the native Powersoft INFORMIX database interfaces in PowerBuilder on the Windows platform.

On UNIX

On the UNIX platform, you can access INFORMIX databases through the Powersoft ODBC interface and the INTERSOLV ODBC driver supplied with PowerBuilder for UNIX. You *cannot* access INFORMIX through the native Powersoft INFORMIX database interfaces in PowerBuilder for UNIX.

Supported versions and platforms for INFORMIX

Versions

You can access the following INFORMIX databases using the Powersoft INFORMIX database interfaces:

- INFORMIX-OnLine Versions 5.x, 6.x, and 7.x
- INFORMIX-SE Versions 5.x, 6.x, and 7.x

Platforms

PowerBuilder provides two different INFORMIX database interfaces. These interfaces require different versions of the INFORMIX client software, use different Powersoft DLLs, and are supplied on different Windows platforms.

Interface	Client software	Windows NT	Windows 95
INFORMIX IN5	I-Net v5.x	PBIN560.DLL	PBIN560.DLL
INFORMIX IN7	INFORMIX ESQL v7.2	PBIN760.DLL	PBIN760.DLL

Supported INFORMIX data types

PowerBuilder supports the following INFORMIX data types in DataWindows and embedded SQL.

Byte, text, and VarChar data types are not supported in INFORMIX SE. Float and real data types are not supported in INFORMIX Local.

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)

Data type conversion

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

FOR INFO For information about the definitions of PowerScript data types, see the *PowerScript Reference*.

INFORMIX DateTime data type

The DateTime data type 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 INFO For a list of qualifiers, see your INFORMIX documentation.

❖ **To create your own variation of the DateTime data type:**

1 In the Table painter, create a table with a DateTime column.

FOR INFO For instructions on creating a table, see the *PowerBuilder User's Guide*.

2 Click the SQL Syntax button in the PainterBar.
or
Select Design>Syntax from the menu bar.

SQL Syntax view displays pending changes to the table definition. These changes execute only when you click the Save button to save the table definition or you click the Close button and then save the changes.

3 In SQL Syntax view, click the Copy button.
or
Press CTRL+C.

The SQL syntax (or the portion you selected) is copied to the clipboard.

4 In the Database Administration painter, modify the DateTime syntax and execute the CREATE TABLE statement.

FOR INFO For instructions on using the Database Administration painter, see the *PowerBuilder User's Guide*.

INFORMIX Time data type

PowerBuilder also supports a time data type. The time data type is a subset of the DateTime data type. The time data type uses only the time qualifier boxes.

INFORMIX Interval data type

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

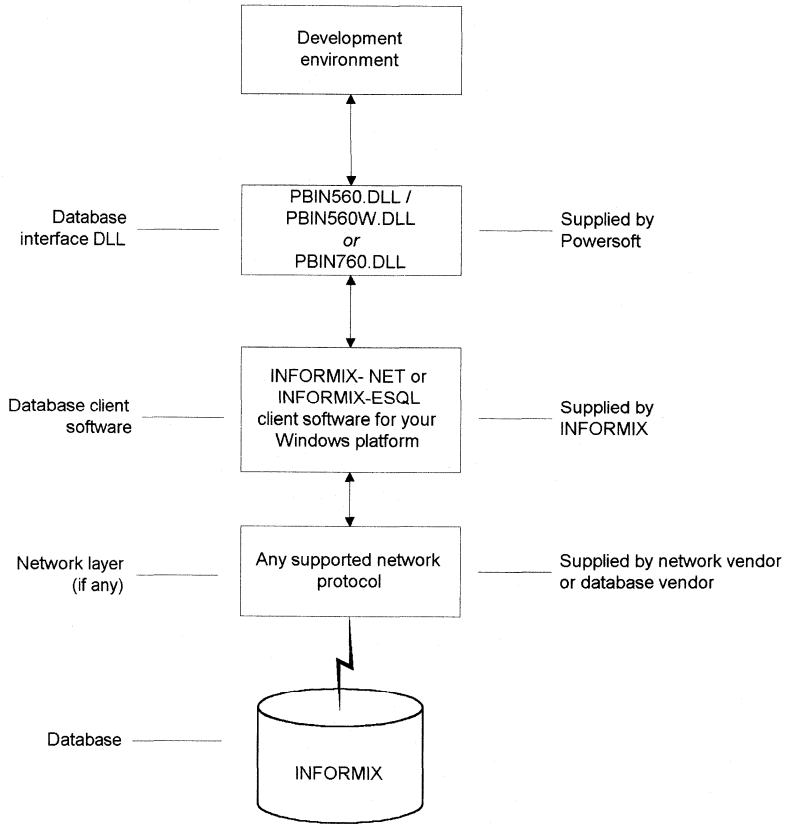
INTERVAL *largest_qualifier* **TO** *smallest_qualifier*

PowerBuilder defaults to Day(3) TO Day.

FOR INFO For more about interval data types, see your INFORMIX documentation.

Basic software components for INFORMIX

The following diagram shows the basic software components you need to access an INFORMIX database using the Powersoft INFORMIX database interfaces.



Preparing to use the INFORMIX database

What you do

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

Overview of basic steps for INFORMIX

Preparing an INFORMIX database for use with PowerBuilder involves three basic steps.

❖ **To prepare to use an INFORMIX database:**

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

2 Install the Powersoft INFORMIX IN5 or IN7 database interface.

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

Step 1: install and configure the required database server, network, and client software for INFORMIX

❖ **To install and configure the 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.

If you are using INFORMIX Version 5.x You must install database server software *and* database network software.

If you are using INFORMIX Version 6.x or higher You can install just the database server software. You need *not* install a separate database network component (such as INFORMIX-STAR or INFORMIX-NET) on the server.

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

FOR INFO For installation instructions, see your INFORMIX documentation.

2 Install the required INFORMIX-NET Version 5.x or 7.x client software on each client computer on which PowerBuilder is installed.

If you are using the IN5 interface Install INFORMIX-NET Version 5.x client software.

If you are using the IN7 interface Install INFORMIX ESQL Version 7.2 (or ESQLC for the C language).

You must obtain the INFORMIX client software from INFORMIX. 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 INFO For installation instructions, see your INFORMIX documentation.

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

For example, when you install INFORMIX-NET Version 5.x client software on your computer, 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 Powersoft INFORMIX database interface, they default to the values specified in your configuration file.

FOR INFO For instructions on setting up the INFORMIX-NET 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 Powersoft INFORMIX database interface

❖ **To install the Powersoft INFORMIX IN5 or IN7 database interface:**

- When prompted to do so by the PowerBuilder setup program, select the appropriate Powersoft INFORMIX database interface.

FOR INFO For installation instructions, see the *PowerBuilder User's Guide*.

Step 3: verify the INFORMIX connection outside PowerBuilder

❖ **To verify the INFORMIX 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 INFO 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 IN5 or Database Profile Setup - INFORMIX IN7 dialog box. You can then select this profile at any time to connect to your database in the development environment.

❖ **To create a database profile for an INFORMIX connection:**

1. Click the Database Profile button in the PowerBar.
   or
   In the Database painter, select File>Connect>Setup from the menu bar.

   The Database Profiles dialog box displays, listing your installed Powersoft database interfaces and defined database profiles.

2. Select IN5 or IN7 and click New.
   or
   Display the popup menu for IN5 or IN7 and select New.

   The Database Profile Setup - INFORMIX IN5 or Database Profile Setup - INFORMIX IN7 dialog box displays.

3 On the Connection tab, type the profile name and supply values for the basic connection parameters required by the INFORMIX database interface.

The required connection parameters include the host name, server (for INFORMIX IN7 only), user ID, password, and database. If you omit any of these values, a dialog box displays to prompt you for the missing information when you connect to the database.

FOR INFO For information about the values you should supply for these basic connection parameters, click Help.

4 (Optional) On the Connection and other tabs, supply values for any additional options (DBParm parameters and SQLCA properties) you want to set to take advantage of DBMS-specific features that the interface supports.

FOR INFO For information about additional connection parameters for the INFORMIX IN5 or IN7 interface and how to set them, click Help.

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

The Database Profiles dialog box displays, with the new profile name highlighted. The database profile values are saved in the PowerBuilder initialization file.

Specifying the server name for the INFORMIX IN7 interface

When you specify the server name value, you *must* use the following format to connect to the database through the INFORMIX IN7 interface:

host_name@server_name

Parameter	Description
host_name	The name of the host computer running the INFORMIX database server. This corresponds to the INFORMIX HOSTNAME environment variable
server_name	The name of the server containing the INFORMIX database. This corresponds to the INFORMIX SERVER environment variable

For example, to use the INFORMIX IN7 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 IN7 dialog box. PowerBuilder saves this server name as *sales@server01* in the database profile entry in your initialization file.
- **In a PowerBuilder script** Type the following in your PowerBuilder application script:

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

  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 IN7 dialog box. You can then copy the syntax from the Preview tab into your script.

Connecting to INFORMIX in a PowerBuilder script

If you are connecting to an INFORMIX database in 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 INFO For instructions on connecting to the database, see "Connecting to a database" on page 295.

Microsoft SQL Server 6.x

This section describes how to use the Microsoft SQL Server 6.x database interface in PowerBuilder.

Supported versions and platforms for SQL Server 6.x

Versions

You can access Microsoft SQL Server Version 6.0 and 6.5 databases using the Microsoft SQL Server 6.x interface. The SQL Server 6.x interface uses a Powersoft DLL named PBMSS60.DLL to access the database.

Platforms

The Microsoft SQL Server 6.x interface is available on the following operating system platforms:

- Windows NT
- Windows 95

FOR INFO For more information about supported interfaces for your product, see Appendix A, "Supported ODBC Drivers and Powersoft Database Interfaces".

Features of the SQL Server 6.x interface

The SQL Server 6.x database interface supports the following DBMS features:

- Declarative referential integrity
- Server-based cursors
- Decimal and numeric data types
- Identity columns for DataWindows, reports, and forms

The following sections briefly describe how PowerBuilder takes advantage of these features.

FOR INFO For complete information about how to use these features in SQL Server 6.x, see your Microsoft SQL Server documentation.

Declarative referential integrity

What is referential integrity? **Referential integrity** preserves the defined relationships between related tables in your database. Simply put, a DBMS enforces referential integrity to make sure that the values of a table's primary key and the foreign keys that point to it always match.

In SQL Server 6.x SQL Server 6.x supports declarative referential integrity in Transact-SQL Data Definition Language (DDL) statements (such as CREATE TABLE and ALTER TABLE) by allowing you to place constraints on an individual column (column-level constraints) or on a combination of columns (table-level constraints). This differs from earlier versions of Microsoft SQL Server that did not support declarative referential integrity in DDL syntax.

In PowerBuilder The Powersoft SQL Server 6.x database interface takes advantage of declarative referential integrity in SQL Server 6.x. When you create, alter, or drop tables, primary keys, and foreign keys in a SQL Server 6.x database, PowerBuilder generates Transact-SQL DDL statements with the correct constraint syntax to support referential integrity.

Server-based cursors

What is a cursor? Relational database management systems like SQL Server 6.x are set oriented, not row oriented. Sometimes, however, you need to update or delete individual rows within a result set. A **cursor** allows you to do this by providing a way to step through a result set one row at a time.

In SQL Server 6.x SQL Server 6.x supports ANSI SQL cursors that are server-based, which allows you to use them in server-based procedures. This differs from earlier versions of Microsoft SQL Server that provided cursors only through the DB-Library or ODBC cursor APIs.

In PowerBuilder The Powersoft SQL Server 6.x database interface supports the following types of embedded SQL FETCH statements when you retrieve a specific row from a cursor in a SQL Server 6.x database:

FETCH statement	Description
FETCH NEXT	(Default) Returns the first row of the result set if this is the first fetch against the cursor; otherwise, it moves the cursor one row within the result set
FETCH FIRST	Moves the cursor to the first row within the result set and returns the first row
FETCH PRIOR	Returns the previous row within the result set
FETCH LAST	Moves the cursor to the last row within the result set and returns the last row

FOR INFO For more information about the FETCH statement and examples of how to use it, see the *PowerScript Reference*.

Decimal and numeric data types

In SQL Server 6.x SQL Server 6.x supports the following exact numeric data types. (**Exact numeric data** is numeric data with accuracy preserved to the least significant digit.)

- Decimal
- Numeric

In PowerBuilder The Powersoft SQL Server 6.x database interface supports both of these data types in DataWindow objects and embedded SQL.

Identity columns

In SQL Server 6.x Columns that have the IDENTITY property contain system-generated values that uniquely identify each row within a table. This feature lets you automatically generate sequential numbers, such as employee identification numbers.

When you insert values into a table containing an identity column, SQL Server 6.x automatically generates the next column identifier based on the last identity value (incremented by adding rows) and the increment value specified when you created the column.

In PowerBuilder The Powersoft SQL Server 6.x database interface supports identity columns for DataWindows.

Supported SQL Server 6.x data types

PowerBuilder supports these Microsoft SQL Server 6.x data types:

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

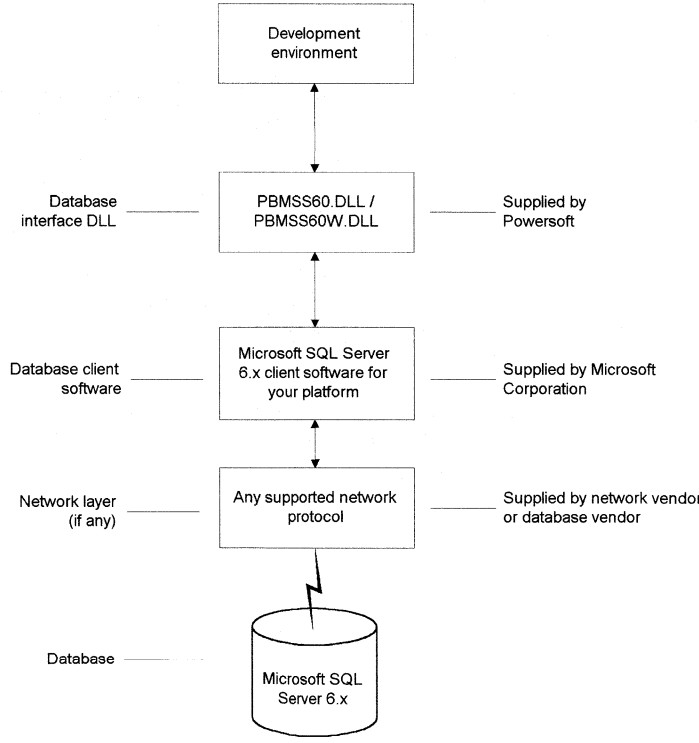
Data type conversion

When you retrieve or update columns, PowerBuilder converts data appropriately between the Microsoft SQL Server 6.x data type and the PowerScript data type. Keep in mind, however, that similarly or identically named SQL Server 6.x and PowerScript data types do *not* necessarily have the same definitions.

FOR INFO For information about the definitions of PowerScript data types, see the *PowerScript Reference*.

Basic software components for SQL Server 6.x

You must install the following software components to access a database with the Microsoft SQL Server 6.x interface:



Preparing to use the SQL Server 6.x database

What you do

Before you define the database interface and connect to a Microsoft SQL Server 6.x database in PowerBuilder, follow these steps to prepare the database for use.

Overview of basic steps for SQL Server 6.x

Preparing a Microsoft SQL Server 6.x database for use with PowerBuilder involves three basic steps.

❖ **To prepare to use a Microsoft SQL Server 6.x database:**

1. Install and configure the required database server, network, and client software.
2. Install the Powersoft SQL Server 6.x database interface.
3. Verify that you can connect to the Microsoft SQL Server 6.x server and database outside PowerBuilder.

Installing Powersoft stored procedures is not required

Unlike other Powersoft SQL Server database interfaces, the Powersoft SQL Server 6.x database interface does *not* require you to install certain Powersoft stored procedures in the master database before connecting to SQL Server 6.x in PowerBuilder.

The Powersoft SQL Server 6.x database interface uses SQL Server 6.x catalog stored procedures to get information about tables and columns from the SQL Server catalog. It does not need the Powersoft stored procedures to do this.

Therefore, the information in "Installing Powersoft stored procedures in SQL Server databases" on page 274 *does not apply* when you are using the Powersoft SQL Server 6.x database interface.

Step 1: install and configure the database server, network, and client software for SQL Server 6.x

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

1 Make sure the Microsoft SQL Server 6.x 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 INFO For installation instructions, see your Microsoft SQL Server 6.x documentation.

Upgrading from an earlier version of SQL Server

For instructions on upgrading to SQL Server 6.x from an earlier 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 6.x database, make sure the required network software (for example, TCP/IP) is installed and running on your computer and is properly configured so you can connect to the SQL Server 6.x database server at your site.

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

3 Install the required Microsoft SQL Server 6.x client software on each client computer on which PowerBuilder is installed.

You must obtain the SQL Server 6.x 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 INFO For installation instructions, see your Microsoft SQL Server documentation.

4 Make sure the SQL Server 6.x client software is properly configured so you can connect to the SQL Server 6.x database server at your site.

Once you install the SQL Server 6.x client software, you can configure optional client connection parameters by using the SQL Server Client Configuration Utility that comes with the client software.

FOR INFO For configuration instructions, see your Microsoft SQL Server documentation.

5 If required by your operating system, make sure the directory containing the SQL Server 6.x client software is in your system path.

6 Make sure only one copy of each of the following files is installed on your computer:

- PBMSS050.DLL (Powersoft SQL Server 6.x interface DLL)
- MSDBLIB3.DLL (16-bit SQL Server 6.x DLL)

  or

  NTWDBLIB.DLL (32-bit SQL Server 6.x DLL)

Step 2: install the Powersoft SQL Server 6.x database interface

❖ **To install the Powersoft SQL Server 6.x database interface:**

- When prompted to do so by the PowerBuilder setup program, select the Powersoft SQL Server 6.x database interface.

  FOR INFO For installation instructions, see the *PowerBuilder Installation Guide*.

Step 3: verify the SQL Server 6.x connection outside PowerBuilder:

❖ **To verify the SQL Server 6.x connection:**

- Make sure you can connect to the SQL Server 6.x 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 6.x database interface

To define a connection through the Microsoft SQL Server 6.x 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 6.x dialog box. You can then select this profile at any time to connect to your database in the development environment.

❖ **To create a database profile for a Microsoft SQL Server 6.x connection:**

1 Click the Database Profile button in the PowerBar.
or
In the Database painter, select File>Connect>Setup from the menu bar.

The Database Profiles dialog box displays, listing your installed Powersoft database interfaces and defined database profiles.

2 Select MSS and click New.
or
Display the popup menu for MSS and select New.

The Database Profile Setup - Microsoft SQL Server 6.x dialog box displays.

3 On the Connection tab, type the profile name and supply values for the basic connection parameters required by the Microsoft SQL Server 6.x database interface.

The required connection parameters include the server, login ID, password, and database. If you omit any of these values, a dialog box displays to prompt you for the missing information when you connect to the database.

FOR INFO For information about the values you should supply for these basic connection parameters, click Help.

4 (Optional) On the Connection and other tabs, supply values for any additional options (DBParm parameters and SQLCA properties) you want to set to take advantage of DBMS-specific features that the interface supports.

FOR INFO For information about additional connection parameters for the Microsoft SQL Server 6.x interface and how to set them, click Help.

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

The Database Profiles dialog box displays, with the new profile name highlighted. The database profile values are saved in the PowerBuilder initialization file.

What to do next

FOR INFO For instructions on connecting to the database, see "Connecting to a database" on page 295.

Oracle

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

Supported versions and platforms for Oracle

PowerBuilder provides several different Oracle database interfaces. These interfaces use different Powersoft DLLs and shared libraries, access different versions of Oracle, and are supplied on different operating system platforms.

For more information
As additional Powersoft database interfaces are supported in PowerBuilder, updated information will be available electronically from Powersoft services on CompuServe, FTP, BBS, and the World Wide Web.

Oracle interface	Windows NT and Windows 95	Power Macintosh	Solaris	HP-UX	AIX
OR7 Oracle Version 7.0	—	OR7 Oracle 7.0 Driver	—	—	—
O71 Oracle Version 7.1	PBO7160.DLL	O71 Oracle 7.1 Driver	—	—	—
O72 Oracle Version 7.2	PBO7260.DLL	—	libpbo7260.so	—	—
O73 Oracle Version 7.3	PBO7360.DLL	O73 Oracle 7.3 Driver	libpbo7360.so	libpbo7360.sl	libpbo7360.a

Supported Oracle data types

PowerBuilder supports the following Oracle data types in DataWindow objects and embedded SQL:

Char	Number
Date	Raw
Float	VarChar
Long	VarChar2
LongRaw	

Data type conversion

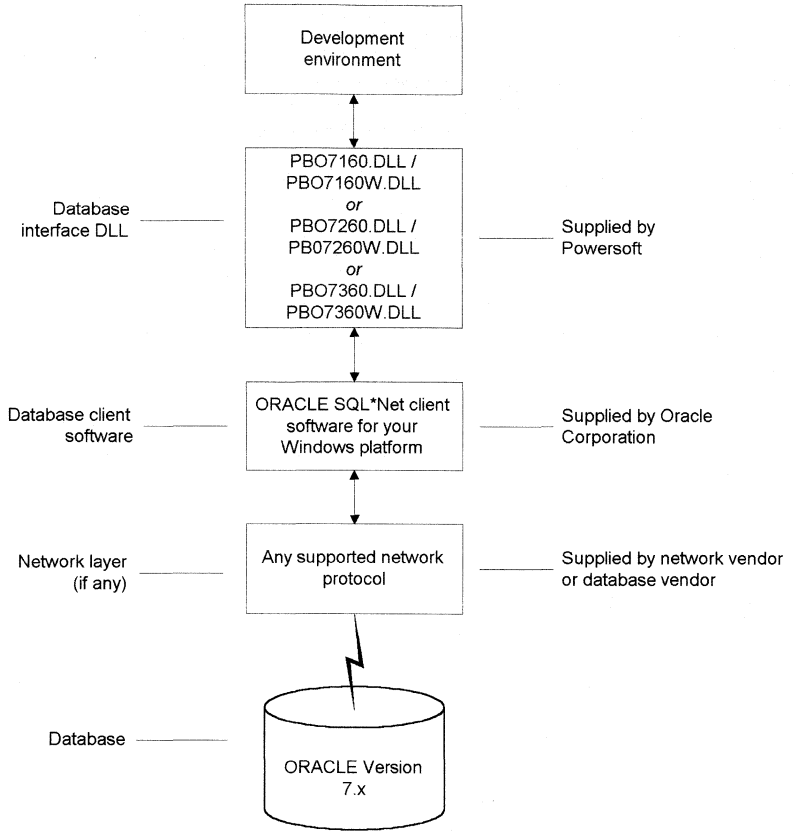
When you retrieve or update columns, PowerBuilder converts data appropriately between the Oracle data type and the PowerScript data type. Keep in mind, however, that similarly or identically named Oracle and PowerScript data types do *not* necessarily have the same definitions.

FOR INFO For information about the definitions of PowerScript data types, see the *PowerScript Reference*.

Basic software components for Oracle

The following diagrams show the basic software components you need to access an Oracle database in PowerBuilder.

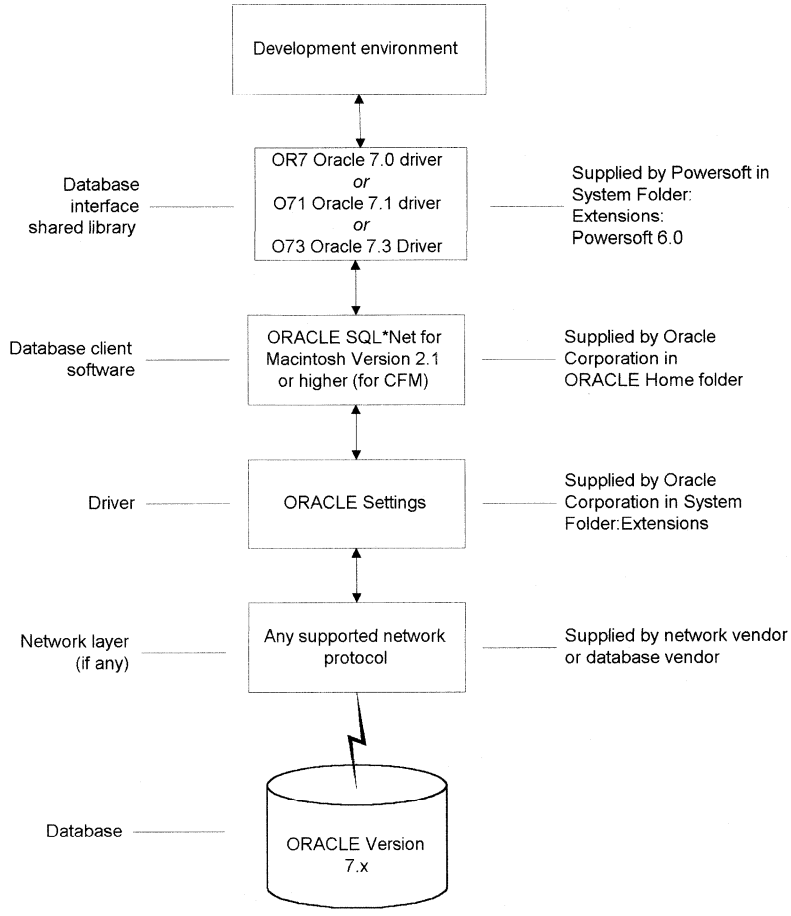
Accessing Oracle on Windows



What to do next

FOR INFO For instructions on preparing your Oracle database for use with PowerBuilder on the Windows platform, see "Preparing to use the Oracle database on Windows" on page 177.

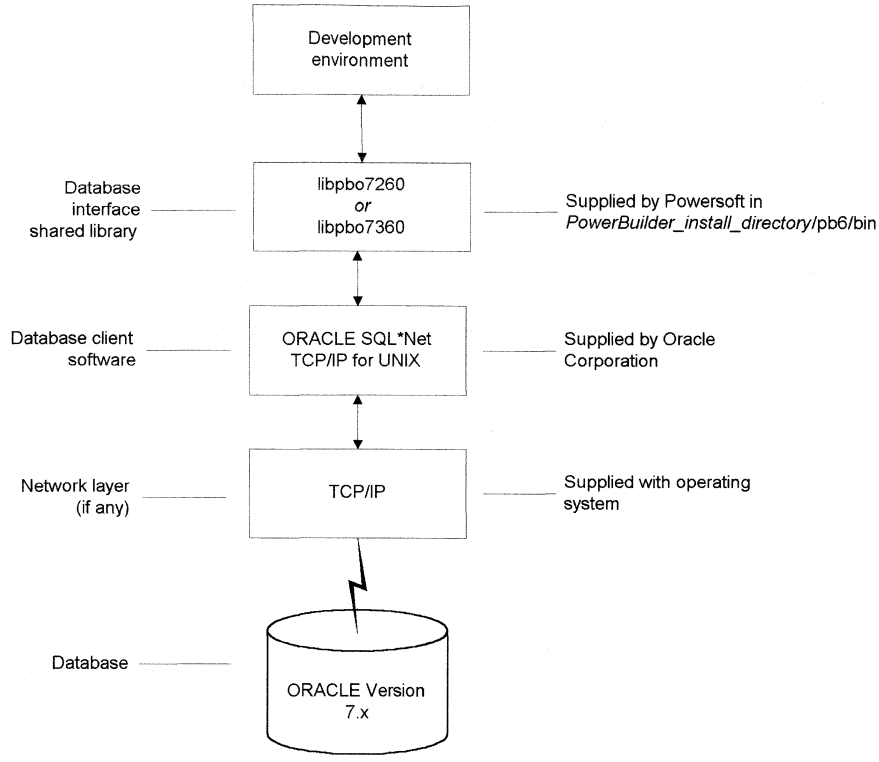
Accessing Oracle on Macintosh



What to do next

FOR INFO For instructions on preparing your Oracle database for use with PowerBuilder on the Macintosh platform, see "Preparing to use the Oracle database on Macintosh" on page 180.

Accessing Oracle on UNIX



What to do next

FOR INFO For instructions on preparing your Oracle database for use with PowerBuilder on the UNIX platform, see "Preparing to use the Oracle database on UNIX" on page 183.

Preparing to use the Oracle database on Windows

What you do

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

Overview of basic steps for Oracle

Preparing an Oracle database for use with PowerBuilder involves the same three basic tasks on all supported platforms. However, some of the steps for performing the tasks are different on each platform.

❖ **To prepare to use an Oracle database:**

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

2 Install the Powersoft Oracle database interface for the version of Oracle you want to access.

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

Step 1: install and configure the database server, network, and client software

❖ **To install and configure the database server, network, and client software for Oracle on Windows:**

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

You must obtain the database server software from Oracle Corporation.

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

3 Install the required Oracle SQL*Net client software on each client computer on which PowerBuilder is installed.

You must obtain the SQL*Net client software from Oracle Corporation. Make sure the version of SQL*Net 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 for Oracle 7.3
To use the Oracle 7.3 (O73) interface on Windows, you must install Oracle SQL*Net client software Version 2.3 or higher.

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

Installing the 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 on Windows, 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 INFO For information about setting up the TNSNAMES.ORA or other required Oracle configuration file, see your SQL*Net documentation.

5 If required by your operating system, make sure the directory containing the SQL*Net software is in your system path.

Step 2: install the Powersoft Oracle database interface

❖ **To install the Powersoft Oracle database interface for your version of Oracle:**

- When prompted to do so by the PowerBuilder setup program, select the appropriate Powersoft Oracle database interface for the version of Oracle you want to access.

  FOR INFO For a list of the Powersoft Oracle database interfaces available on each platform, see "Supported versions and platforms for Oracle" on page 173.

  FOR INFO For installation instructions, see the *PowerBuilder Installation Guide*.

Step 3: verify the Oracle connection outside PowerBuilder

❖ **To verify the Oracle connection on Windows:**

- Make sure you can connect to the Oracle database server and log on 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 on 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 INFO For instructions on defining the Oracle database interface in PowerBuilder, see "Defining the Oracle database interface" on page 187.

Preparing to use the Oracle database on Macintosh

What you do

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

Overview of basic steps for Oracle

Preparing an Oracle database for use with PowerBuilder involves the same three basic tasks on all supported platforms. However, some of the steps for performing the tasks are different on each platform.

❖ **To prepare to use an Oracle database:**

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

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

Step 1: install and configure the database server, network, and client software

❖ **To install and configure the database server, network, and client software for Oracle on Macintosh:**

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

You must obtain the database server software from Oracle Corporation.

FOR INFO For installation instructions, see your Oracle documentation.

2 Make sure the required TCP/IP network software is installed and running on your Macintosh and is properly configured for your environment.

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

If you are using MacTCP network software, make sure that:

- **You've used the MacTCP control panel** You or your system administrator must use the MacTCP control panel to set your Macintosh's IP address and (if necessary) gateway IP address. After setting the address, you may need to restart your Macintosh so it recognizes the address.
- **The Hosts and Services files are present** The Hosts and Services files must be present on your Macintosh and properly configured for your environment.

FOR INFO For MacTCP installation and configuration instructions, see your network administrator.

3 Install the required Oracle SQL*Net for Macintosh client software on each client computer on which PowerBuilder is installed.

To connect to an Oracle database in PowerBuilder on Macintosh, you must install Oracle SQL*Net for Macintosh Version 2.1 or higher. This version of SQL*Net supports Apple Code Fragment Manager (CFM).

You must obtain the SQL*Net client software from Oracle Corporation.

Required client software for Oracle 7.3
To use the Oracle 7.3 (O73) interface on Macintosh, you must install Oracle SQL*Net client software Version 2.3 or higher.

4 Make sure the config.ora configuration file is properly set up for your environment.

The Oracle configuration file config.ora is a text file used to connect to an Oracle database. The Oracle installation automatically creates config.ora and places it in the Oracle Home folder on either the database server or client workstation. If necessary, you can edit config.ora to customize your Oracle database connection.

FOR INFO For instructions on editing config.ora, see your Oracle documentation.

5 Make sure that Oracle Home is set to the proper folder.

The Oracle Home folder contains all the required client software for connecting to Oracle. Typically, the Oracle installation program automatically specifies the Oracle Home folder. However, if you move this folder after installation, you must reset Oracle Home to the correct folder before you can connect to the database. To set Oracle Home, you use the Set Oracle Home application.

FOR INFO For detailed instructions on using the Set Oracle Home application, see your Oracle documentation.

These steps summarize this procedure:

1 Open the Applications folder in your Oracle folder.

2 Double-click the Set Oracle Home icon to start it.

3 In the listbox, browse to and select the folder you want to set as Oracle Home.

4 Click the Set Home button to set Oracle Home to the selected folder.

Step 2: install the Powersoft Oracle database interface

❖ **To install the Powersoft Oracle database interface for your version of Oracle:**

♦ When prompted to do so by the PowerBuilder setup program, select the appropriate Powersoft Oracle database interface for the version of Oracle you want to access.

FOR INFO For a list of the Powersoft Oracle database interfaces available on each platform, see "Supported versions and platforms for Oracle" on page 173.

FOR INFO For installation instructions, see the *PowerBuilder for Macintosh Installation Guide*.

Step 3: verify the Oracle connection outside PowerBuilder

❖ **To verify the Oracle connection on Macintosh:**

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

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

  - **Accessing the database server** The MacTCP Ping utility checks that you can reach the database server from your Macintosh.
  - **Accessing the database** The Oracle Server Manager utility checks that you can access the database from your Macintosh. It lets you log on to an Oracle database and perform database operations.

  FOR INFO For instructions on using MacTCP Ping and the Oracle Server Manager, see your Oracle documentation.

What to do next

FOR INFO For instructions on defining the Oracle database interface in PowerBuilder, see "Defining the Oracle database interface" on page 187.

Preparing to use the Oracle database on UNIX

What you do

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

Overview of basic steps for Oracle

Preparing an Oracle database for use with PowerBuilder involves the same three basic tasks on all supported platforms. However, some of the steps for performing the tasks are different on each platform.

❖ **To prepare to use an Oracle database:**

1. Install and configure the required database server, network, and client software.
2. Install the Powersoft Oracle database interface.
3. Verify that you can connect to the Oracle server and database outside PowerBuilder.

Step 1: install and configure the database server, network, and client software

❖ **To install and configure the database server, network, and client software for Oracle on UNIX:**

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

You must obtain the database server software from Oracle Corporation.

FOR INFO For installation instructions, see your Oracle documentation.

2 Make sure the required TCP/IP network software is installed and running on your computer and is properly configured for your environment.

The TCP/IP software is supplied as part of the UNIX operating system.

3 Install the required Oracle SQL*Net TCP/IP client software on each client computer on which PowerBuilder for UNIX is installed.

To connect to an Oracle database in PowerBuilder for UNIX, you must install Oracle SQL*Net TCP/IP client software Version 1.x or 2.x.

You must obtain the SQL*Net client software from Oracle Corporation.

FOR INFO For SQL*Net installation and configuration instructions, see your Oracle documentation.

Required client software for Oracle 7.3
To use the Oracle 7.3 (O73) interface on UNIX, you must install Oracle SQL*Net client software Version 2.3 or higher (on Solaris and AIX) or Version 2.3.3 or higher (on HP-UX).

---

4 Make sure the Oracle_HOME environment variable is defined in your shell initialization file.

The Oracle_HOME environment variable points to the directory where the Oracle SQL*Net client software is installed.

Here is a sample Oracle_HOME definition for a C shell initialization file:

```
setenv ORACLE_HOME /export/home/oracle
```

5 Append $ORACLE_HOME/bin to the PATH environment variable in your shell initialization file.

Here is the PATH definition for a C shell initialization file:

```
setenv PATH ${ORACLE_HOME}/bin:${PATH}
```

6 Append $ORACLE_HOME/lib to the library path environment variable in your shell initialization file.

Here is the LD_LIBRARY_PATH definition for a C shell initialization file on Solaris:

```
setenv LD_LIBRARY_PATH ${ORACLE_HOME}/lib:
${LD_LIBRARY_PATH}
```

7 Make sure the PBHOME environment variable is defined in your shell initialization file.

The PBHOME environment variable points to the directory where PowerBuilder for UNIX is installed (for example, /export/home/pb6).

Here is a sample PBHOME definition for a C shell initialization file:

```
setenv PBHOME /export/home/pb6
```

Step 2: install the Powersoft Oracle database interface

❖ **To install the Powersoft Oracle database interface:**

1 When prompted to do so by the PowerBuilder setup program, select the Powersoft Oracle database interface.

FOR INFO For a list of the Powersoft Oracle database interfaces available on each platform, see "Supported versions and platforms for Oracle" on page 173.

FOR INFO For installation instructions, see the *PowerBuilder Installation Guide (UNIX)*.

2 (Optional) Use the **which ldd** command to make sure the **ldd** command is in your path.

Typically the ldd command is in the /usr/bin directory.

3 (Optional) Issue an ldd command on the Powersoft Oracle interface shared library you are using to make sure it can locate the appropriate shared libraries. For example, on Solaris, type:

```
ldd $PBHOME/bin/libpbo7360.so
```

You should see output similar to the following:

```
libclntsh.so.1.0  => /export/home/oracle/lib/libclntsh.so.1.0
libsocket.so.1    => /usr/lib/libsocket.so.1
libnsl.so.1       => /usr/lib/libnsl.so.1
libm.so.1         => /opt/tools/SUNWspro/lib/libm.so.1
libdl.so.1        => /usr/lib/libdl.so.1
libaio.so.1       => /usr/lib/libaio.so.1
```

```
libc.so.1          => /usr/lib/libc.so.1
libintl.so.1       => /usr/lib/libintl.so.1
libmp.so.1         => /usr/lib/libmp.so.1
libw.so.1          => /usr/lib/libw.so.1
```

All of these libraries must be present to connect to Oracle in PowerBuilder for UNIX.

Step 3: verify the Oracle connection outside PowerBuilder

❖ **To verify the Oracle connection on UNIX:**

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

  Tools such as Oracle SQL*Plus check whether you can log on 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.

  FOR INFO For instructions on using SQL*Plus, see your Oracle documentation. The following are sample sqlplus commands for connecting to your database.

SQL*Net 2.x sqlplus example

If you are using Oracle SQL*Net Version 2.x to access the database, the sqlplus command syntax is:

sqlplus *user_ID/ password@* **tns:** *ORACLEServiceName*

Assume you are using the following connection parameters in PowerBuilder for UNIX to access an Oracle Version 7.3 database:

Parameter	Value
user_ID	fran
password	abc
ORACLEServiceName (from the TNSNAMES.ORA configuration file)	ora73unix

To verify your connection to the database outside PowerBuilder for UNIX, type the following sqlplus command:

```
sqlplus fran/abc@tns:ora73unix
```

If the connection succeeds, the SQL prompt (SQL>) will display.

SQL*Net 1.x sqlplus example

If you are using Oracle SQL*Net Version 1.x to access the database, the sqlplus command syntax is:

sqlplus *user_ID/ password@* **t** : *server_name* : *database_name*

Assume you are using the following connection parameters in PowerBuilder for UNIX to access an Oracle Version 7.2 database:

Parameter	Value
user_ID	bob
password	xyz
server_name	oracle
database_name	testdb

To verify your connection to the database outside PowerBuilder for UNIX, type the following sqlplus command:

```
sqlplus bob/xyz@t:oracle:testdb
```

If the connection succeeds, the SQL prompt (SQL>) will display.

What to do next

FOR INFO For instructions on defining the Oracle database interface in PowerBuilder, see "Defining the Oracle database interface" next.

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.

❖ **To create a database profile for an Oracle connection:**

1 Click the Database Profile button in the PowerBar.
or
In the Database painter, select File>Connect>Setup from the menu bar.

The Database Profiles dialog box displays, listing your installed Powersoft database interfaces and defined database profiles.

2 Select one of the Oracle interfaces and click New.
or
Display the popup menu for one of the Oracle interfaces and select New.

The Database Profile Setup dialog box for your Oracle interface displays. For example, if you select the O73 interface, the Database Profile Setup - Oracle 7.3 dialog box displays.

3 On the Connection tab, type the profile name and supply values for the basic connection parameters required by your Oracle database interface.

The required connection parameters include the server, login ID, and password. If you omit any of these values, a dialog box displays to prompt you for the missing information when you connect to the database.

FOR INFO For information about the values you should supply for these basic connection parameters, click Help.

4 (Optional) On the Connection and other tabs, supply values for any additional options (DBParm parameters and SQLCA properties) you want to set to take advantage of DBMS-specific features that the interface supports.

FOR INFO For information about additional connection parameters for your Oracle interface and how to set them, click Help.

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

The Database Profiles dialog box displays, with the new profile name highlighted. The database profile values are saved in the PowerBuilder initialization file.

Specifying the Oracle server connect string

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

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

Using a connect descriptor or connect string

The version of the SQL*Net client software you are using (on any platform) determines whether you should specify an Oracle connect descriptor or connect string in the Server box.

If you are using	Specify this in Server box
SQL*Net Version 2.x	Oracle connect descriptor
SQL*Net Version 1.x	Oracle connect string
SQL*Net for DOS	Oracle connect string

Specifying a connect descriptor

If you are using SQL*Net Version 2.x, you must specify an Oracle connect descriptor in the Server box. The syntax is:

@ **TNS:** *ORACLEServiceName*

Parameter	Description
@	The at (@) sign is required
TNS	The identifier for the Oracle Transparent Network Substrate (TNS) technology, on which SQL*Net Version 2.x is based
:	The colon (:) is required
ORACLEServiceName	The service name assigned to your server in the Oracle configuration file for your platform. For example, on Windows, TNSNAMES.ORA is the required configuration files for SQL*Net Version 2.x FOR INFO For instructions on configuring SQL*Net Version 2.x, see your Oracle SQL*Net documentation

SQL*Net Version 2.x example To use SQL*Net Version 2.x client software to connect to the service named ORA73, type the following connect descriptor in the Server box on the Connection tab of the Database Profile Setup - Oracle 7.3 dialog box:

@TNS:ORA73

Specifying a connect string

The syntax of the connect string depends on the Oracle client software you are using: SQL*Net Version 1.x or SQL*Net for DOS.

If you are using SQL*Net Version 1.x, the syntax is:

@ *identifier* : *server_name* : *database_name*

If you are using SQL*Net for DOS, the syntax is:

@ *identifier* : *server_name*

Parameter	Description
@	The at (@) sign is required.
identifier	The appropriate SQL*Net communications identifier for your network protocol and driver. Examples of supported network protocols and drivers on the Windows platform are: **B** NetBios, SQLNTB driver **D** DECnet, SQLDNT driver **P** Named Pipes, SQLNMP driver **T** TCP/IP, SQLTCP driver **V** Vines, SQLVIN driver **X** Novell, SQLSPX driver **You must use T on Macintosh and UNIX** When defining an Oracle database interface in PowerBuilder on the Macintosh or UNIX platform, you must specify the **T** network identifier to represent the TCP/IP network protocol. FOR INFO For help determining the required network protocol, driver, and identifier for your site, see your network or database administrator
:	The colon (:) is required
server_name	The name or IP address assigned to the database server
database_name	The name assigned to the Oracle database (instance) you are logging on to

SQL*Net Version 1.x example To use TCP/IP with SQL*Net Version 1.x client software to connect to an Oracle database named TESTDB on an Oracle server named TIGER, type the following connect string in the Server box on the Connection tab of the Database Profile Setup dialog box:

```
@T:TIGER:TESTDB
```

SQL*Net for DOS example To use NetBios with SQL*Net for DOS client software to connect to an Oracle server named LION, type the following connect string in the Server box on the Connection tab of the Database Profile Setup dialog box:

```
@B:LION
```

Using Oracle stored procedures as a data source

This section describes how you can use Oracle stored procedures in your PowerBuilder application.

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 (in Oracle 7.2 or higher) 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 INFO For information about the syntax for using the DECLARE PROCEDURE statement with the RPCFUNC keyword, see the *PowerScript Reference*.

Kinds of Oracle stored procedures

PowerBuilder lets you use two kinds of Oracle stored procedures in your application, depending on the version of your Oracle database server and the PowerBuilder platform you are using.

If your database server is	You can use
Oracle Version 7.2 or higher	An Oracle stored procedure that has a result set as an IN OUT (reference) parameter **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
Oracle Version 7.x	An Oracle stored procedure that uses PBDBMS.Put_Line function calls to build the SQL SELECT statement that will return results to DataWindows or reports

Platform differences

The kind of Oracle stored procedures you can use in PowerBuilder applications depends on the platform you are using:

Oracle stored procedure	Windows	Macintosh	UNIX
With result sets	x	—	x
With PBDBMS.Put_Line calls	x	x	x

Using Oracle stored procedures with result sets

Overview of basic steps

The following procedure assumes you are creating the stored procedure in the PowerBuilder Database Administration painter.

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

1. Set up the Database Administration 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 and reports that use the stored procedure as a data source.

Setting up the Database Administration painter

When you create a stored procedure in the Database Administration 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 Administration 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.

Follow these steps to change the default SQL terminator character in the Database Administration painter.

❖ **To set up the Database Administration painter to create the stored procedure:**

1 Using the Powersoft O72 or higher database interface, connect to your Oracle database in PowerBuilder as the System user.

You can use Oracle stored procedures with result sets in your application *only* if you are accessing the database with the Powersoft O72, O73, or higher database interface. This feature is not supported in any Powersoft Oracle database interface earlier than O72.

FOR INFO For instructions, see "Defining the Oracle database interface" on page 187.

2 Open the Database painter.

3 Click the Database Preferences button in the PainterBar.
or
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.

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).

Creating the stored procedure

After setting up the Database Administration 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 INFO 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 191.

❖ **One way to create Oracle stored procedures with result sets:**

1 Make sure your Oracle user account has the necessary database access and privileges to 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 *rctl* and a stored procedure named *proc1*. The tt%ROWTYPE attribute defines *rctl* 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 *rctl*.

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

4 Create the Oracle stored procedure as part of 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* that 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 Set the PBDBMS DBParm parameter to 0 in one of the following ways to enable use of an Oracle 7.2 stored procedure with a result set as a data source.

- **Database profile** Clear the PBDBMS checkbox on the Connection tab in the Database Profile Setup dialog box for your Oracle connection.

- **PowerBuilder application script** Type the following in your script:

```
SQLCA.DBParm = "PBDBMS = 0"
```

FOR INFO For more about the PBDBMS DBParm parameter, see its description in Chapter 7, "DBParm Parameters".

2 In the DataWindow painter, click the New button in the Select dialog box.

The New DataWindow dialog box displays. The Stored Procedure icon displays as a data source.

3 Select the Stored Procedure data source, choose a presentation style, and click OK.

The Select Stored Procedure dialog box displays, listing the stored procedures available in your database.

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

5 Define your DataWindow object.

You call a DataWindow Retrieve function as you normally do to get the data. PowerBuilder fetches the result set from the cursor in order to populate the DataWindow object.

FOR INFO For instructions on defining DataWindow objects, see the *PowerBuilder User's Guide*.

Using Oracle stored procedures with PBDBMS.Put_Line calls

Overview of basic steps

The following procedure assumes you are creating the stored procedure in the PowerBuilder Database Administration painter.

❖ **To use an Oracle stored procedure with PBDBMS.Put_Line calls:**

1 Set up the Database Administration painter to create the stored procedure.

2 Install the PBDBMS package on your Oracle 7.x database server.

3 Create the stored procedure using PBDBMS.Put_Line function calls.

4 Create DataWindow object that uses the stored procedure as a data source.

Setting up the Database Administration painter

When you create a stored procedure in the Database Administration 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 Administration 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.

Follow these steps to change the default SQL terminator character in the Database Administration painter.

❖ **To set up the Database Administration painter to create the stored procedure:**

1 Connect to your Oracle database in PowerBuilder as the System user.

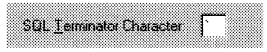
You can use any of the Powersoft Oracle database interfaces to access the database.

2 Open the Database painter.

3 Click the Database Preferences button in the PainterBar.
or
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.

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).

Installing the PBDBMS package on the server

You must run the PBOR7CAT.SQL script to install the PBDBMS package on the Oracle 7.x database server. The PBDBMS package enables you to use Oracle stored procedures with PBDBMS.Put_Line function calls as data sources for DataWindow objects.

❖ **To install the PBDBMS package on the Oracle 7.x database server:**

- In the Database Administration painter, open and execute the PBOR7CAT.SQL script.

  The location of the PBOR7CAT.SQL file depends on the PowerBuilder platform you are using.

Platform	Location	Comments
Windows	\SERVER directory on the PowerBuilder CD-ROM	The \SERVER directory contains server-side installation components. Its contents is *not installed* with PowerBuilder on your computer
Macintosh	Database Extras folder on the last disk of the Deployment Kit	The Database Extras folder is *not installed* with PowerBuilder on your computer
UNIX	pb6/bin directory in the PowerBuilder install directory	The pbor7cat.sql file is automatically installed if you install the Oracle 7.2 or 7.3 database interface

  You can access the PBOR7CAT.SQL file from your computer's CD-ROM drive or copy it to your computer.

  FOR INFO For instructions on executing SQL in the Database Administration painter, see the *PowerBuilder User's Guide*.

Creating the stored procedure

❖ **To create the Oracle stored procedure with PBDBMS.Put_Line calls:**

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

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

2. Create the Oracle stored procedure as you normally do. The only difference is that you must code PBDBMS.Put_Line function calls to build the SQL SELECT statement.

   The PBDBMS.Put_Line function takes one parameter, a string of up to 255 characters. The SELECT statement is a concatenation of the parameters passed to the Put_Line function. You can call Put_Line repeatedly in the procedure (up to 100 times) until you have built the entire SELECT statement.

Prohibited SQL syntax with PBDBMS.Put_Line calls
You cannot use the following SQL syntax when building SELECT statements with PBDBMS.Put_Line calls. Instead, you must specify the column names.

```
// This SQL syntax produces incorrect results.
PBDBMS.Put_Line('SELECT * FROM table_name')
```

Example The following Oracle 7.x stored procedure, usp_address_list, is built using PBDBMS.Put_Line function calls.

```
CREATE PROCEDURE usp_address_list
    (starting_id in number, ending_id in number)
  AS
  BEGIN
     PBDBMS.Put_Line('SELECT cust_fname || ');
     PBDBMS.Put_Line('lpad(rpad
             (cust_lname,20),21),');
     PBDBMS.Put_Line('cust_street,cust_city,
             cust_state,');
     PBDBMS.Put_Line('cust_zipcode ');
     PBDBMS.Put_Line('FROM customer WHERE cust_id
             BETWEEN ');
     PBDBMS.Put_Line(starting_id);
     PBDBMS.Put_Line(' AND ');
     PBDBMS.Put_Line(ending_id);
     PBDBMS.Put_Line(' ORDER BY cust_zipcode
             ASC');END;`
```

The SQL SELECT statement that will be generated by the usp_address_list stored procedure is:

```
SELECT cust_fname,lpad(rpad
      (cust_lname,20),21),cust_street,
      cust_city,cust_state,cust_zipcode
FROM customer
WHERE cust_id BETWEEN starting_id AND ending_id
ORDER BY cust_zipcode ASC
```

A DataWindow object using this procedure will have two retrieval arguments: starting_id and ending_id.

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 Oracle stored procedure as a data source.

❖ **To create a DataWindow object using a stored procedure with PBDBMS.Put_Line calls:**

1 Make sure the PBDBMS DBParm parameter is set to 1 (the default) to enable use of Oracle stored procedures with PBDBMS.Put_Line calls as a data source.

Because 1 is the default value for PBDBMS, you should not have to set it in your database profile or PowerBuilder application script. If necessary, however, you can set it as follows.:

- **Database profile** Select the PBDBMS checkbox on the Connection tab in the Database Profile Setup dialog box for your Oracle connection.
- **PowerBuilder application script** Type the following in your script:

```
SQLCA.DBParm = "PBDBMS = 1"
```

FOR INFO For more about the PBDBMS DBParm parameter, see PBDBMS on page 496.

2 n the DataWindow painter, click the New button in the Select dialog box.

The New DataWindow dialog box displays. The Stored Procedure icon displays as a data source.

3 Select the Stored Procedure data source, choose a presentation style, and click OK.

The Select Stored Procedure dialog box displays, listing the stored procedures available in your database.

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

5 Define your DataWindow object.

You call a DataWindow Retrieve function as you normally do to get the data. The SELECT statement is generated by the specified stored procedure and accessed internally by the DataWindow object through the PBDBMS package on the database server.

FOR INFO For instructions on defining DataWindow objects, see the *PowerBuilder User's Guide*.

Limitations

Using an Oracle stored procedure with PBDBMS.Put_Line calls as a DataWindow object data source has the following limitations:

- The stored procedure must have *no* output parameters.
- The SELECT statement is limited to 255 characters * 100, or 25,500 characters.

What to do next

FOR INFO For instructions on connecting to the database, see "Connecting to a database" on page 295.

SQL Server Version 4.x

This section describes how to use the Powersoft SQL Server Version 4.x database interface in PowerBuilder.

DB-Library API

The Powersoft SQL Server 4.x database interface uses the DB-Library (DB-Lib) application programming interface (API) to access the database.

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

If you want to use the Sybase Client Library (CT-Lib) API to access a Sybase System 10 or System 11 database, you must install the Powersoft Sybase SQL Server System 10 and System 11 database interface, as described on page 240.

Supported versions and platforms for SQL Server 4.x

SQL Server 4.x versions

PowerBuilder provides *two* different SQL Server Version 4.x database interfaces. These interfaces both use the DB-Lib client API, but they access SQL Server 4.x from different vendors and are supplied on different platforms:

Powersoft DBMS identifier	Accesses this version of SQL Server	Win 95	Win NT	Mac	UNIX
SYB	Sybase SQL Server Version 4.x	—	—	x	x
	Microsoft SQL Server Version 4.x	x	x	x	x
SYT	Sybase SQL Server DB-Lib	x	x	—	—

Accessing Microsoft SQL Server 4.x on Macintosh and UNIX

To access a Microsoft SQL Server Version 4.x database in PowerBuilder on the Macintosh or UNIX platform, you *must* obtain the required Open Client software from Sybase, Inc. The Sybase Open Client software is *not* included when you purchase the Microsoft SQL Server 4.x database software for these platforms.

Powersoft interface DLL or shared library names

The name of the DLL (on Windows) or shared library (on Macintosh and UNIX) that PowerBuilder uses to access SQL Server 4.x depends on the platform you are using:

Powersoft database interface	Win 95	Win NT	Mac	UNIX
SYB SQL Server 4.x	PBSYB60.DLL	PBSYB60.DLL	SYB SQL Server 4.x Driver	libpbsyb60.*
SYT Sybase SQL Server	PBSYT60.DLL	PBSYT60.DLL	—	—

SQL Server DB-Library cursor processing

To use SQL Server DB-Library cursor processing in PowerBuilder instead of the default cursor processing, you must set the Release DBParm parameter to **'4.2'** in the database profile or PowerBuilder application script.

FOR INFO For instructions, see the description of the Release DBParm parameter in Release (SQL Server 4.x) on page 504.

Supported SQL Server 4.x data types

PowerBuilder supports the following SQL Server DB-Lib data types in DataWindow objects and embedded SQL:

Binary	Money
Bit	Numeric
Character (less than 255 characters)	SmallInt
DateTime	Text
Decimal	Timestamp
Float	TinyInt
Image	VarBinary
Int	VarChar

Data type conversion

When you retrieve or update columns, PowerBuilder converts data appropriately between the SQL Server data type and the PowerScript data type.

Keep in mind, however, that similarly or identically named SQL Server and PowerScript data types do *not* necessarily have the same definitions.

FOR INFO For information about the definitions of PowerScript data types, see the *PowerScript Reference*.

Conversion in PowerBuilder scripts

When you use substitution variables in a PowerBuilder script, PowerBuilder does some special handling when converting from PowerScript data types of double and decimal to substitution strings. A decimal type is always converted to a string that begins with $. For example, the decimal value 12.34567 is converted to a string value of $12.34567.

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

Short data types

In addition to the data types listed above, PowerBuilder also supports the following short data types in SQL Server Version 4.x. Each of these data types has a storage size of four bytes:

Real	SmallMoney
SmallDateTime	

FOR INFO For more about these data types, see your SQL Server documentation.

Updating image and text columns

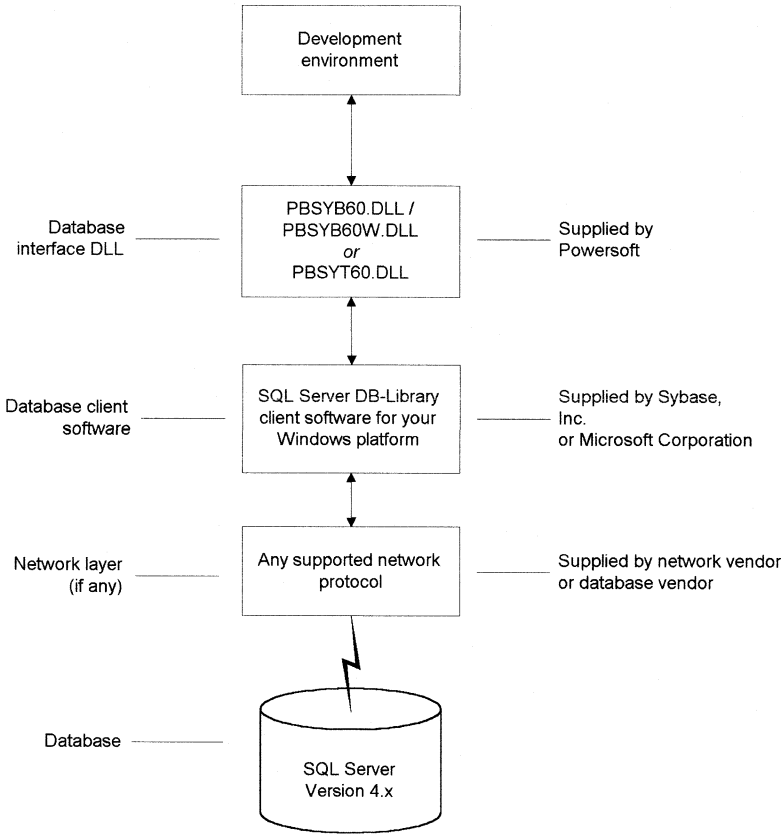
To update image or text columns within a transaction, AutoCommit must be set to False (the default).

FOR INFO For instructions, see "Setting database preferences" on page 340.

Basic software components for SQL Server 4.x

The following diagram shows the basic software components you need on the Windows, Macintosh, and UNIX platforms to access a SQL Server Version 4.x database in PowerBuilder.

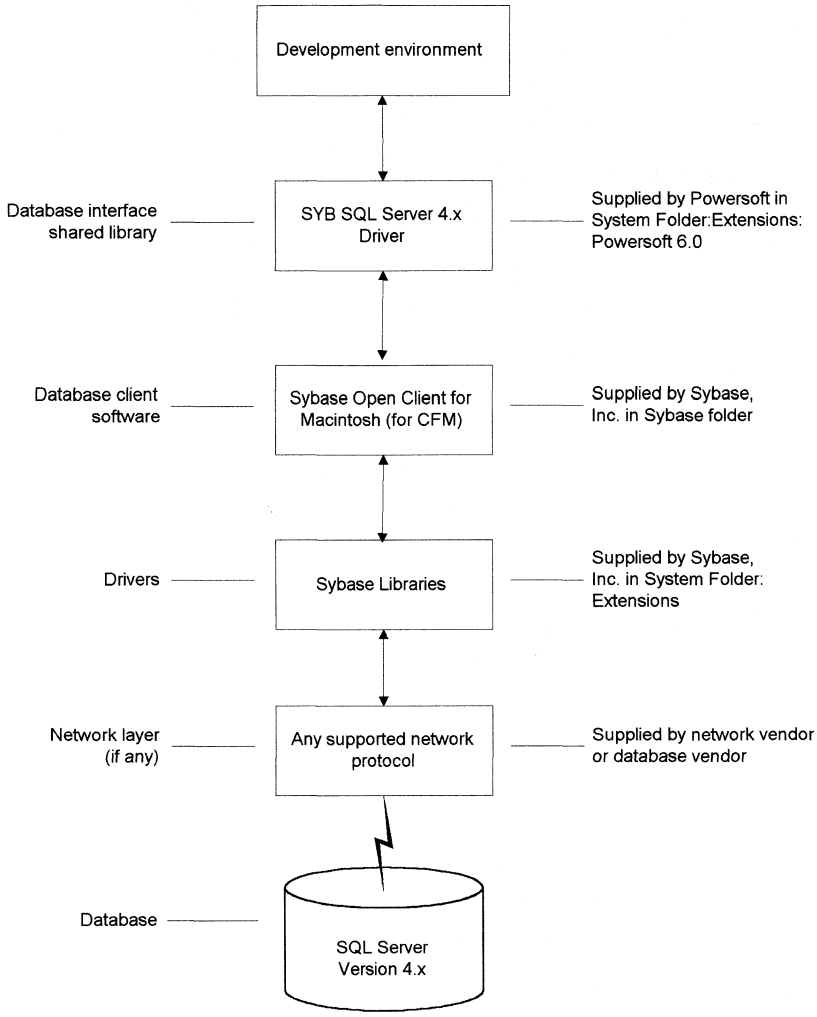
Accessing SQL Server 4.x on Windows



What to do next

FOR INFO For instructions on preparing your SQL Server 4.x database for use with PowerBuilder on the Windows platform, see "Preparing to use the SQL Server 4.x database on Windows" on page 207.

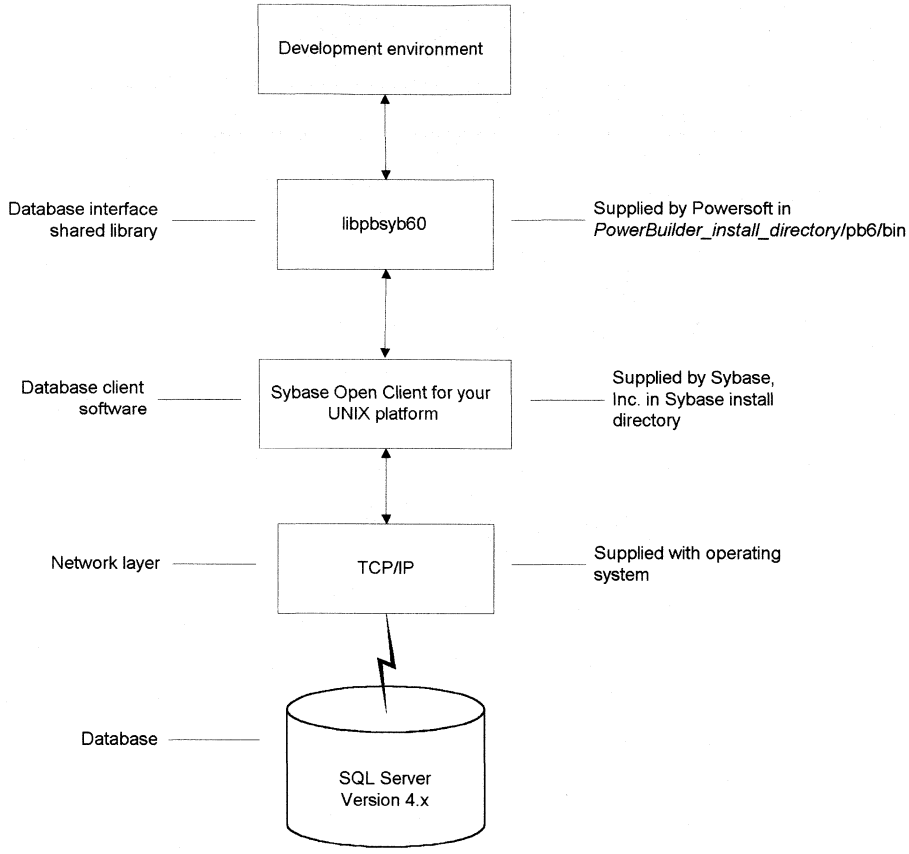
Accessing SQL Server 4.x on Macintosh



What to do next

FOR INFO For instructions on preparing your SQL Server 4.x database for use with PowerBuilder on the Macintosh platform, see "Preparing to use the SQL Server 4.x database on Macintosh" on page 211.

Accessing SQL Server 4.x on UNIX



What to do next

FOR INFO For instructions on preparing your SQL Server 4.x database for use with PowerBuilder on the UNIX platform, see "Preparing to use the SQL Server 4.x database on UNIX" on page 215.

Preparing to use the SQL Server 4.x database on Windows

What you do

Before you define the database interface and connect to a SQL Server 4.x database in PowerBuilder on Windows, follow these steps to prepare the database for use.

Overview of basic steps for SQL Server 4.x

Preparing a SQL Server 4.x database for use with PowerBuilder involves the same four basic tasks on all supported platforms. However, some of the steps for performing the tasks are different on each platform.

❖ **To prepare to use a SQL Server 4.x database:**

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

2 Install the Powersoft SQL Server 4.x database interface for your vendor and operating system platform.

3 Verify that you can connect to SQL Server outside PowerBuilder.

4 Install the required Powersoft stored procedures in the master database.

Step 1: install and configure the database server, network, and client software

❖ **To install and configure the database server, network, and client software for SQL Server 4.x on Windows:**

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

You must obtain the database server software from your database vendor (Sybase, Inc. or Microsoft Corporation).

FOR INFO For installation instructions, see your SQL 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 you can connect to the database server at your site.

For example, you must install the network communication driver that supports the network protocol and operating system platform you are using. If you are accessing Sybase SQL Server 4.x, the driver is installed as part of the Sybase Net-Library client software.

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

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

You must obtain the SQL Server client software from your database vendor (Sybase, Inc. or 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 SQL Server 4.x (Sybase or Microsoft) that you want to access
- The version of PowerBuilder that you are running

FOR INFO For installation instructions, see your SQL Server documentation.

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

Installing the client software places the correct configuration file in the SQL Server directory on your computer. For example, if you are using the Sybase Open Client software, the required configuration file is called SQL.INI.

The configuration file provides information that SQL 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 SQLEDIT configuration utility that comes with the Open Client software.

FOR INFO For information about setting up the SQL.INI or other required configuration file, see your 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 client computer:

- Powersoft SQL Server 4.x interface DLL
- Network communication DLL (for example, NLWNSCK.DLL for Windows Sockets-compliant TCP/IP)
- Database vendor DLL (for example, W3DBLIB.DLL)

Step 2: install the Powersoft SQL Server 4.x database interface

❖ **To install the Powersoft SQL Server 4.x database interface:**

- When prompted to do so by the PowerBuilder setup program, select the appropriate Powersoft SQL Server 4.x database interface for your vendor and operating system platform.

FOR INFO For a list of available SQL Server 4.x database interfaces, see "Supported versions and platforms for SQL Server 4.x" on page 202.

FOR INFO For installation instructions, see the *PowerBuilder Installation Guide*.

Step 3: verify the SQL Server 4.x connection outside PowerBuilder

❖ **To verify the SQL Server connection on Windows:**

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

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

  - **Accessing the database server** Tools such as Sybase SybPing (or any other 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 on 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 Powersoft stored procedures for SQL Server 4.x

PowerBuilder requires you to install certain Powersoft stored procedures in the master database *before* you connect to a SQL Server 4.x database for the first time. PowerBuilder uses these stored procedures to get information about tables and columns from the DBMS system catalog.

❖ **To install the Powersoft stored procedures:**

- Run the SQL script or scripts you need to install the Powersoft stored procedures in the master database.

  FOR INFO For instructions, see "Installing Powersoft stored procedures in SQL Server databases" on page 274.

What to do next

FOR INFO For instructions on defining the SQL Server 4.x database interface in PowerBuilder, see "Defining the SQL Server 4.x database interface" on page 220.

Preparing to use the SQL Server 4.x database on Macintosh

What you do

Before you define the database interface and connect to a SQL Server 4.x database in PowerBuilder on Macintosh, follow these steps to prepare the database for use.

Overview of basic steps for SQL Server 4.x

Preparing a SQL Server 4.x database for use with PowerBuilder involves the same four basic tasks on all supported platforms. However, some of the steps for performing the tasks are different on each platform.

❖ **To prepare to use a SQL Server 4.x database:**

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

2 Install the Powersoft SQL Server 4.x database interface for your vendor and operating system platform.

3 Verify that you can connect to SQL Server outside PowerBuilder.

4 Install the required Powersoft stored procedures in the master database.

Step 1: install and configure the database server, network, and client software

❖ **To install and configure the database server, network, and client software for SQL Server 4.x on Macintosh:**

1 Make sure the appropriate SQL Server 4.x database software is installed and running on the server machine specified in your Hosts and Interfaces files. (For more on the Interfaces file, see step 4.)

You must obtain the database server software from your database vendor (Sybase, Inc. or Microsoft Corporation).

FOR INFO For installation instructions, see your Sybase or Microsoft documentation.

2 Make sure the supported network software is installed and running on your Macintosh and is properly configured so you can connect to the database server at your site.

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

Sybase Open Client for Macintosh currently supports communication using either of the following network protocols:

- **MacTCP** If you are using MacTCP, you or your system administrator must use the MacTCP control panel to set your computer's IP address and (if necessary) gateway IP address. After setting the IP address, you may need to restart your Macintosh so it recognizes the address. In addition, make sure the Hosts and Services configuration files are properly set up for your environment.
- **Communications Toolbox** If you are using Communications Toolbox, make sure a supported third-party communications tool (such as the TSS DECnet Tool) is installed and running on your Macintosh.

3 Install Sybase Open Client for Macintosh software on your Macintosh.

To connect to a SQL Server 4.x database in PowerBuilder on Macintosh:

- **On Power Macintosh** Install Sybase Open Client for Power Macintosh Release *10.0.3 or higher*. This version supports Apple Code Fragment Manager.

The Sybase runtime installation provides all the components you need to access the database.

You must obtain Sybase Open Client for Macintosh software from Sybase, Inc.

Accessing Microsoft SQL Server 4.x on Macintosh

To access a Microsoft SQL Server Version 4.x database in PowerBuilder on Macintosh, you *must* obtain the required Open Client for Macintosh software from Sybase, Inc. On the Macintosh platform, the Open Client for Macintosh software is *not* included when you purchase the Microsoft SQL Server 4.x database software.

---

4 Edit the Interfaces file to provide correct network information for each database server you want to access.

The Interfaces file is a text file that you edit to provide network name and address information for each database server you want to access. PowerBuilder checks this file to determine the location of the server specified in your database profile. (The server name specified in your database profile *must match exactly* the server name specified in the Interfaces file.)

The Interfaces file must contain an entry for each database server you want to access. Each entry contains information about where the server is located on your network. The format of the entry depends on the network protocol you are using to connect to the server—MacTCP or Communications Toolbox.

FOR INFO For complete instructions on editing the Interfaces file, see your Sybase Open Client for Macintosh documentation and the comments in the Interfaces file. Here are two examples:

- **MacTCP example** The following is a sample Interfaces file entry for the MacTCP protocol. *TEST* is the server name, *sybase* is the machine name, and *5003* is the port number. The values *query*, *MacTCP*, and *mac_ether* are required. A tab character must precede the query line.

```
#TEST on sybase using MacTCP
TEST
    query MacTCP mac_ether sybase 5003
```

- **Communications Toolbox example** The following is a sample Interfaces file entry for the Communications Toolbox protocol. The specific format of the entry depends on the third-party communications tool you are using. *SALES* is the Sybase server name, *TSS DECnet Tool* is the communications tool, *sybase* is the DECnet node name, and *312* is the DECnet object number. The values *query*, *ctb*, and *mac_ether* are required. A tab character must precede the query line.

```
#SALES on sybase using Communications Toolbox
SALES
    query ctb mac_ether {TSS DECnet Tool}
        {Node sybase Object 312}
```

5 Use the SybaseConfig control panel to make sure the SYBASE environment variable is properly set. The SYBASE environment variable is set from the location of your Interfaces file.

FOR INFO For complete instructions on using the SybaseConfig control panel, see your Sybase Open Client for Macintosh documentation.

These steps summarize this procedure:

1 Open the SybaseConfig control panel and click the Interfaces File button.

2 Move to and select the Interfaces file in the listbox.

3 Click the Open button to set the SYBASE environment variable.

4 Close the SybaseConfig control panel.

Step 2: install the Powersoft SQL Server 4.x database interface

❖ **To install the Powersoft SQL Server 4.x database interface:**

♦ When prompted to do so by the PowerBuilder setup program, select the appropriate Powersoft SQL Server 4.x database interface for your vendor and operating system platform.

FOR INFO For a list of available SQL Server 4.x database interfaces, see "Supported versions and platforms for SQL Server 4.x" on page 202.

FOR INFO For installation instructions, see the *PowerBuilder Installation Guide (Macintosh).*

Step 3: verify the connection outside PowerBuilder

❖ **To verify the SQL Server 4.x connection on Macintosh:**

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

2 Some possible ways you can verify a SQL Server connection are by running either of the following:

♦ **SybPing** The SybPing network diagnostic utility checks that you can reach the SQL Server database server from your Macintosh. SybPing is included with the Sybase Net-Library software.

- **isql** The isql interactive SQL utility checks that you can access the database from your Macintosh. It lets you connect to and query the database. ISQL is included with the Sybase Open Client for Macintosh software.

  FOR INFO For instructions on using SybPing and isql, see your Sybase Open Client for Macintosh documentation.

Step 4: install the Powersoft stored procedures for SQL Server 4.x

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

❖ **To install the Powersoft stored procedures:**

- Run the SQL script or scripts you need to install the Powersoft stored procedures in the master database.

  FOR INFO For instructions, see "Installing Powersoft stored procedures in SQL Server databases" on page 274.

What to do next

FOR INFO For instructions on defining the SQL Server 4.x database interface in PowerBuilder, see "Defining the SQL Server 4.x database interface" on page 220.

Preparing to use the SQL Server 4.x database on UNIX

What you do

Before you define the database interface and connect to a SQL Server 4.x database in PowerBuilder on UNIX, follow these steps to prepare the database for use.

Overview of basic steps for SQL Server 4.x

Preparing a SQL Server 4.x database for use with PowerBuilder involves the same four basic tasks on all supported platforms. However, some of the steps for performing the tasks are different on each platform.

❖ **To prepare to use a SQL Server 4.x database:**

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

2 Install the Powersoft SQL Server 4.x database interface for your vendor and operating system platform.

3 Verify that you can connect to SQL Server outside PowerBuilder.

4 Install the required Powersoft stored procedures in the master database.

Step 1: install and configure the database server, network, and client software

❖ **To install and configure the database server, network, and client software for SQL Server 4.x on UNIX:**

1 Make sure the appropriate SQL Server 4.x database software is installed and running on the server machine specified in your interfaces files and database profile. (For more on the interfaces file, see step 4.)

You must obtain the database server software from your database vendor (Sybase, Inc. or Microsoft Corporation).

FOR INFO For installation instructions, see your Sybase or Microsoft documentation.

2 Make sure the required TCP/IP software is running on your workstation and is properly configured so you can connect to the database server at your site.

The TCP/IP software is supplied as part of the UNIX operating system.

3 Install and configure the Sybase Open/Client Server software on each client machine on which PowerBuilder for UNIX is installed.

To connect to a SQL Server 4.x database in PowerBuilder for UNIX, you must install Sybase Open Client/Server Release 10.0.1 or 10.0.3 for SunOS Release 5.x (SVR4).

You must obtain the Open Client/Server software from Sybase, Inc.

Accessing Microsoft SQL Server 4.x on UNIX

To access a Microsoft SQL Server Version 4.x database in PowerBuilder on UNIX, you *must* obtain the required Open Client/Server software from Sybase, Inc. On the UNIX platform, the Open Client/Server software is *not* included when you purchase the Microsoft SQL Server 4.x database software.

Configuring your Sybase Open Client/Server software involves using the *sybinit* utility to perform the following tasks:

- **Initializing the software** You must initialize (configure) each of the Sybase Open Client/Server products you installed.
- **Setting up the interfaces file** When you install Sybase Open Client/Server, the **interfaces file** is automatically created in the Sybase install directory. The interfaces file contains network name and address information for each Sybase SQL Server you want to access.

  PowerBuilder for UNIX checks the interfaces file to determine the location of the server specified in your database profile. Therefore, the server name in your database profile *must match exactly* the server name specified in your interfaces file.

  You should use sybinit to edit the interfaces file to ensure that the file entries are formatted correctly.

FOR INFO For instructions on using the sybinit utility to initialize your Open Client/Server software and set up the interfaces file, see your Sybase Open Client/Server documentation.

4 Make sure the SYBASE environment variable is defined in your shell initialization file.

  The SYBASE environment variable points to the directory where the Sybase Open Client/Server software is installed.

  Here is a sample SYBASE definition for a C shell initialization file:

```
setenv SYBASE /export/home/sybase
```

5 Append $SYBASE/bin to the PATH environment variable in your shell initialization file.

  Here is the PATH definition for a C shell initialization file:

```
setenv PATH ${SYBASE}/bin:${PATH}
```

6 Append $SYBASE/lib to the library path environment variable in your shell initialization file.

  Here is the LD_LIBRARY_PATH definition for a C shell initialization file on Solaris:

```
setenv LD_LIBRARY_PATH ${SYBASE}/lib:
${LD_LIBRARY_PATH}
```

7 Make sure the PBHOME environment variable is defined in your shell initialization file.

The PBHOME environment variable points to the directory where PowerBuilder for UNIX is installed (for example, /export/home/pb6).

Here is a sample PBHOME definition for a C shell initialization file:

```
setenv PBHOME /export/home/pb6
```

Step 2: install the Powersoft SQL Server 4.x database interface

❖ **To install the Powersoft SQL Server 4.x database interface:**

1 When prompted to do so by the PowerBuilder setup program, select the Powersoft SQL Server 4.x database interface.

FOR INFO For a list of the Powersoft SQL Server 4.x database interfaces available on each platform, see "Supported versions and platforms for SQL Server 4.x" on page 202.

FOR INFO For installation instructions, see the *PowerBuilder Installation Guide (UNIX)*.

2 (Optional) Use the **which ldd** command to make sure the **ldd** command is in your path.

Typically the ldd command is in the /usr/bin directory.

3 (Optional) Issue an ldd command on libpbsyb60 (the Powersoft SQL Server 4.x database interface shared library) to make sure it can locate the appropriate shared libraries. For example, on Solaris, type:

```
ldd $PBHOME/bin/libpbsyb60.so
```

For example, you should see output similar to the following:

```
libsybdb.so=>/export/home/sybase/lib/libsybdb.so
```

This library must be present to connect to SQL Server 4.x in PowerBuilder for UNIX.

Step 3: verify the SQL Server 4.x connection outside PowerBuilder

❖ **To verify the SQL Server 4.x connection on UNIX:**

♦ Make sure you can connect to the SQL Server 4.x database server and database you want to access from outside PowerBuilder.

You can use the *isql* interactive SQL utility that comes with the Sybase Open Client/Server software to verify your connection.

Type the isql command as follows. Be sure to specify the same connection parameters you plan to use in your PowerBuilder database profile to access the SQL Server 4.x database.

```
isql -S server_name -U user_name -P password
```

Parameter	Description
server_name	The name of the database server you want to access. This name *must match exactly* the server name specified in your interfaces file. Server names are case sensitive
user_name	The user name required to log in to the database server. User names are case sensitive
password	The password required to log in to the database server. Passwords are case sensitive

Example

Assume you are using the following connection parameters in PowerBuilder for UNIX to access a SQL Server 4.x database:

Parameter	Value
server_name	sybase
user_name	sam
password	beta2

To verify your connection to the database outside PowerBuilder for UNIX, type the following isql command:

```
isql -S sybase -U sam -P beta2
```

Step 4: install the Powersoft stored procedures for SQL Server 4.x

PowerBuilder requires you to install certain Powersoft stored procedures in the master database *before* you connect to a SQL Server 4.x database for the first time. PowerBuilder uses these stored procedures to get information about tables and columns from the DBMS system catalog.

❖ **To install the Powersoft stored procedures:**

- Run the SQL script or scripts you need to install the Powersoft stored procedures in the master database.

  FOR INFO For instructions, see "Installing Powersoft stored procedures in SQL Server databases" on page 274.

What to do next

FOR INFO For instructions on defining the SQL Server 4.x database interface in PowerBuilder, see "Defining the SQL Server 4.x database interface" on page 220.

Defining the SQL Server 4.x database interface

To define a connection through the SQL Server 4.x interface, you must create a database profile by supplying values for at least the basic connection parameters in the Database Profile Setup - SQL Server Version 4.x dialog box. You can then select this profile at any time to connect to your database in the development environment.

❖ **To create a database profile for a SQL Server 4.x connection:**

1. Click the Database Profile button in the PowerBar.
   or
   In the Database painter, select File>Connect>Setup from the menu bar.

   The Database Profiles dialog box displays, listing your installed Powersoft database interfaces and defined database profiles.

2. Select the SYB or SYT interface and click New.
   or
   Display the popup menu for the SYB or SYT interface and select New.

   The Database Profile Setup - SQL Server Version 4.x dialog box displays.

3 On the Connection tab, type the profile name and supply values for the basic connection parameters required by the SQL Server 4.x database interface.

The required connection parameters include the server, login ID, password, and database. If you omit any of these values, a dialog box displays to prompt you for the missing information when you connect to the database.

FOR INFO For information about the values you should supply for these basic connection parameters, click Help.

4 (Optional) On the Connection and other tabs, supply values for any additional options (DBParm parameters and SQLCA properties) you want to set to take advantage of DBMS-specific features that the interface supports.

FOR INFO For information about additional connection parameters for the SQL Server 4.x interface and how to set them, click Help.

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

The Database Profiles dialog box displays, with the new profile name highlighted. The database profile values are saved in the PowerBuilder initialization file.

What to do next

FOR INFO For instructions on connecting to the database, see "Connecting to a database" on page 295.

Sybase InformationConnect DB2 Gateway Interface

This section describes how to use the Sybase InformationConnect DB2 Gateway interface in PowerBuilder. (This interface was formerly called the MDI Database Gateway Interface for DB2.)

Supported versions and platforms for InformationConnect Gateway

Versions

You can access a DB2 database using the Sybase InformationConnect DB2 Gateway Version 2 Release 1 or higher with PowerBuilder. The InformationConnect Gateway interface uses a Powersoft DLL named PBMDI60.DLL to access the database.

Platforms

The InformationConnect Gateway interface is available on the following operating system platforms:

- Windows NT
- Windows 95

Supported DB2 data types for InformationConnect Gateway

PowerBuilder supports the following DB2 data types in DataWindow objects and embedded SQL:

Char *	Long VarChar *
Date	SmallInt
Decimal	Time
Float	Timestamp (DateTime)
Int	VarChar *

* See the following sections for information about limits when creating and updating tables.

Data type conversion

When you retrieve or update columns, the InformationConnect Gateway converts data appropriately between the DB2 data type and the SQL Server data type. PowerBuilder then converts the data appropriately between the SQL Server data type and the PowerScript data type.

Keep in mind, however, that similarly or identically named SQL Server and PowerScript data types do *not* necessarily have the same definitions.

FOR INFO For information about the definitions of PowerScript data types, see the *PowerScript Reference*.

Updating and inserting columns

InformationConnect Gateway Version 2.5 or higher If you are using the InformationConnect Gateway Version 2.5 or higher, you can update and insert Char, VarChar, and Long VarChar columns that are greater than 254 characters in length.

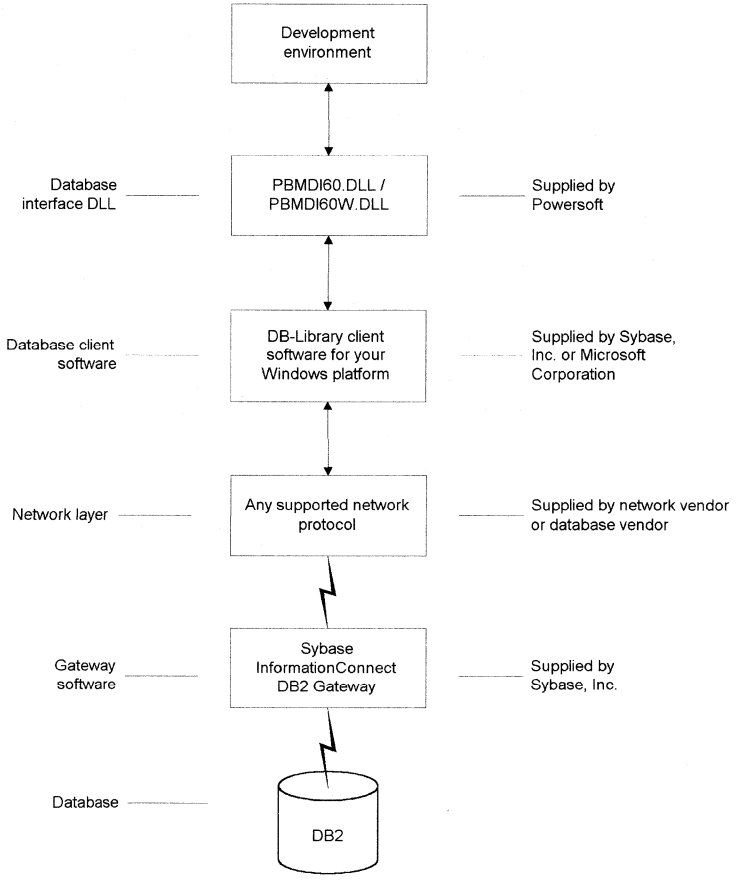
InformationConnect Gateway earlier than Version 2.5 If you are using a version of the InformationConnect Gateway *earlier* than Version 2.5, you can update and insert only the first 254 characters of a Char, VarChar, and Long VarChar column.

Creating tables with VarChar columns

Regardless of the version of the InformationConnect Gateway you are using, you can create tables with VarChar columns up to a maximum length of 4046 characters.

Basic software components for InformationConnect Gateway interface

The following diagram shows the basic software components you need to access a DB2 database using the InformationConnect Gateway interface:



Preparing to use the database with InformationConnect Gateway

What you do

Before you define the interface and connect to a DB2 database through the InformationConnect Gateway interface, follow these steps to prepare the database for use.

Overview of basic steps for InformationConnect Gateway interface

Preparing a DB2 database for use with the Powersoft InformationConnect Gateway interface involves four basic steps.

❖ **To prepare a DB2 database that you access with the InformationConnect Gateway interface:**

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

2 Install the Powersoft InformationConnect Gateway interface.

3 Verify that you can connect to the DB2 server and database outside PowerBuilder.

4 (Optional) Create the Powersoft repository outside PowerBuilder.

Step 1: install and configure the database server, network, gateway, and client software for InformationConnect Gateway

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

1 Make sure the DB2 database software is installed and running on the server specified in your database profile.

You must obtain the database server software from IBM.

FOR INFO For installation instructions, see your IBM DB2 documentation.

2 Make sure the required network software is installed and properly configured so you can connect to the gateway machine and DB2 database server.

FOR INFO For instructions, see the installation and administration guide in your InformationConnect Gateway documentation.

3 Make sure the required InformationConnect Gateway software is installed on the gateway machine.

FOR INFO For instructions, see the installation and administration guide in your InformationConnect Gateway documentation.

4 Install the required DB-Library client software on each client computer on which PowerBuilder is installed:

- **Windows 95 and Windows NT** If you are using the Powersoft InformationConnect Gateway interface in PowerBuilder on Windows 95 or Windows NT, you should install Sybase Open Client DB-Library Version 10.0.3 or higher.

You must obtain the client software from Sybase, Inc. or Microsoft Corporation.

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

Step 2: install the Powersoft InformationConnect Gateway interface

❖ **To install the Powersoft InformationConnect Gateway Interface:**

- When prompted to do so by the PowerBuilder setup program, select the Powersoft Sybase InformationConnect DB2 Gateway interface.

FOR INFO For instructions, see the *PowerBuilder Installation Guide*.

Step 3: verify the InformationConnect Gateway connection outside PowerBuilder

❖ **To verify the InformationConnect Gateway connection:**

1 Make sure the InformationConnect Gateway is started and the network protocol is properly set up.

2 Use ISQL or any Windows-based utility to connect to the InformationConnect Gateway.

When connecting, be sure to specify the same parameters you plan to use in your PowerBuilder database profile to access the database.

3 Issue a request to the database through the InformationConnect Gateway to verify that you can perform database operations.

FOR INFO For more information, see the installation and administration guide in your InformationConnect Gateway documentation.

Step 4: create the Powersoft repository outside PowerBuilder when using the InformationConnect Gateway interface

PowerBuilder uses a collection of five Powersoft system tables, known as the **repository**, to store extended attribute information. These tables are created automatically the first time a user connects to the database using PowerBuilder.

System administrators at DB2 sites may prefer to create the Powersoft repository themselves outside PowerBuilder to control the access rights and location of these tables.

❖ **To create the Powersoft repository outside PowerBuilder:**

- Run the DB2SYSPB.SQL script outside PowerBuilder using the SQL tool of your choice.

  FOR INFO For instructions, see "Creating the Powersoft repository in DB2 databases" on page 271.

Defining the InformationConnect Gateway interface

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

❖ **To create a database profile for a Sybase InformationConnect connection:**

1. Click the Database Profile button in the PowerBar.
   or
   In the Database painter, select File>Connect>Setup from the menu bar.

   The Database Profiles dialog box displays, listing your installed Powersoft database interfaces and defined database profiles.
2. Select the MDI interface and click New.
   or
   Display the popup menu for the MDI interface and select New.

   The Database Profile Setup - Sybase InformationConnect dialog box displays.
3. On the Connection tab, type the profile name and supply values for the basic connection parameters required by the Sybase InformationConnect interface.

   The required connection parameters include the server, login ID, password, and database. If you omit any of these values, a dialog box displays to prompt you for the missing information when you connect to the database.

   FOR INFO For information about the values you should supply for these basic connection parameters, click Help.

4 (Optional) On the Connection and other tabs, supply values for any additional options (DBParm parameters and SQLCA properties) you want to set to take advantage of DBMS-specific features that the interface supports.

FOR INFO For information about additional connection parameters for the Sybase InformationConnect interface and how to set them, click Help.

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

The Database Profiles dialog box displays, with the new profile name highlighted. The database profile values are saved in the PowerBuilder initialization file.

Specifying additional InformationConnect Gateway interface parameters

PowerBuilder prompts you to specify the following additional connection parameters (if you have not already done so) when you are using the InformationConnect Gateway interface to access a DB2 database:

- Owner of the PowerBuilder catalog tables (Powersoft repository)
- Tablespace where you want PowerBuilder to create tables
- Owner of the DB2 system tables
- Owner of the table containing database stored requests

❖ **To specify additional parameters for an InformationConnect Gateway interface connection:**

- Specify the connection parameters described in the following table *before you connect* to a DB2 database though the Powersoft InformationConnect Gateway interface. (If you change any of these parameters while connected to the database, PowerBuilder uses the new values the *next time you connect*.)

  FOR INFO For information about the PBCatalogOwner and SystemOwner DBParm parameters, see PBCatalogOwner on page 494 and SystemOwner on page 547.

Parameter	Description	Specify in
PowerBuilder catalog table owner	Specifies an owner for the five tables in the Powersoft repository	PowerBuilder Catalog Owner Name dialog box that displays when you select a database profile to connect *or* PBCatalogOwner DBParm parameter in your database profile
Tablespace	Specifies the name of the tablespace in which you want PowerBuilder to create tables A **tablespace** is a logical storage unit of a database. The data in a database is logically stored in the database and physically stored in data files	Table Space Name dialog box that displays the first time you create a table in a DB2 database If a value is already specified for the TableSpace variable in the [Database] section of the PowerBuilder initialization file, the Table Space Name dialog box does *not* display
DB2 system tables owner	Specifies the owner of the DB2 system tables that you want PowerBuilder to use to get information about tables and columns in your database	System Owner Name dialog box that displays the first time you open the Database painter or Database Administration painter *or* SystemOwner DBParm parameter in your database profile

Parameter	Description	Specify in
Stored requests table owner	Specifies the owner of the DB2 table containing database stored requests **Database stored requests** are SQL statements that reside in the host request library You can use stored requests in embedded SQL statements to perform retrieval or update operations on the database	Stored Requests dialog box that displays when you create a new DataWindow object with a stored procedure data source

What to do next

FOR INFO For instructions on connecting to the database, see "Connecting to a database" on page 295.

Sybase Net-Gateway for DB2 Interface

This section describes how to use the Sybase Net-Gateway for DB2 interface in PowerBuilder.

Supported versions and platforms for Sybase Net-Gateway

Versions

You can access a DB2 database using the Sybase Net-Gateway for DB2 Version 2.0 or higher with PowerBuilder. The Net-Gateway interface uses a Powersoft DLL named PBNET60.DLL to access the database.

Platforms

The Powersoft Net-Gateway interface is available on the following operating system platforms:

- Windows NT
- Windows 95

Supported DB2 data types for Sybase Net-Gateway

PowerBuilder supports the following DB2 data types in DataWindow objects, and embedded SQL:

Char (less than 255 characters)	Long VarChar
Date	SmallInt
Decimal	Time
Float	Timestamp (DateTime)
Int	VarChar (less than 255 characters)

Data type conversion

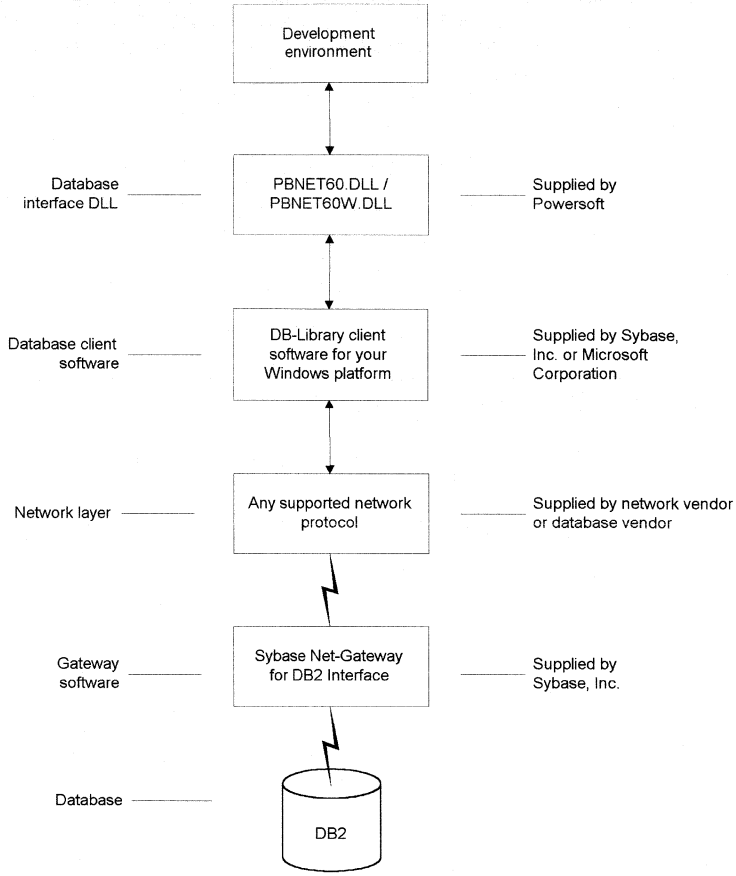
When you retrieve or update columns, the Sybase Net-Gateway for DB2 converts data appropriately between the DB2 data type and the SQL Server data type. PowerBuilder then converts the data appropriately between the SQL Server data type and the PowerScript data type.

Keep in mind, however, that similarly or identically named SQL Server and PowerScript data types do *not* necessarily have the same definitions.

FOR INFO For information about the definitions of PowerScript data types, see the *PowerScript Reference*.

Basic software components for Net-Gateway interface

The following diagram shows the basic software components you need to access a DB2 database using the Sybase Net-Gateway for DB2 interface;



Preparing to use the database with Sybase Net-Gateway

What you do

Before you define the interface and connect to a DB2 database through the Powersoft Net-Gateway interface, follow these steps to prepare the database for use.

Overview of basic steps for Sybase Net-Gateway

Preparing a DB2 database for use with the Powersoft Net-Gateway interface involves four basic steps.

❖ **To prepare a DB2 database that you access with the Powersoft Net-Gateway interface:**

1. Install and configure the required database server, network, gateway, and client software.
2. Install the Powersoft Net-Gateway interface.
3. Verify that you can connect to the DB2 server and database outside PowerBuilder.
4. (Optional) Create the Powersoft repository outside PowerBuilder.

Step 1: install and configure the database server, network, gateway, and client software for Sybase Net-Gateway

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

1. Make sure the DB2 database software is installed and running on the server specified in your database profile.

   You must obtain the database server software from IBM.

   FOR INFO For installation instructions, see your IBM DB2 documentation.

2. Make sure the required network software is installed and properly configured so you can connect to the gateway machine and DB2 database server.

   FOR INFO For instructions, see your network or database administrator.

3. Make sure the required Sybase Net-Gateway for DB2 interface software is installed on the gateway machine.

   FOR INFO For instructions, see your Sybase Net-Gateway documentation.

4. Install the required DB-Library client software on each client computer on which PowerBuilder is installed:

- **Windows 95 and Windows NT** If you are using the Sybase Net-Gateway interface in PowerBuilder on Windows 95 or Windows NT, you should install Sybase Open Client DB-Library Version 10.0.3 or higher.

You must obtain the client software from Sybase, Inc. or Microsoft Corporation.

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

Step 2: install the Net-Gateway interface

❖ **To install the Net-Gateway interface:**

- When prompted to do so by the PowerBuilder setup program, select the Sybase Net-Gateway for DB2 interface.

FOR INFO For instructions, see the *PowerBuilder Installation Guide*.

Step 3: verify the Sybase Net-Gateway connection outside PowerBuilder

❖ **To verify the Sybase Net-Gateway connection:**

1 Make sure the database gateway is started and the network protocol is properly set up.

2 Use ISQL or any Windows-based utility to connect to the Sybase Net-Gateway.

When connecting, be sure to specify the same parameters you plan to use in your PowerBuilder database profile to access the database.

3 Issue a request to the database through the Sybase Net-Gateway to verify that you can perform database operations.

FOR INFO For more information, see your Sybase Net-Gateway documentation.

Step 4: create the Powersoft repository outside PowerBuilder when using the Sybase Net-Gateway interface

PowerBuilder uses a collection of five Powersoft system tables, known as the **repository**, to store extended attribute information. These tables are created automatically the first time a user connects to the database using PowerBuilder.

System administrators at DB2 sites may prefer to create the Powersoft repository themselves outside PowerBuilder to control the access rights and location of these tables.

❖ **To create the Powersoft repository outside PowerBuilder:**

- Run the DB2SYSPB.SQL script outside PowerBuilder using the SQL tool of your choice.

  FOR INFO For instructions, see "Creating the Powersoft repository in DB2 databases" on page 271.

Grant PUBLIC authority to all Sybase Net-Gateway plans
Make sure you or your database administrator has granted PUBLIC authority for all Sybase Net-Gateway plans. Otherwise, creation of the Powersoft repository will fail.

Defining the Net-Gateway interface

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

❖ **To create a database profile for a Net-Gateway interface connection:**

1. Click the Database Profile button in the PowerBar.
   or
   In the Database painter, select File>Connect>Setup from the menu bar.

   The Database Profiles dialog box displays, listing your installed Powersoft database interfaces and defined database profiles.
2. Select the NET interface and click New.
   or
   Display the popup menu for the NET interface and select New.

   The Database Profile Setup - Sybase Net-Gateway dialog box displays.

3 On the Connection tab, type the profile name and supply values for the basic connection parameters required by the Sybase Net-Gateway interface.

The required connection parameters include the server, login ID, password, and database. If you omit any of these values, a dialog box displays to prompt you for the missing information when you connect to the database.

FOR INFO For information about the values you should supply for these basic connection parameters, click Help.

4 (Optional) On the Connection and other tabs, supply values for any additional options (DBParm parameters and SQLCA properties) you want to set to take advantage of DBMS-specific features that the interface supports.

FOR INFO For information about additional connection parameters for the Sybase Net-Gateway interface and how to set them, click Help.

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

The Database Profiles dialog box displays, with the new profile name highlighted. The database profile values are saved in the PowerBuilder initialization file.

Specifying additional Net-Gateway interface parameters

PowerBuilder prompts you to specify the following additional connection parameters (if you have not already done so) when you are using the Net-Gateway interface to access a DB2 database:

- Owner of the PowerBuilder catalog tables (Powersoft repository)
- Tablespace where you want PowerBuilder to create tables
- Owner of the DB2 system tables

❖ **To specify additional parameters for a Net-Gateway interface connection:**

- Specify the connection parameters described in the following table *before you connect* to a DB2 database through the Powersoft Net-Gateway interface. (If you change either of these parameters while connected to the database, PowerBuilder uses the new values the *next time you connect*.)

  FOR INFO For information about the PBCatalogOwner and SystemOwner DBParm parameters, see their descriptions in PBCatalogOwner on page 494 and SystemOwner on page 547.

Parameter	Description	Specify in
PowerBuilder catalog table owner	Specifies an owner for the five tables in the Powersoft repository	PowerBuilder Catalog Owner Name dialog box that displays when you select a database profile to connect *or* PBCatalogOwner DBParm parameter in your database profile
Tablespace	Specifies the name of the tablespace in which you want PowerBuilder to create tables A **tablespace** is a logical storage unit of a database. The data in a database is logically stored in the database and physically stored in data files	Table Space Name dialog box that displays the first time you create a table in a DB2 database If a value is already specified for the TableSpace variable in the [Database] section of the PowerBuilder initialization file, the Table Space Name dialog box does *not* display

Parameter	Description	Specify in
DB2 system tables owner	Specifies the owner of the DB2 system tables that you want PowerBuilder to use to get information about tables and columns in your database	System Owner Name dialog box that displays the first time you open the Database painter or Database Administration painter *or* SystemOwner DBParm parameter in your database profile

What to do next

FOR INFO For instructions on connecting to the database, see "Connecting to a database" on page 295.

Sybase SQL Server System 10.x and System 11.x

This section describes how to use the Powersoft Sybase System 10.x and System 11.x database interface in PowerBuilder.

Client Library API The Powersoft System 10.x and System 11.x database interface uses the Open Client CT-Library (CT-Lib) application programming interface (API) to access the database.

When you connect to a System 10 or System 11 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.

If you want to use the DB-Library (DB-Lib) API to access a SQL Server database, you must install one of the Powersoft SQL Server Version 4.x database interfaces, as described in "SQL Server Version 4.x" on page 202.

Sybase Adaptive Server Enterprise Although this section refers generically to Sybase System 11.x, the name of the Version 11.5 or higher product family is Sybase Adaptive Server Enterprise.

Supported versions and platforms for System 10.x and System 11.x

Versions

You can access Sybase SQL Server Release 10.0.x and 11.0.x using the Powersoft System 10.x and System 11.x database interface.

Platforms

The Powersoft System 10.x and System 11.x database interface is available on all supported platforms.

The interface uses a Powersoft DLL (on Windows) or shared library (on Macintosh and UNIX) to access the database through the Sybase Open Client CT-Lib API. The name of the DLL or shared library depends on the PowerBuilder platform you are using.

SYD distributed application interface on UNIX
On UNIX platforms, PowerBuilder provides the SYD Sybase Systems 10 and 11 distributed application interface in addition to the SYC Sybase System 10 and 11 interface. The SYD interface works with the thread-safe libraries provided by Sybase Open Client Version 11.1 or higher to enable you to build distributed applications that access Sybase SQL Server 10.x and 11.x databases.

FOR INFO See "Systems 10.x and 11.x distributed application interface on UNIX" on page 242.

Powersoft DBMS identifier	Windows 95	Windows NT	Macintosh	UNIX
SYC	PBSYC60.DLL	PBSYC60.DLL	SYC Sybase System 10/11 Driver	libpbsyc60.*
SYD	—	—	—	libpbsyd60.*

Accessing System 10.x and System 11.x databases

You should use the Powersoft System 10.x and System 11.x database interface (SYC or SYD DBMS identifier) to connect to Sybase System 10.x and System 1.x1 databases in PowerBuilder. This interface uses CT-Lib to access the database.

If you are currently using one of the Powersoft SQL Server 4.x database interfaces (SYB or SYT DBMS identifier) that goes through DB-Lib to access System 1.x0 or System 1.x, upgrading to the System 10 and System 11 database interface provides you with the following benefits:

- **Additional features** The Powersoft System 10.x and System 11.x database interface supports additional features not available in the Powersoft SQL Server 4.x interfaces, including:
  - Support for decimal, numeric, and identity data types
  - Support for declarative referential integrity constraints
  - Using sybsystemprocs database to hold Powersoft stored procedures
  - Support for Sybase Open Client network-based security and directory services

FOR INFO See "Using Sybase Open Client security services" on page 263 and "Using Sybase Open Client directory services" on page 265

- Using PRINT statements in stored procedures for debugging purposes

  FOR INFO See "Using PRINT statements in SQL Server stored procedures" on page 270

- **Future enhancements** Future development efforts at Powersoft will focus on enhancing the System 10.x and System 11.x database interface.

Systems 10.x and 11.x distributed application interface on UNIX

On all UNIX platforms, PowerBuilder provides the Sybase Systems 10.x and 11.x distributed application interface (SYD DBMS identifier). This interface works with the thread-safe version of the Sybase Open Client Version 11.1 or higher libraries.

When to use

When building a distributed PowerBuilder server application that accesses the database through an API, you *must* use a thread-safe client library. Therefore, you *must* use the SYD interface instead of the SYC interface if you are building a distributed PowerBuilder application on UNIX that accesses a SQL Server 10.x or 11.x database.

Interface	Accesses	Client software	When to use
SYD	Sybase SQL Server 10.x or 11.x	Sybase Open Client/Server 11.1.1 or higher thread-safe libraries	To build *distributed* PowerBuilder applications
SYC	Sybase SQL Server 10.x or 11.x	Sybase Open Client 10.0.01, 10.0.3, 10.0.4, or 11.1.1 or higher	To build *nondistributed* PowerBuilder applications

Required client software

To use the Sybase Systems 10 and 11 distributed application interface on UNIX, you *must* install the Sybase Open Client Version 11.1.1 or higher thread-safe libraries on the client machine.

Open Client security and directory services

Like the SYC interface on UNIX and other platforms, the SYD Sybase Systems 10 and 11 distributed application interface on UNIX lets you set several DBParm parameters that enable Open Client 11.1 network-based security and directory services in your PowerBuilder application.

FOR INFO See "Using Sybase Open Client security services" on page 263 and "Using Sybase Open Client directory services" on page 265.

Supported System 10.x and System 11.x data types

PowerBuilder supports the following System 10.x and System 11.x data types in DataWindow objects and embedded SQL:

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

Data type conversion

When you retrieve or update columns, PowerBuilder converts data appropriately between the SQL Server data type and the PowerScript data type.

Keep in mind, however, that similarly or identically named SQL Server and PowerScript data types do *not* necessarily have the same definitions.

FOR INFO For information about the definitions of PowerScript data types, see the *PowerScript Reference*.

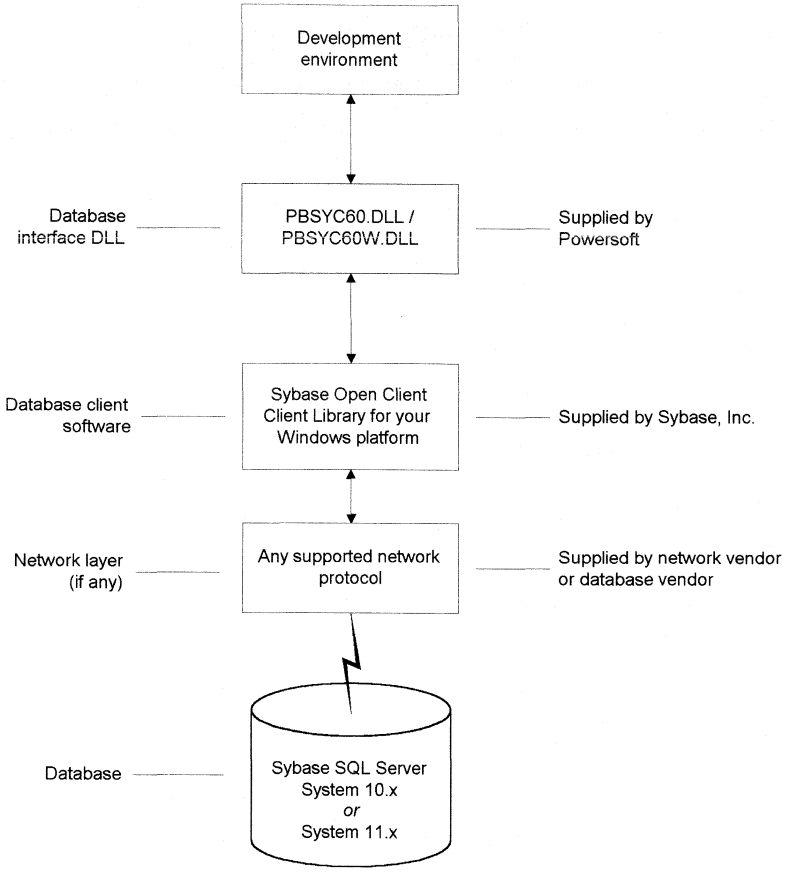
Conversion in PowerBuilder scripts

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

Basic software components for System 10.x and System 11

The following diagrams shows the basic software components you need to access a Sybase System 10.x or System 11.x database in PowerBuilder.

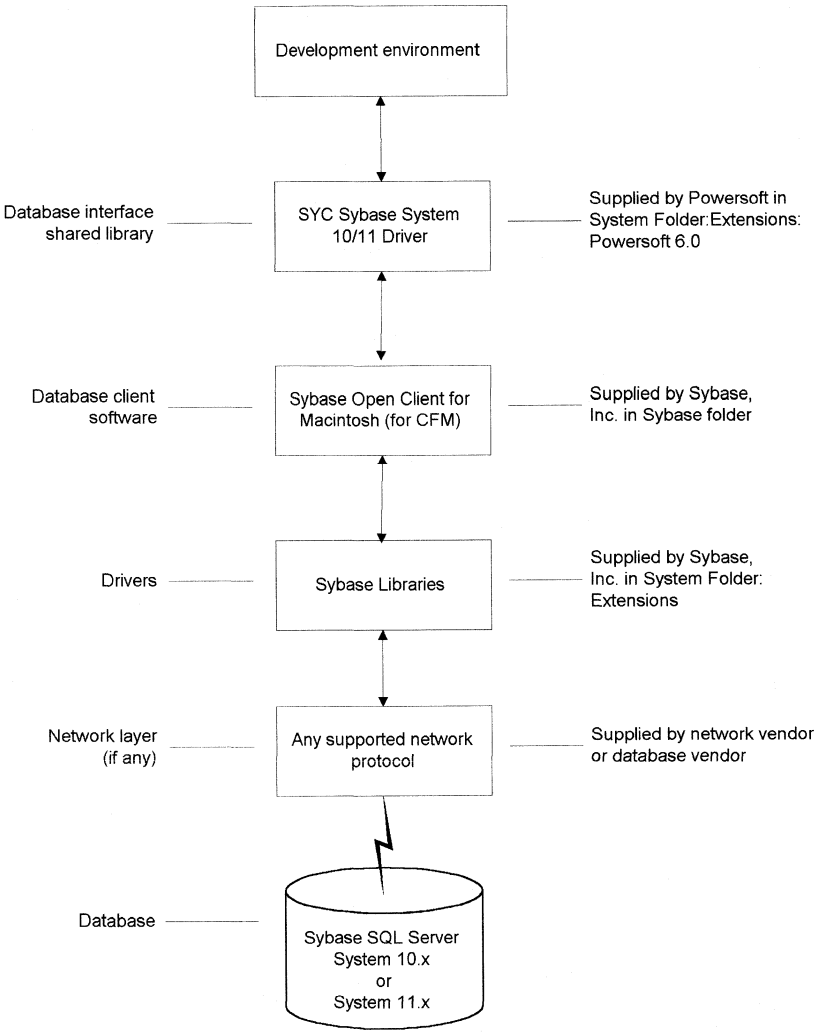
Accessing System 10.x or System 11.x on Windows



What to do next

FOR INFO For instructions on preparing your System 10.x or System 11.x database for use with PowerBuilder on the Windows platform, see "Preparing to use the System 10.x or System 11.x database on Windows" on page 247.

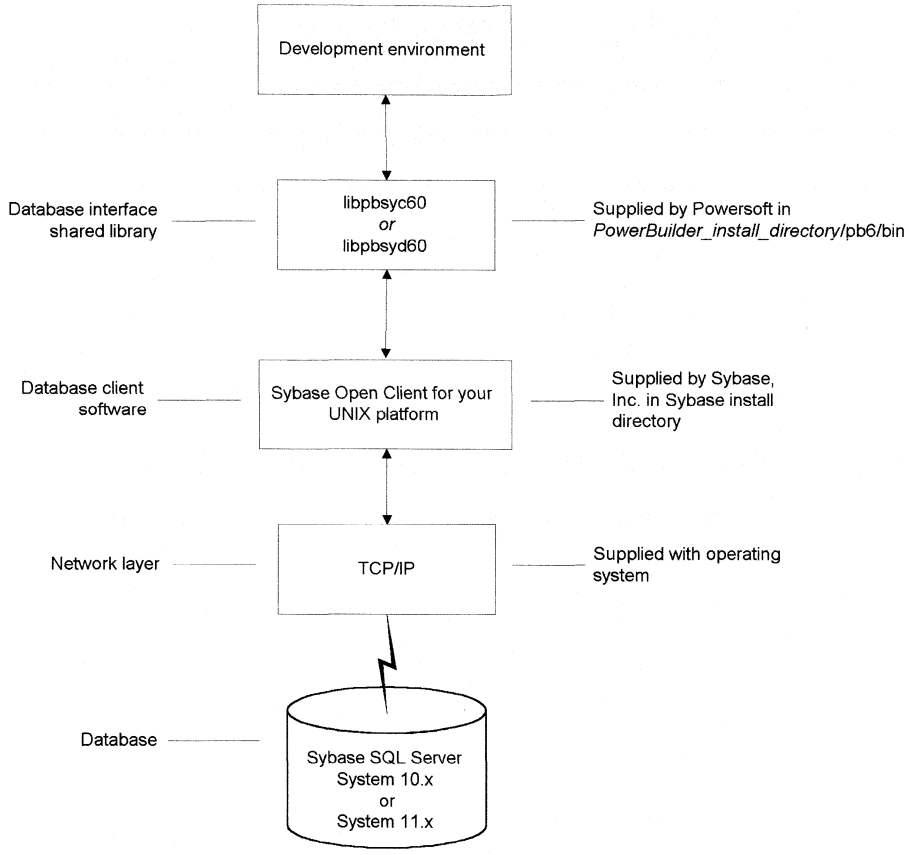
Accessing System 10.x or System 11.x on Macintosh



What to do next

FOR INFO For instructions on preparing your System 10.x or System 11.x database for use with PowerBuilder on the Macintosh platform, see "Preparing to use the System 10.x or System 11.x database on Macintosh" on page 251.

Accessing System 10 or System 11 on UNIX



What to do next

FOR INFO For instructions on preparing your System 10 or System 11 database for use with PowerBuilder on the UNIX platform, see "Preparing to use the System 10.x or System 11.x database on UNIX" on page 255.

Preparing to use the System 10.x or System 11.x database on Windows

What you do

Before you define the interface and connect to a Sybase SQL Server System 10 or System 11 database in PowerBuilder on Windows, follow these steps to prepare the database for use.

Overview of basic steps for System 10.x and System 11.x

Preparing a System 10 or System 11 database for use with PowerBuilder involves the same four basic tasks on all supported platforms. However, some of the steps for performing the tasks are different on each platform.

❖ **To prepare to use a System 10.x or System 11.x database:**

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

2 Install the Powersoft System 1.x and System 11.x database interface.

3 Verify that you can connect to SQL Server outside PowerBuilder.

4 Install the required Powersoft stored procedures in the sybsystemprocs database.

Step 1: install and configure the database server, network, and client software

❖ **To install and configure the database server, network, and client software for System 10.x or System 11.x on Windows:**

1 Make sure the Sybase System 10.x or System 11.x database software is installed on the server specified in your database profile.

You must obtain the database server software from Sybase, Inc.

FOR INFO For installation instructions, see your SQL 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 you can connect to the SQL Server database server at your site.

For example, 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 Sybase Net-Library client software.

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

3 Install the required Sybase 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, Inc. 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 SQL Server that you want to access

- The version of PowerBuilder that you are running

FOR INFO For installation instructions, see your SQL Server documentation.

4 Make sure the Sybase Open Client software is properly configured so you can connect to the System 10 or System 11 database 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 needs to find and connect to the database server at your site. You can enter and modify information in SQL.INI by using the SQLEDIT configuration utility that comes with the Open Client software.

FOR INFO For information about setting up the SQL.INI or other required configuration file, see your SQL 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:

- Powersoft SQL Server System 10 and System 11 interface DLL
- Network communication DLL (for example, NLWNSCK.DLL for Windows Sockets-compliant TCP/IP)
- Database vendor DLL (for example, WCTLIB.DLL)

Step 2: install the Powersoft System 10.x and System 11.x database interface

❖ **To install the Powersoft System 10.x and System 11.x database interface:**

- When prompted to do so by the PowerBuilder setup program, select the Powersoft System 10.x and System 11.x (CT-Lib) database interface.

FOR INFO For installation instructions, see the *PowerBuilder Installation Guide*.

Step 3: verify the System 10.x or System 11.x connection outside PowerBuilder

❖ **To verify the System 10.x or System 11.x connection on Windows:**

- Make sure you can connect to the System 10.x or System 11.x database server and log on 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 Sybase SybPing (or any other 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 on 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 Powersoft stored procedures for System 10.x or System 11.x

PowerBuilder requires you to install certain Powersoft stored procedures in the sybsystemprocs database *before* you connect to a Sybase System 10.x or System 11.x database for the first time. PowerBuilder uses these stored procedures to get information about tables and columns from the DBMS system catalog.

❖ **To install the Powersoft stored procedures:**

- Run the SQL script or scripts you need to install the Powersoft stored procedures in the sybsystemprocs database.

  FOR INFO For instructions, see "Installing Powersoft stored procedures in SQL Server databases" on page 274.

What to do next

FOR INFO For instructions on defining the System 10.x and System 11.x database interface in PowerBuilder, see "Defining the System 10.x and System 11.x database interface" on page 260.

Preparing to use the System 10.x or System 11.x database on Macintosh

What you do

Before you define the interface and connect to a Sybase System 10.x or System 11.x database in PowerBuilder on Macintosh, follow these steps to prepare the database for use.

Overview of basic steps for System 10.x and System 11.x

Preparing a System 10 or System 11 database for use with PowerBuilder involves the same four basic tasks on all supported platforms. However, some of the steps for performing the tasks are different on each platform.

❖ **To prepare to use a System 10.x or System 11.x database:**

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

2 Install the Powersoft System 10.x and System 11.x database interface.

3 Verify that you can connect to SQL Server outside PowerBuilder.

4 Install the required Powersoft stored procedures in the sybsystemprocs database.

Step 1: install and configure the database server, network, and client software

❖ **To install and configure the database server, network, and client software for System 10.x or System 11.x on Macintosh:**

1 Make sure the appropriate Sybase System 10.x or System 11.x database software is installed and running on the server machine specified in your Hosts and Interfaces files. (For more on the Interfaces file, see step 4.)

You must obtain the database server software from Sybase, Inc.

FOR INFO For installation instructions, see your Sybase or Microsoft documentation.

2 Make sure the supported network software is installed and running on your Macintosh and is properly configured so you can connect to the database server at your site.

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

Sybase Open Client for Macintosh currently supports communication using either of the following network protocols:

- **MacTCP** If you are using MacTCP, you or your system administrator must use the MacTCP control panel to set your computer's IP address and (if necessary) gateway IP address. After setting the IP address, you may need to restart your Macintosh so it recognizes the address. In addition, make sure the Hosts and Services configuration files are properly set up for your environment.
- **Communications Toolbox** If you are using Communications Toolbox, make sure a supported third-party communications tool (such as the TSS DECnet Tool) is installed and running on your Macintosh.

3 Install Sybase Open Client for Macintosh software on your Macintosh.

To connect to a System 10.x or System 11.x database in PowerBuilder on Macintosh, you must:

- **On Power Macintosh** Install Sybase Open Client for Power Macintosh Release *10.0.3 or higher.* This version supports Apple Code Fragment Manager.

The Sybase runtime installation provides all the components you need to access the database.

You must obtain Sybase Open Client for Macintosh software from Sybase, Inc.

4 Edit the Interfaces file to provide correct network information for each database server you want to access.

The Interfaces file is a text file that you edit to provide network name and address information for each database server you want to access. PowerBuilder checks this file to determine the location of the server specified in your database profile. (The server name specified in your database profile *must match exactly* the server name specified in the Interfaces file.)

The Interfaces file must contain an entry for each database server you want to access. Each entry contains information about where the server is located on your network. The format of the entry depends on the network protocol you are using to connect to the server—MacTCP or Communications Toolbox.

FOR INFO For complete instructions on editing the Interfaces file, see your Sybase Open Client for Macintosh documentation and the comments in the Interfaces file.

Here are two examples:

- **MacTCP example** This is a sample Interfaces file entry for the MacTCP protocol. *TEST* is the server name, *sybase* is the machine name, and *5003* is the port number. The values *query*, *MacTCP*, and *mac_ether* are required. A tab character must precede the query line.

```
#TEST on sybase using MacTCP
TEST
    query MacTCP mac_ether sybase 5003
```

- **Communications Toolbox example** This is a sample Interfaces file entry for the Communications Toolbox protocol. The specific format of the entry depends on the third-party communications tool you are using. *SALES* is the Sybase server name, *TSS DECnet Tool* is the communications tool, *sybase* is the DECnet node name, and *312* is the DECnet object number. The values *query*, *ctb*, and *mac_ether* are required. A tab character must precede the query line.

```
#SALES on sybase using Communications Toolbox
SALES
    query ctb mac_ether {TSS DECnet Tool}
        {Node sybase Object 312}
```

5 Use the SybaseConfig control panel to make sure the SYBASE environment variable is properly set. The SYBASE environment variable is set from the location of your Interfaces file.

FOR INFO For complete instructions on using the SybaseConfig control panel, see your Sybase Open Client for Macintosh documentation.

These steps summarize this procedure:

1 Open the SybaseConfig control panel and click the Interfaces File button.

2 Move to and select the Interfaces file in the listbox.

3 Click the Open button to set the SYBASE environment variable.

4 Close the SybaseConfig control panel.

Step 2: install the Powersoft System 10.x and System 11.x database interface

❖ **To install the Powersoft System 10 and System 11 database interface:**

- When prompted to do so by the PowerBuilder setup program, select the Powersoft System 10.x and System 11.x database interface.

  FOR INFO For installation instructions, see the *PowerBuilder Installation Guide (Macintosh).*

Step 3: verify the System 10.x or System 11.x connection outside PowerBuilder

❖ **To verify the System 10.x or System 11.x connection on Macintosh:**

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

2 Two possible ways you can verify a System 10.x or System 11.x connection are by running either of the following:

- **SybPing** The SybPing network diagnostic utility checks that you can reach the database server from your Macintosh. SybPing is included with the Sybase Net-Library software.
- **isql** The isql interactive SQL utility checks that you can access the database from your Macintosh. It lets you connect to and query the database. The isql utility is included with the Sybase Open Client for Macintosh software.

FOR INFO For instructions on using SybPing and isql, see your Sybase Open Client for Macintosh documentation.

Step 4: install the Powersoft stored procedures for System 10.x or System 11.x

PowerBuilder requires you to install certain Powersoft stored procedures in the master database *before* you connect to a System 10.x or System 11.x database for the first time. PowerBuilder uses these stored procedures to get information about tables and columns from the DBMS system catalog.

❖ **To install the Powersoft stored procedures:**

- Run the SQL script or scripts you need to install the Powersoft stored procedures in the master database.

  FOR INFO For instructions, see "Installing Powersoft stored procedures in SQL Server databases" on page 274.

What to do next

FOR INFO For instructions on defining the System 10.x and System 11.x database interface in PowerBuilder, see "Defining the System 10.x and System 11.x database interface" on page 260.

Preparing to use the System 10.x or System 11.x database on UNIX

What you do

Before you define the interface and connect to a Sybase System 10.x or System 11.x database in PowerBuilder on UNIX, follow these steps to prepare the database for use.

Overview of basic steps for System 10 and System 11

Preparing a System 10 or System 11 database for use with PowerBuilder involves the same four basic tasks on all supported platforms. However, some of the steps for performing the tasks are different on each platform.

❖ **To prepare to use a System 10 or System 11 database:**

1. Install and configure the required database server, network, and client software.
2. Install the Powersoft System 10 and System 11 database interface.
3. Verify that you can connect to SQL Server outside PowerBuilder.
4. Install the required Powersoft stored procedures in the sybsystemprocs database.

Step 1: install and configure the database server, network, and client software

❖ **To install and configure the database server, network, and client software for System 10.x or System 11.x on UNIX:**

1 Make sure the appropriate Sybase System 10 or System 11 database software is installed and running on the server machine specified in your interfaces files and database profile. (For more on the interfaces file, see step 4.)

You must obtain the database server software Sybase, Inc.

FOR INFO For installation instructions, see your Sybase documentation.

2 Make sure the required TCP/IP software is running on your workstation and is properly configured so you can connect to the database server at your site.

The TCP/IP software is supplied as part of the UNIX operating system.

3 Install and configure the Sybase Open Client/Server software on each client machine on which PowerBuilder for UNIX is installed.

For the SYC interface Install Sybase Open Client/Server Version 10.0.01, 10.0.3, 10.0.4, or 11.1.1 or higher.

For the SYD interface Install Sybase Open Client/Server Version 11.1.1 or higher thread-safe libraries.

You must obtain the Open Client/Server software from Sybase, Inc.

Configuring your Sybase Open Client/Server software involves using the *sybinit* utility to perform the following tasks:

- **Initializing the software** You must initialize (configure) each of the Sybase Open Client/Server products you installed.
- **Setting up the interfaces file** When you install Sybase Open Client/Server, the **interfaces file** is automatically created in the Sybase install directory. The interfaces file contains network name and address information for each Sybase SQL Server you want to access.

  PowerBuilder for UNIX checks the interfaces file to determine the location of the server specified in your database profile. Therefore, the server name in your database profile *must match exactly* the server name specified in your interfaces file.

You should use sybinit to edit the interfaces file to ensure that the file entries are formatted correctly.

FOR INFO For instructions on using the sybinit utility to initialize your Open Client/Server software and set up the interfaces file, see your Sybase Open Client/Server documentation.

4 Make sure the SYBASE environment variable is defined in your shell initialization file.

The SYBASE environment variable points to the directory where the Sybase Open Client/Server software is installed.

Here is a sample SYBASE definition for a C shell initialization file:

```
setenv SYBASE /export/home/sybase
```

5 Append $SYBASE/bin to the PATH environment variable in your shell initialization file.

Here is the PATH definition for a C shell initialization file:

```
setenv PATH ${SYBASE}/bin:${PATH}
```

6 Append $SYBASE/lib to the library path environment variable in your shell initialization file.

Here is the LD_LIBRARY_PATH definition for a C shell initialization file on Solaris:

```
setenv LD_LIBRARY_PATH ${SYBASE}/lib:
${LD_LIBRARY_PATH}
```

7 Make sure the PBHOME environment variable is defined in your shell initialization file.

The PBHOME environment variable points to the directory where PowerBuilder for UNIX is installed (for example, /export/home/pb6).

Here is a sample PBHOME definition for a C shell initialization file:

```
setenv PBHOME /export/home/pb6
```

8 Make sure the locales.dat file includes the locale names you need to run PowerBuilder for UNIX.

The locales.dat file contains information about the languages, character sets, and sort orders you are using with the Sybase Open Client/Server software. It is typically located in the $SYBASE/locales directory.

If a message displays when PowerBuilder for UNIX tries to connect indicating that the software cannot initialize the client library context, you may need to edit the locales.dat file to add C as a locale name.

The following example shows (in bold) the line you should add to locales.dat if this message displays:

```
[sun_svr4]
; refer to "man setlocale()"
locale = default, us_english, iso_1
; add the following line to access System 10 in
; PowerBuilder for UNIX
locale = C, us_english, iso_1
```

Step 2: install the Powersoft System 10.x and System 11.x database interface

❖ **To install the Powersoft System 10 and System 11 database interface:**

1 When prompted to do so by the PowerBuilder setup program, select the Powersoft Sybase System 10 and System 11 database interface.

FOR INFO For installation instructions, see the *PowerBuilder Installation Guide (UNIX).*

2 (Optional) Use the **which ldd** command to make sure the **ldd** command is in your path.

Typically the **ldd** command is in the /usr/bin directory.

3 (Optional) Issue an **ldd** command on the Systems 10 and 11 interface shared library you are using to make sure it can locate the appropriate shared libraries. For example, if you are using the SYC interface on Solaris, type:

```
ldd $PBHOME/bin/libpbsyc60.so
```

For example, you should see output similar to the following:

```
libct.so=>/export/home/sybase/lib/libct.so
libcs.so=>/export/home/sybase/lib/libcs.so
libtcl.so=>/export/home/sybase/lib/libtcl.so
libcomn.so=>/export/home/sybase/lib/libcomn.so
libintl.so=>/export/home/sybase/lib/libintl.so
```

All of these libraries must be present to connect to System 10 or System 11 in PowerBuilder for UNIX.

Step 3: verify the System 10.x or System 11.x connection outside PowerBuilder

❖ **To verify the System 10 or System 11 connection on UNIX:**

♦ Make sure you can connect to the System 10 or System 11 database server and database you want to access from outside PowerBuilder.

You can use the **isql** interactive SQL utility that comes with the Sybase Open Client/Server software to verify your connection.

Type the **isql** command as follows. Be sure to specify the same connection parameters you plan to use in your PowerBuilder database profile to access the System 10 or System 11 database.:

```
isql -S server_name -U user_name -P password
```

Parameter	Description
server_name	The name of the database server you want to access. This name *must match exactly* the server name specified in your interfaces file. Server names are case sensitive
user_name	The user name required to log in to the database server. User names are case sensitive
password	The password required to log in to the database server. Passwords are case sensitive

Example

Assume you are using the following connection parameters in PowerBuilder for UNIX to access a System 10 or System 11 database:

Parameter	Value
server_name	sybase
user_name	sam
password	beta2

To verify your connection to the database outside PowerBuilder for UNIX, type the following isql command:

```
isql -S sybase -U sam -P beta2
```

Step 4: install the Powersoft stored procedures for System 10.x or System 11.x

PowerBuilder requires you to install certain Powersoft stored procedures in the master database *before* you connect to a System 10 or System 11 database for the first time. PowerBuilder uses these stored procedures to get information about tables and columns from the DBMS system catalog.

❖ **To install the Powersoft stored procedures:**

- Run the SQL script or scripts you need to install the Powersoft stored procedures in the master database.

  FOR INFO For instructions, see "Installing Powersoft stored procedures in SQL Server databases" on page 274.

What to do next

FOR INFO For instructions on defining the System 10 and System 11 database interface in PowerBuilder, see "Defining the System 10.x and System 11.x database interface" on page 260.

Defining the System 10.x and System 11.x database interface

To define a connection through the Sybase Systems 10.x and 11x interface, you must create a database profile by supplying values for at least the basic connection parameters in the Database Profile Setup - Sybase System 10/System 11 dialog box. You can then select this profile at any time to connect to your database in the development environment.

❖ **To create a database profile for a Sybase System 10.x or 11.x connection:**

1 Click the Database Profile button in the PowerBar.
or
In the Database painter, select File>Connect>Setup from the menu bar.

The Database Profiles dialog box displays, listing your installed Powersoft database interfaces and defined database profiles.

2 Select the SYC or SYD interface and click New.
or
Display the popup menu for the SYC or SYD interface and select New.

The Database Profile Setup - Sybase System 10/System 11 dialog box displays.

3 On the Connection tab, type the profile name and supply values for the basic connection parameters required by the Sybase System 10 and System 11 interface.

The required connection parameters include the server, login ID, password, and database. If you omit any of these values, a dialog box displays to prompt you for the missing information when you connect to the database.

FOR INFO For information about the values you should supply for these basic connection parameters, click Help.

Server name with Open Client directory services
When you are using Open Client 11.1 or higher directory services with the SYC or SYD interface, you *must* use the syntax required by your directory service provider when specifying the server name in a database profile.

FOR INFO For information and examples, see "Using Sybase Open Client directory services" on page 265.

---

4 (Optional) On the Connection and other tabs, supply values for any additional options (DBParm parameters and SQLCA properties) you want to set to take advantage of DBMS-specific features that the interface supports.

FOR INFO For information about additional connection parameters for the Sybase System 10 and System 11 interface and how to set them, click Help.

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

The Database Profiles dialog box displays, with the new profile name highlighted. The database profile values are saved in the PowerBuilder initialization file.

Setting the PWEncrypt DBParm parameter on UNIX

Sybase Open Client/Server and PWEncrypt

In PowerBuilder on the UNIX platform, you can use various versions of the Sybase Open Client/Server client software to connect to a System 10.x or 11.x database. Version 10.0.3 or higher *supports* password encryption, but Version 10.0.1 *does not support* password encryption.

By default, the PWEncrypt DBParm parameter is set to Yes in PowerBuilder. This tells the Open Client software to encrypt your password when connecting to a System 10 or System 11 database.

What you do

Since password encryption is supported in Open Client/Server 10.0.3 or higher but not in 10.0.01, you must set PWEncrypt as follows in PowerBuilder depending on the release of Open Client/Server you are using:

Sybase Open Client/Server release	How to set PWEncrypt in PowerBuilder for UNIX
Version 10.0.1	`PWEncrypt='No'` If you do not set PWEncrypt to No to turn off PowerBuilder's default password encryption behavior, an invalid login message will display when PowerBuilder tries to connect
Version 10.0.3 or higher	You need not set PWEncrypt because the default value (Yes) is supported by Sybase Open Client/Server Version 10.0.3 or higher

How to do it

❖ **To set PWEncrypt to No in PowerBuilder for UNIX:**

- If you are using Sybase Open Client/Server Release 10.0.1 to access a System 10 or System 11 database, set the PWEncrypt DBParm parameter to No in *either* of the following ways:
  - **In a database profile** Clear the Encrypt Password checkbox on the Network tab in the Database Profile Setup dialog box.
  - **In a PowerBuilder application script** To set PWEncrypt to No in a PowerBuilder application, type the following in your PowerBuilder application script:

```
SQLCA.dbParm = " PWEncrypt = 'No' "
```

Using Sybase Open Client security services

The Sybase System 10 and System 11 interfaces (SYC or SYD) provide several DBParm parameters that support Sybase Open Client 11.1 or higher 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 Sybase Open Client security services.

What are Open Client security services?

Open Client 11.1 or higher **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 INFO For information about the third-party security mechanisms and operating system platforms that Powersoft has tested with Open Client 11.1 security services, see the Release Notes.

Requirements for using Open Client security services

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

- Sybase SQL Server 10.x or 11.x through Open Client 11.1

  You are accessing a Sybase SQL Server 10.x or 11.x database server using Sybase Open Client Client-Library (CT-Lib) 11.1 or higher software

- 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 INFO For information about the third-party security mechanisms and operating system platforms that Powersoft has tested with Open Client security services, see the Release Notes.

- Can access the secure server outside PowerBuilder

  You must be able to access a secure SQL Server 10.x or 11.x server using Sybase Open Client 11.1 or higher software from outside PowerBuilder.

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

- Powersoft database interface

  You are using the SYC Sybase System 10 and System 11 interface on all platforms or the SYD Sybase Systems 10 and 11 distributed application interface on UNIX to access the database.

- Release DBParm parameter set to 11

  You have set the Release DBParm parameter to 11 to specify that your application should use Open Client CT-Lib 11.x behavior.

  FOR INFO For instructions, see Release (Sybase System 10 and System 11) on page 505.

- 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 263, 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 Sybase Open Client 11.1 or higher 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 INFO For instructions on setting these DBParm parameters, see their descriptions in Chapter 7, "DBParm Parameters".

Per-packet security DBParms

The following per-packet security DBParm parameters correspond to Sybase Open Client 11.1 or higher 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 sever.

Sec_Confidential
Sec_Data_Integrity
Sec_Data_Origin
Sec_Replay_Detection
Sec_Seq_Detection

FOR INFO For instructions on setting these DBParm parameters, see their descriptions in Chapter 7, "DBParm Parameters".

Using Sybase Open Client directory services

The System 10 and System 11 interfaces (SYC or SYD) provide several DBParm parameters that support Sybase Open Client 11.1 or higher 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 Sybase Open Client directory services.

What are Open Client directory services?

Open Client 11.1 or higher **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 INFO For information about the third-party directory service providers and operating system platforms that Powersoft has tested with Open Client 11.1 directory services, see the Release Notes.

Requirements for using Open Client directory services

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

- Sybase Systems 10.x or 11.x through Open Client 11.1

  You are accessing a Sybase Systems 10.x or 11.x database server using Sybase Open Client Client-Library (CT-Lib) 11.1 or higher software

- Directory service provider and driver

  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 INFO For information about the directory service providers and operating system platforms that Powersoft has tested with Open Client directory services, see the Release Notes.

- Can access the secure server outside PowerBuilder

  You must be able to access a secure SQL Server 10.x or 11.x server using Sybase Open Client 11.1 or higher software from outside PowerBuilder.

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

- Powersoft database interface

  You are using the SYC Sybase System 10 and System 11 interface on all platforms or the SYD Sybase Systems 10 and 11 distributed application interface on UNIX to access the database.

- Correct server name specification

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

- Release DBParm parameter set to 11

  You have set the Release DBParm parameter to 11 to specify that your application should use Open Client CT-Lib 11.x behavior.

  FOR INFO For instructions, see Release (Sybase System 10 and System 11) on page 505.

- Directory service provider and driver support requested service

  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 Sybase 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 on page 452.)

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\SYS11NT
DIT base: SALES:software\sybase\server
Server name:SYS11NT

❖ **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 (\).

  SYS11NT

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

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

```
SQLCA.ServerName = "SYS11NT"
```

  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/SYS11
DIT base: /.../boston.sales/dataservers
Server name:sybase/SYS11

❖ **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/SYS11

❖ **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/SYS11"
```

  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: SYS11@sales@chicago
DIT base: chicago
Server name:SYS11@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 @.

  SYS11@sales

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

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

```
SQLCA.ServerName = "SYS11@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=SYS11.OU=miami.OU=sales.O=sybase
DIT base: OU=miami.OU=sales.O=sybase
Server name:SYS11

❖ **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=.

  SYS11

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

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

```
SQLCA.ServerName = "SYS11"
```

  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 266, 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 Sybase Open Client 11.1 or higher directory services connection parameters:

DS_Alias
DS_Copy
DS_DitBase
DS_Failover
DS_Principal
DS_Provider
DS_TimeLimit

FOR INFO For instructions on setting these DBParm parameters, see their descriptions in Chapter 7, "DBParm Parameters".

Using PRINT statements in SQL Server stored procedures

The SYC Sybase Systems 10 and 11 database interface (on all platforms) and the SYD Sybase Systems 10 and 11 distributed application interface (on UNIX) allow you to use PRINT statements in your stored procedures for debugging purposes.

Unlike the SQL Server 4.x (SYB or SYT) interface, the Sybase Systems 10 and 11 interfaces do *not* treat PRINT statements as fatal errors. This means, for example, that if you turn on Database Trace when accessing the database through the SYC or SYD interface, PRINT messages appear in the trace log but they do not return errors or cancel the rest of the stored procedure.

What to do next

FOR INFO For instructions on connecting to the database, see "Connecting to a database" on page 295.

Creating the Powersoft repository in DB2 databases

This section describes how PowerBuilder creates the Powersoft repository (Powersoft system tables) in your DB2 database to store extended attribute information. It then explains why you may want to use the DB2SYSPB.SQL script to create the repository 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 Powersoft database interfaces:

- Sybase InformationConnect DB2 Gateway interface
- Sybase Net-Gateway for DB2 interface

Creating the repository

When you create or modify a table in PowerBuilder, the information you provide is stored in five Powersoft system tables in your database. These system tables, known collectively as the **repository**, contain extended attribute information such as the text to use for labels and column headings, validation rules, display formats, and edit styles. (The Powersoft system tables are different from the system tables provided by your DB2 database.)

By default, the Powersoft repository is created automatically the first time a user connects to the database using PowerBuilder.

FOR INFO For more about the Powersoft repository and the tables it contains, see "About the Powersoft repository" on page 289.

❖ **To ensure that the Powersoft repository is 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 on as the database owner, database administrator, system user, system administrator, or system owner, as specified by your DBMS.

Using the DB2SYSPB.SQL script

Reasons to use

If you are a system administrator at a DB2 site, you may prefer to create the Powersoft repository outside PowerBuilder for the following 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 repository tables, it places them in the default tablespace. This may not be appropriate for your needs.

What you do

To create the Powersoft repository yourself, you can run the DB2SYSPB.SQL script outside PowerBuilder. This script contains SQL commands that create and initialize the repository 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. The \SERVER directory is also on Disk 1 of the Deployment Kit.

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 Powersoft repository 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 (drive D).

❖ **To create the Powersoft repository in a DB2 database outside PowerBuilder:**

1. Log on to the database server or gateway as the system administrator.
2. Insert the PowerBuilder CD-ROM into the computer's CD-ROM drive.

   The DB2SYSPB.SQL script you need to create the repository is in the \SERVER directory on the CD-ROM.
3. Use any text editor to modify D:\SERVER\DB2SYSPB.SQL for your environment. You can do any of the following:
   - Change all instances of PBOwner to another name. (You can also leave the table owner as PBOwner, which is the default.)

     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 INFO For instructions, see PBCatalogOwner on page 494.

4 Save any changes you made to the DB2SYSPB.SQL script.

5 Execute the DB2SYSPB.SQL script from the database server or gateway by using the SQL tool of your choice.

The DB2SYSPB.SQL script creates the Powersoft repository tables with the specified owner and tablespace.

Installing Powersoft stored procedures in SQL Server databases

This section describes how to install Powersoft stored procedures in a SQL Server database by running SQL scripts that Powersoft provides for this purpose.

You *must* run these scripts outside PowerBuilder *before* connecting to a SQL Server database for the first time through any of the following Powersoft database interfaces:

- SQL Server 4.x (SYB or SYT DBMS identifiers)
- Sybase Systems 10.x and 11.x (SYC or SYD DBMS identifier)

Not required for Microsoft SQL Server 6.x database interface
Unlike other Powersoft SQL Server database interfaces, the Microsoft SQL Server 6.x database interface (MSS DBMS identifier) does *not* require you to install Powersoft stored procedures in the database before connecting to SQL Server 6.x in PowerBuilder.

The Microsoft SQL Server 6.x database interface uses SQL Server 6.x catalog stored procedures to get information about tables and columns from the SQL Server catalog. It does not need the Powersoft stored procedures to do this.

Therefore, the information in this section *does not apply* when you are using the Powersoft SQL Server 6.x database interface.

What are the Powersoft stored procedure scripts?

What you do

In order to work with a SQL Server database in PowerBuilder, you or your system administrator must install certain Powersoft stored procedures in the database *before* you connect to SQL Server from these products *for the first time*.

You must run the Powersoft stored procedure scripts only once per database server, and not before each PowerBuilder session. If you have already installed the Powersoft stored procedures in your SQL Server database before connecting in PowerBuilder on any supported platform, you need *not* install the stored procedures again before connecting in PowerBuilder on a different platform.

Powersoft stored procedures

A **stored procedure** is a group of pre-compiled and pre-optimized SQL statements that performs some database operation. Stored procedures reside on the database server where they can be accessed as needed.

PowerBuilder uses the Powersoft stored procedures to get information about tables and columns from the SQL Server system catalog. (The Powersoft stored procedures are different from the stored procedures you may create in your database.)

Four SQL scripts

Powersoft provides four SQL script files for installing the required stored procedures in the master database (for SQL Server 4.x) or sybsystemprocs database (for Sybase System 10.x and System 11.x):

Script	Use for
PBSYB.SQL	SQL Server 4.x databases
PBSYBRT.SQL	SQL Server 4.x application deployment
PBSYC.SQL	Sybase System 10.x or System 11.x databases
PBSYC2.SQL	Sybase System 10.x or System 11.x databases to restrict the Select Tables list

Where to find the scripts

The location of the Powersoft stored procedure scripts depends on the PowerBuilder platform you are using.

Platform	Location	Comments
Windows	\SERVER directory on the PowerBuilder CD-ROM or on Disk 1 of the Deployment Kit	The \SERVER directory contains server-side installation components. Its contents is *not installed* with PowerBuilder on your computer
Macintosh	**PowerBuilder** Database Extras folder on the last disk of the Deployment Kit **InfoMaker** Database Extras folder on the last disk of the product CD	The Database Extras folder is *not installed* with PowerBuilder on your computer

Platform	Location	Comments
UNIX	pb6/bin directory in the PowerBuilder install directory	The pbsyb.sql and pbsybrt.sql scripts are *automatically installed* if you install the SQL Server 4.x database interface The pbsyc.sql and pbsyc2.sql scripts are *automatically installed* if you install either the SYC or SYD Sybase System 10.x or 11.x database interface

PBSYB.SQL script

What it does

The PBSYB.SQL script contains SQL code that drops all existing Powersoft stored procedures in the SQL Server 4.x master database and then recreates them.

When to run it

Before you connect to a SQL Server 4.x database in PowerBuilder *for the first time* using the SYB or SYT DBMS identifiers, you or your database administrator *must run* the PBSYB.SQL script once per database server into the master database.

Run PBSYB.SQL if the server at your site will be *accessed by anyone using the* PowerBuilder *development environment.*

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

FOR INFO For instructions on running PBSYB.SQL, see "How to run the scripts" on page 281.

Stored procedures it creates

The PBSYB.SQL script creates the following Powersoft stored procedures in the SQL Server 4.x master database. The procedures are listed in the order in which the script creates them.

PBSYB.SQL stored procedure	What it does
sp_pbcolumn	Lists the columns in a specified table
sp_pbdb	Retrieves the names of all databases available for this server
sp_pbindex	Retrieves information about all indexes for a specified table

PBSYB.SQL stored procedure	What it does
sp_pbproc	Lists available stored procedures
sp_pbhelprotect	Retrieves security information
sp_pbtable	Retrieves information about all tables in a database or about a specified table
sp_pbtext	Retrieves the text of a stored procedure from the SYSCOMMENTS table
sp_pbprimarykey	Lists the columns in a table's primary key
sp_pbforeignkey	Lists all foreign keys associated with a specified table
sp_pbfktable	Lists the tables that reference a specified table

PBSYBRT.SQL script

What it does

The PBSYBRT.SQL script is a subset of the PBSYB.SQL script and is used for application deployment. PBSYBRT.SQL contains SQL code that drops a subset of the existing Powersoft stored procedures in the SQL Server 4.x master database and then recreates them.

When to run it

Running PBSYBRT.SQL is optional. Before you connect to a SQL Server 4.x database in PowerBuilder for the first time using the SYB or SYT DBMS identifiers, you or your database administrator *can run* PBSYBRT.SQL once per database server into the master database.

Run PBSYBRT.SQL if the server at your site will be *accessed only by deployment machines*.

FOR INFO For instructions on running PBSYBRT.SQL, see "How to run the scripts" on page 281.

Should you run PBSYBRT.SQL or PBSYB.SQL?

Since PBSYBRT.SQL installs a subset of the procedures in PBSYB.SQL, it is unnecessary to run PBSYBRT.SQL on database servers against which you've already run PBSYB.SQL.

If your database server has limited disk space and will be accessed only by deployment machines, you may want to run PBSYBRT.SQL instead of PBSYB.SQL.

However, if you have no disk space limitations, it may be simpler to run PBSYB.SQL on the database server and all servers at your site that will deploy PowerBuilder applications. PBSYB.SQL installs all Powersoft stored procedures required for SQL Server 4.x application development and deployment.

Stored procedures it creates

The PBSYBRT.SQL script creates the following Powersoft stored procedures in the SQL Server 4.x master database. The procedures are listed in the order in which the script creates them.

PBSYBRT.SQL stored procedure	What it does
sp_pbindex	Retrieves information about all indexes for a specified table
sp_pbtable	Retrieves information about all tables in a database or about a specified table
sp_pbprimarykey	Lists the columns in a specified table's primary key

PBSYC.SQL script

What it does

The PBSYC.SQL script contains SQL code that drops all existing Powersoft stored procedures in the Sybase SQL Server System 10.x or System 11.x sybsystemprocs database and then recreates them.

The PBSYC.SQL script uses the sybsystemprocs database to hold the Powersoft stored procedures. This database is created when you install Sybase System 10.x or System 11.x.

When to run it

Before you connect to a Sybase System 10 or System 11 database in PowerBuilder *for the first time* using the SYC or SYD 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 Powersoft stored procedures in the sybsystemprocs database on your server, you need not rerun the script to install the stored procedures again.

FOR INFO For instructions on running PBSYC.SQL, see "How to run the scripts" on page 281.

Stored procedures it creates

The PBSYC.SQL script creates the following Powersoft stored procedures in the Sybase System 10.x or System 11.x sybsystemprocs database. The procedures are listed in the order in which the script creates them.

PBSYC.SQL stored procedure	What it does
sp_pb60column	Lists the columns in a table
sp_pb60primarykey	Lists the columns in a table's primary key
sp_pb60pkcheck	Determines whether a table has a primary key
sp_pb60fktable	Lists the tables that reference the current table
sp_pb60foreignkey	Lists all foreign keys associated with a specified table
sp_pb60extcat	Checks the status of the PowerBuilder catalog (Powersoft repository)
sp_pb60procdesc	Retrieves a description of the argument list for a specified stored procedure
sp_pb60proclist	Lists available stored procedures If the SystemProcs DBParm parameter is set to 1 or Yes (the default), sp_pb60proclist displays both system stored procedures and user-defined stored procedures. If SystemProcs is set to 0 or No, sp_pb60proclist displays only user-defined stored procedures
sp_pb60text	Retrieves the text of a stored procedure from the SYSCOMMENTS table
sp_pb60db	Retrieves the names of all databases available for this server
sp_pb60table	Retrieves information about *all* tables in a database, including those for which the current user has no permissions PBSYC.SQL contains the default version of sp_pb60table. If you want to replace the default version of sp_pb60table with a version that restricts the table list to those tables for which the user has SELECT permission, you can run the PBSYC2.SQL script, described in "PBSYC2.SQL script" next
sp_pb60index	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 recreates one Powersoft stored procedure in the Sybase System 10.x or System 11.x sybsystemprocs database: a replacement version of sp_pb60table.

The default version of sp_pb60table is installed by the PBSYC.SQL script. PowerBuilder uses the sp_pb60table 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_pb60table with a new version that displays a restricted table list including only the following:

- Tables and views owned by the current user
- Tables and views for which the current user has SELECT authority
- Tables and views for which the current user's group has SELECT authority
- Tables and views for which SELECT authority was granted to PUBLIC

When to run it

If you are accessing a Sybase System 10 or System 11 database using the SYC or SYD DBMS identifier in PowerBuilder, *you must first run PBSYC.SQL* once per database server to install the required Powersoft stored procedures in the sybsystemprocs database.

After you run PBSYC.SQL, you can optionally run PBSYC2.SQL if you want to replace sp_pb60table 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 INFO For instructions on running PBSYC2.SQL, see "How to run the scripts" on page 281.

Stored procedure it creates

The PBSYC2.SQL script creates the following Powersoft stored procedure in the Sybase System 10 or System 11 sybsystemprocs database:

PBSYC2.SQL stored procedure	What it does
sp_pb60table	Retrieves information about those tables in the database for which the current user has SELECT permission This version of sp_pb60table replaces the default version of sp_pb60table installed by the PBSYC.SQL script

How to run the scripts

There are various ways to run the Powersoft stored procedure scripts, depending on the SQL tools you have available. The following table lists some of the tools that you can use to run the scripts outside PowerBuilder.:

You can use this tool	To run these scripts	On these platforms
ISQL	PBSYB.SQL PBSYBRT.SQL PBSYC.SQL PBSYC2.SQL	Windows Macintosh UNIX
WISQL	PBSYB.SQL PBSYBRT.SQL PBSYC.SQL PBSYC2.SQL	Windows

The following sections describe how to run the Powersoft stored procedure scripts using each of these tools.

Using ISQL to run the stored procedure scripts

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

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

❖ **To use ISQL to run the Powersoft stored procedure scripts:**

1 Connect to the SQL Server database as the system administrator. The database to which you connect depends on the script you are running, as follows:

To run this script	Connect to this SQL Server database
PBSYB.SQL	master
PBSYBRT.SQL	master
PBSYC.SQL	sybsystemprocs
PBSYC2.SQL	sybsystemprocs

2 Open one of the following files containing the Powersoft stored procedure script you want to run:

- PBSYB.SQL
- PBSYBRT.SQL
- PBSYC.SQL
- PBSYC2.SQL

FOR INFO For information about the location of the Powersoft stored procedure scripts on different PowerBuilder platforms, see "Where to find the scripts" on page 275.

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 *not* change this user ID
SERVERNAME	The name of the computer running the SQL 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 SQL Server installation creates the sa user ID without a password. If you changed the password for sa during the installation, replace *password* with your new password

For example, if you are using PowerBuilder on Windows 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
```

On UNIX

If you are running isql on the UNIX platform, use dashes (-) instead of slashes (/) in the isql command, as follows:

isql -U sa -S *server_name* **-i** *pathname* **-P** *password*

Using WISQL to run the stored procedure scripts on Windows

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

FOR INFO For complete instructions on using WISQL, see your Sybase Open Client documentation.

❖ **To use WISQL to run the Powersoft stored procedure scripts:**

1 Start the WISQL utility.

2 Open a connection to the SQL Server database as the system administrator. The database to which you connect depends on the script you are running.

To run this script	Connect to this SQL Server database
PBSYB.SQL	master
PBSYBRT.SQL	master
PBSYC.SQL	sybsystemprocs
PBSYC2.SQL	sybsystemprocs

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

- PBSYB.SQL
- PBSYBRT.SQL

- PBSYC.SQL
- PBSYC2.SQL

FOR INFO For information about the location of the Powersoft stored procedure scripts on different PowerBuilder platforms, see "Where to find the scripts" on page 275.

4 Delete the **use master** or **use sybsystemprocs** command and the **go** command at the beginning of each script.

WISQL requires that you issue the **use master** or **use sybsystemprocs** command by itself, with no other SQL commands following it. When you open a connection to the master or sybsystemprocs database in step 2, you are in effect issuing the **use master** or **use sybsystemprocs** 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 Powersoft stored procedure script *before* executing the script.

Before executing this script	Delete these lines
PBSYB.SQL	`use master` `go`
PBSYBRT.SQL	`use master` `go`
PBSYC.SQL	`use sybsystemprocs` `go`
PBSYC2.SQL	`use sybsystemprocs` `go`

5 Execute all of the statements in the SQL script.

6 Exit the WISQL session.

CHAPTER 4

Managing Database Connections

Where you are

(*Optional*) Get an introduction to database connections
Prepare to use the data source or database
Install the ODBC driver or Powersoft database interface
Define the data source or database
> Connect to the data source or database
(*Optional*) Set additional connection parameters
(*Optional*) Troubleshoot the data connection

About this chapter

After you prepare the database, install the driver or interface, and define the data source or 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 ODBC data source definitions and database profiles, and share database profiles.

Contents

Terminology

In this chapter, the term **database** refers to *both* of the following unless otherwise specified:

- An ODBC data source that you access with the Powersoft ODBC interface and appropriate ODBC driver
- A database or DBMS that you access with the appropriate Powersoft database interface

Before you begin

Before using the procedures in this chapter to connect to your database, make sure you have done *both* of the following:

- Prepared the database and defined the ODBC data source or Powersoft database interface.

  FOR INFO For instructions, see Chapter 2, "Using ODBC Data Sources and Drivers" or Chapter 3, "Using Powersoft Database Interfaces".

- Installed the ODBC driver or Powersoft database interface required to access the data.

  FOR INFO For instructions, see the *PowerBuilder Installation Guide*.

  FOR INFO For more about using an ODBC driver supplied by a vendor other than Powersoft, see "Using the Powersoft ODBC interface" on page 19.

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

Database connections occur at different times in PowerBuilder, depending on the product you are using and whether you are developing or executing an application.

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 the values in the [Database] section of the PowerBuilder initialization file. When you connect to a database, PowerBuilder copies the connection parameters you specified to the [Database] section of the PowerBuilder initialization file, overwriting the current values in the [Database] section.

FOR INFO For instructions on using a particular painter in PowerBuilder, see the *PowerBuilder User's Guide*.

What's in this book

This book describes how to connect to your database when you are working in the PowerBuilder development environment to create an application.

FOR INFO 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 your PowerBuilder initialization file that defines a connection to a particular database in the PowerBuilder development environment.

Database profiles are created as part of the process of defining ODBC data sources and Powersoft database interfaces. When you define an ODBC data source in PowerBuilder (as described in Chapter 2, "Using ODBC Data Sources and Drivers"), a database profile is automatically created for the data source. When you define a Powersoft database interface (as described in Chapter 3, "Using Powersoft Database Interfaces"), you create the database profile yourself.

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.

Because database profiles are created when you define your data and are stored in the PowerBuilder initialization file, 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.

Initialization files on different platforms

The way you create and use database profiles is the same on *all* supported PowerBuilder platforms. However, depending on the platform you are using, the initialization files where profiles are stored have different names and locations.

Windows platforms	Macintosh platform	UNIX platforms
PB.INI in PowerBuilder product directory	System Folder:Preferences:Powersoft 6.0 Preferences:PowerBuilder Preferences	$HOME/.pb.ini

About the Powersoft repository

By default, PowerBuilder creates five Powersoft system tables the first time it connects to a database. These five tables, known collectively as the Powersoft **repository**, 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 Powersoft repository is created with the proper access rights
- Display and open a Powersoft repository table
- Understand the kind of information stored in the Powersoft repository
- Control repository access

Logging on to your database for the first time

By default, PowerBuilder creates the Powersoft repository the first time you connect to a database with PowerBuilder.

To ensure that PowerBuilder creates the repository 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 on with the proper authority.

❖ **To ensure proper creation of the Powersoft repository:**

- 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 on as the database owner, database administrator, system user, system administrator, or system owner, as specified by your DBMS.

Displaying the repository

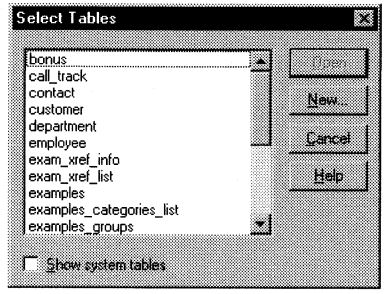
PowerBuilder maintains the Powersoft repository automatically whenever you change the information for a table or column. The Powersoft system tables in the repository are different from the system tables provided by your DBMS.

If you want, you can display and open Powersoft repository tables in the Database painter just like other tables.

❖ **To display the Powersoft repository tables:**

1 Open the Select Tables dialog box in the Database painter.

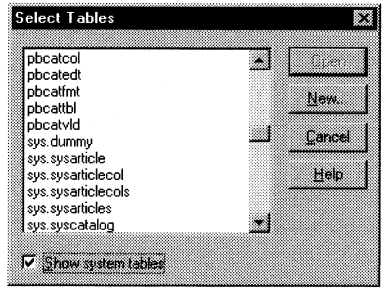
FOR INFO For instructions, see the *PowerBuilder User's Guide*.



2 Select the Show system tables checkbox.

The Powersoft repository tables and DBMS system tables display in the tables list, as follows:

- **Powersoft repository tables** The five tables in the Powersoft repository are: pbcatcol, pbcatedt, pbcatfmt, pbcattbl, and pbcatvld.
- **DBMS system tables** The system tables supplied by the DBMS usually have a DBMS-specific prefix (such as *sys* or *dbo*).



3 Open a Powersoft repository table to display its contents.

FOR INFO For instructions, see the *PowerBuilder User's Guide*.

Do not edit the repository tables
Do not change the values in the Powersoft repository tables.

Contents of the repository

PowerBuilder stores five types of extended attribute information in the Powersoft repository. (For more about the Powersoft repository, see the *PowerBuilder User's Guide*.)

This Powersoft repository table	Stores information about	That includes
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

Prefixes in repository table names

For some databases, PowerBuilder precedes the name of the repository table with a default DBMS-specific prefix. For example, the names of Powersoft repository tables have the prefix DBO in a SQL Server database (such as DBO.pbcatcol), SYSTEM in an ORACLE database (such as SYSTEM.pbcatfmt), and PBCATOWN in a DB2 database (such as PBCATOWN.pbcattbl).

The preceding table gives the base name of each repository table without the DBMS-specific prefix.

Controlling repository access

To control access to the Powersoft repository at your site, you can specify that PowerBuilder not create the repository, that the existing repository should not be updated, or that the repository is accessible only to certain users or groups.

You can control repository access by doing any of the following:

- **Setting Use Powersoft Repository** Set the Use Powersoft Repository 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 repository tables** Grant explicit permissions on the repository tables to users or groups at your site.

Setting Use Powersoft Repository or Read Only to control access

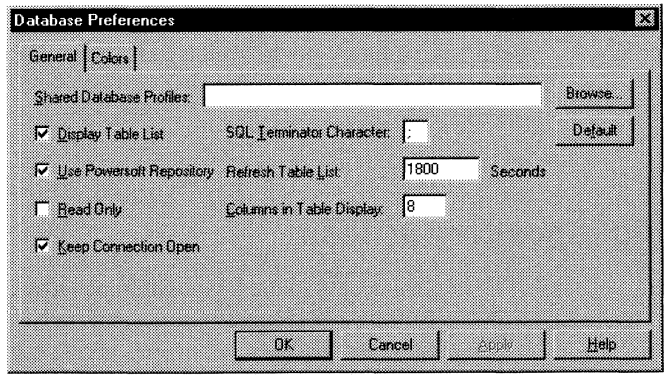
❖ **To control repository access by setting Use Powersoft Repository or Read Only:**

1 In the Database painter, click the Database Preferences button in the PainterBar.
or
Select Design>Options from the menu bar.

Database Preferences button
If your PainterBar does not include the Database Preferences button, use the customize feature to add the button to the PainterBar.

FOR INFO For instructions on customizing toolbars, see the *PowerBuilder User's Guide*.

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



2 Set values for Use Powersoft Repository or Read Only as follows:

Preference	What you do	Effect
Use Powersoft Repository	Clear the checkbox	Does not create the Powersoft repository tables if they do not exist. Instead, the DataWindow and Report painters use the appropriate default values for extended attributes (such as headers, labels, and text color) If the Powersoft repository tables already exist, PowerBuilder does not use them when you create a new DataWindow or report
Read Only	Select the checkbox	If the Powersoft repository tables already exist, PowerBuilder uses them when you create a new DataWindow or report, *but does not update them* Therefore, you *cannot* modify (update) information in the repository tables or any other database tables in the DataWindow and Report painters when the Read Only checkbox 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 [Database] section of the PowerBuilder initialization file.

Granting permissions on repository tables to control access

If your DBMS supports SQL GRANT and REVOKE statements, you can control access to the Powersoft repository tables. The default authorization for each repository table is:

GRANT SELECT, UPDATE, INSERT, DELETE ON *table* TO PUBLIC

After the repository 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.

Connecting to a database

There are two ways to establish or change a database connection in PowerBuilder:

- **Using a database profile (recommended)** You can select the database profile for the database you want to access in the Database Profiles dialog box or from the File>Connect cascading menu on the menu bar. In PowerBuilder, database profiles are created *for you* when you define an ODBC data source, and *by you* when you define a Powersoft database interface.
- **Responding to prompts** You can complete dialog boxes that prompt you for required connection parameters.

Using database profiles is the easiest way to connect to a database in PowerBuilder, especially if you often switch between different database connections.

Selecting a database profile

You can connect to a database by selecting its database profile from:

- The Database Profiles dialog box
- The File>Connect menu

Database Profiles dialog box

❖ **To connect to a database by using the Database Profiles dialog box:**

1 Click the Database Profile button in the PowerBar.
or
In the Database painter, select File>Connect>Setup from the menu bar.

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 INFO For instructions on customizing toolbars, see the *PowerBuilder User's Guide*.

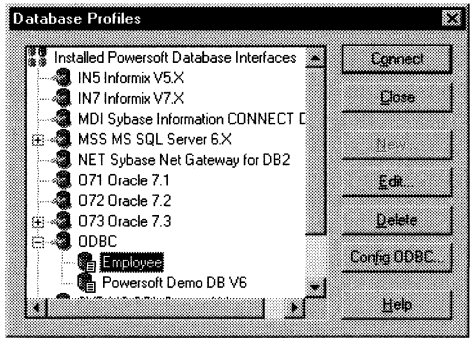
The Database Profiles dialog box displays, listing your installed Powersoft database interfaces. Those interfaces preceded by a plus (+) sign have one or more database profiles already defined.

Where the interface list comes from

When you run the Setup program, it updates the Vendors list in the [Database] section of your PowerBuilder initialization file 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
If your interface name is preceded by a plus sign, 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 popup menu for a database profile and select Connect.

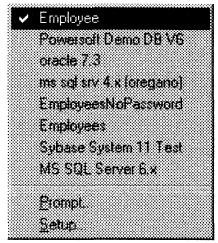
PowerBuilder connects to the specified database and returns you to the painter workspace.

File>Connect menu

❖ **To connect to a database by using the File>Connect menu:**

1 In the Database painter, select File>Connect from the menu bar.

A cascading menu displays, listing the names of defined database profiles. If you are currently connected to a database, the name of its profile is checked.



2 Select the name of the database profile to which you want to connect.

PowerBuilder connects to the selected database and returns you to the painter workspace.

Responding to prompts

You can also connect to a database in PowerBuilder by responding to dialog boxes that prompt you to supply connection information.

When you connect to a database by responding to prompts, PowerBuilder writes the connection parameters you specify to the [Database] section of the PowerBuilder initialization file.

Unlike database profiles, however, connection parameters supplied in response to prompts are *not* permanently stored in the initialization file. PowerBuilder overwrites the existing values in the [Database] section with new values every time you connect to a different database.

Since connection parameters supplied in response to prompts are not permanently stored in the initialization file, you may want to use prompts only when connecting to databases you use infrequently.

Connecting to an ODBC data source through prompts

Before you can connect to an ODBC data source by responding to prompts, you must define (configure) it. You can do this in PowerBuilder on Windows and Macintosh by completing the ODBC setup dialog box for the driver you are using to access the data source.

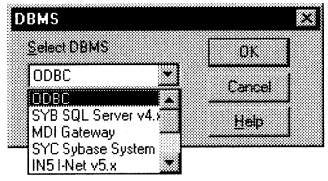
Connecting to ODBC data sources by responding to prompts is *not supported in PowerBuilder for UNIX.*

FOR INFO For instructions on defining an ODBC data source, see the section for your driver in Chapter 2, "Using ODBC Data Sources and Drivers".

❖ **To connect to a database by responding to prompts:**

1 In the Database painter, select File>Connect>Prompt from the menu bar.

The DBMS dialog box displays. The Select DBMS dropdown listbox contains the ODBC identifier and identifiers for any other Powersoft database interfaces you have installed. If you are currently connected to a database, its identifier is highlighted.



Where the DBMS identifiers come from When you install a Powersoft database interface by running the Setup program, PowerBuilder updates the Vendors list in the [Database] section of the PowerBuilder initialization file with the proper DBMS identifier for your interface. (The ODBC identifier displays in the Vendors list by default.)

The identifiers in the Vendors list display in the DBMS dialog box.

On Windows and Macintosh Iif you have installed only ODBC drivers and have *not* installed any Powersoft database interfaces, the DBMS dialog box does not display when you select File>Connect>Prompt. The first dialog box you see is the Select Data Sources dialog box (on Windows) or Data Source for Connection dialog box (on Macintosh), as shown in "ODBC example on Windows and Macintosh" next.

FOR INFO For a complete list of supported DBMS identifiers, see Appendix A, "Supported ODBC Drivers and Powersoft Database Interfaces".

2 Select the identifier of the DBMS to which you want to connect.

You cannot select ODBC on UNIX
Selecting the ODBC identifier in PowerBuilder for UNIX causes an error message to display.

3 Click OK.

One or more dialog boxes display to prompt you for information required to connect to the database. This information can include:

- Data source name (for ODBC data sources on Windows or Macintosh)
- User ID or login ID
- Password or login password
- Server name
- Database name

FOR INFO For examples of the kind of dialog boxes that can display when connecting to an ODBC data source or Powersoft database interface, see ""ODBC example on Windows and Macintosh" next" or "Powersoft database interface example" on page 301.

4 In the dialog boxes that display, supply the connection parameters required to connect to your database.

PowerBuilder connects to the database with the parameters you specify.

FOR INFO For instructions on supplying connection parameters, see the section for your data source driver in Chapter 2, "Using ODBC Data Sources and Drivers" or the section for your database interface in Chapter 3, "Using Powersoft Database Interfaces".

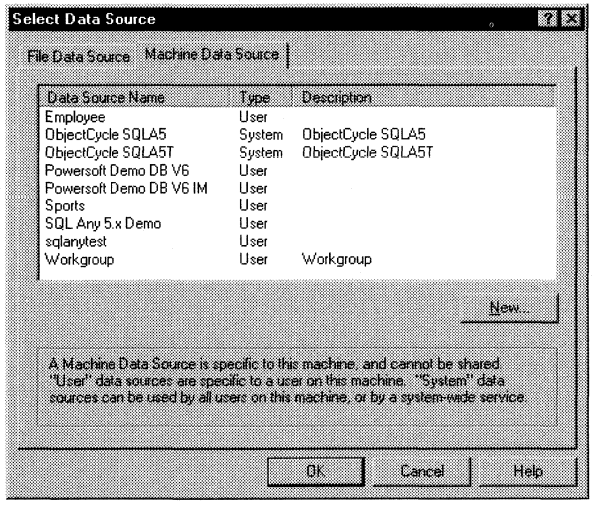
ODBC example on Windows and Macintosh

If you select ODBC in the DBMS dialog box in PowerBuilder on Windows or Macintosh, the Select Data Source dialog box (on Windows) or Data Source for Connection dialog box (on Macintosh) displays, listing all ODBC data sources you have defined.

On UNIX

Connecting to ODBC data sources by responding to prompts is *not supported* in PowerBuilder for UNIX.

To connect to an ODBC data source in PowerBuilder on Windows or Macintosh, select its name and click OK.



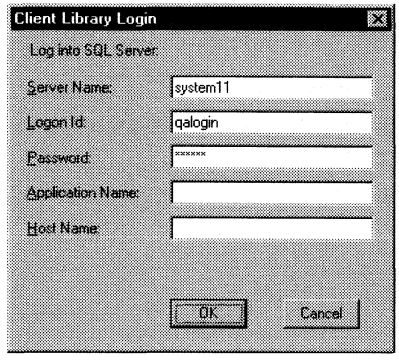
PowerBuilder Desktop

If you are using PowerBuilder Desktop, the Select Data Source dialog box (on Windows) or Data Source for Connection dialog box (on Macintosh) lists only the names of those data sources defined for *supported* ODBC drivers.

FOR INFO For a list of supported ODBC drivers, see Appendix A, "Supported ODBC Drivers and Powersoft Database Interfaces".

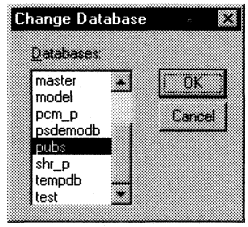
Powersoft database interface example

If you select the identifier for the Sybase System 10.x or System 11.x database interface (SYC or SYD identifier) in the DBMS dialog box, the Client Library Login dialog box displays.



After you supply the necessary connection parameters in the Login dialog box and click OK, the Change Database dialog box displays. The Change Database dialog box lists all available databases for your DBMS.

To connect to a database, select its name and click OK.



What happens when you connect

This section describes what happens in PowerBuilder when you connect to a database.

Connection parameters

When you connect to a database by selecting its database profile or by responding to prompts, PowerBuilder writes the connection parameters to the [Database] section of the PowerBuilder initialization file.

Each time you connect to a different database, PowerBuilder overwrites the existing parameters in the [Database] section with the parameters for the new database connection.

What you are 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 [Database] section of the PowerBuilder initialization file.

For example, the following portion of the [Database] section shows a current connection to the Powersoft Demo database through the ODBC interface:

```
[Database]
DBMS=ODBC
Database=Powersoft Demo DB V6
UserID=dba
DatabasePassword=
LogPassword=
ServerName=
LogId=
...
DbParm=ConnectString='DSN=Powersoft Demo DB V6;
   UID=dba;PWD=sql'
...
Vendors=ODBC,SYC Sybase SQL Server 10.x and 11.x
```

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. (The Preview tab is *not available* in the Database Profile Setup dialog box in InfoMaker.)

FOR INFO 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 ODBC data source definitions

You can easily edit or delete an existing ODBC data source definition in PowerBuilder on the Windows and Macintosh platforms.

Configure ODBC dialog box unavailable on UNIX
The Configure ODBC dialog box is unavailable in PowerBuilder on the UNIX platform. Therefore, you *cannot* use the procedures in this section to edit and delete ODBC data source definitions in PowerBuilder for UNIX.

Editing an ODBC data source definition

In PowerBuilder on Windows and Macintosh, you can edit an ODBC data source definition to change one or more of its connection parameters.

You cannot edit the ODBC data source you are connected to
You *cannot* edit the definition of the ODBC data source to which you are currently connected in PowerBuilder.

❖ **To edit an ODBC data source definition on Windows and Macintosh:**

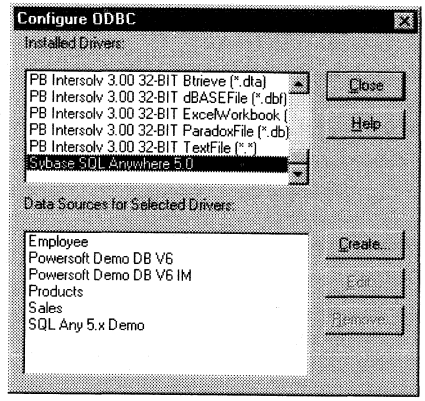
1 Click the Configure ODBC button in the PowerBar.
or
In the Database painter, select File>Configure ODBC from the menu bar.
or
In the Database Profiles dialog box, select the ODBC interface or an ODBC database profile. Click the Config ODBC button, or display the item's popup menu and select Config ODBC.

Configure ODBC button

If your PowerBar doe not include the Configure ODBC button, use the customize feature to add the button to the PowerBar.

FOR INFO For instructions on customizing toolbars, see the *PowerBuilder User's Guide*.

The Configure ODBC dialog box displays, listing the ODBC drivers installed on your computer and the data sources defined for each driver.

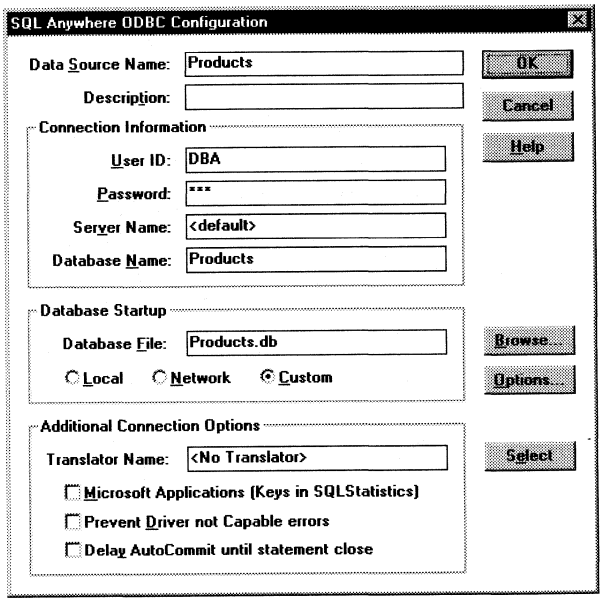


2 Select the ODBC driver that accesses the data source you want to edit.

The names of the data source definitions for that driver display in the Data Sources for Selected Drivers list.

3 Select the name of the data source definition you want to edit and click Edit.

The ODBC setup dialog box for the selected data source displays.

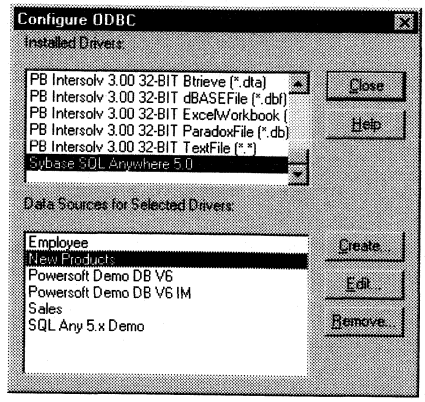


4 Edit the data source definition as required for your connection.

FOR INFO For instructions on completing the ODBC setup dialog box, see the section for your data source driver in Chapter 2, "Using ODBC Data Sources and Drivers".

5 Click OK.

The Configure ODBC dialog box displays. If you changed the data source name in step 5, the new name (New Products) displays in the data source list.



6 Repeat steps 2 through 5 if you want to edit another ODBC data source definition.

7 Click Close when you are finished editing ODBC data source definitions.

The Configure ODBC dialog box closes and you are returned to the painter workspace.

8 If you changed the data source name in step 4, edit the database profile for this data source to make sure it contains the correct name in the connect string.

FOR INFO For instructions, see "Updating a database profile when you change the ODBC data source name" next.

What happens

When you edit an ODBC data source definition, PowerBuilder updates the ODBC initialization file with the new values for this data source.

PowerBuilder does *not* update the existing database profile for this data source in the PowerBuilder initialization file.

Updating a database profile when you change the ODBC data source name

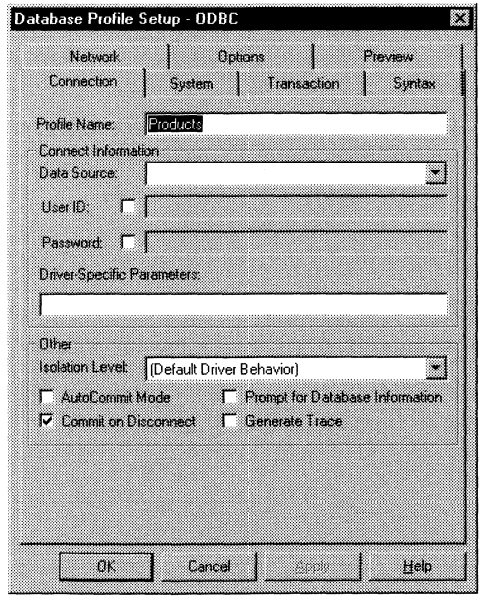
As described in the preceding section, PowerBuilder *does not update the database profile* when you edit an ODBC data source. This means that if you change the name of an ODBC data source for which PowerBuilder has already created a database profile, the connect string in the existing database profile still contains the old data source name. Since the existing database profile does not have the correct data source name, it will no longer connect to the data source.

To update the database profile so it connects to the data source, you must edit the profile to supply the correct data source name.

❖ **To update an existing database profile with the correct ODBC data source name:**

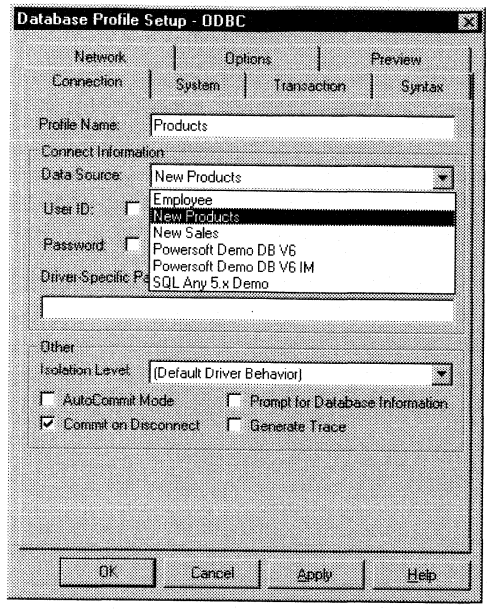
1 In the Database Profiles dialog box, select the ODBC profile you want to edit. Click Edit or select Edit from the item's popup menu.

The Database Profile Setup dialog box for the selected profile displays. Notice that the Data Source dropdown listbox is empty because the profile no longer contains the correct data source name. (In this example, assume that you changed the ODBC data source name from Products to New Products.)



2 Select the correct data source name from the Data Source dropdown listbox.

If you want, you can also edit the profile name to match the new data source name, but this is *not required* to connect to the data source.



3 Click OK in the Database Profile Setup 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.

PowerBuilder connects to the selected data source.

Deleting an ODBC data source definition

In PowerBuilder on Windows and Macintosh, you can delete an ODBC data source definition when you no longer need to access its data.

You cannot delete the ODBC data source you are connected to

You *cannot* delete the definition of the ODBC data source to which you are currently connected in PowerBuilder.

❖ To delete an ODBC data source definition on Windows and Macintosh:

1 Click the Configure ODBC button in the PowerBar.
or
In the Database painter, select File>Configure ODBC from the menu bar.
or
In the Database Profiles dialog box, select the ODBC interface or an ODBC database profile. Click the Config ODBC button, or display the item's popup menu and select Config ODBC.

Configure ODBC button
If your PowerBar does not include the Configure ODBC button, use the customize feature to add the button to the PowerBar.

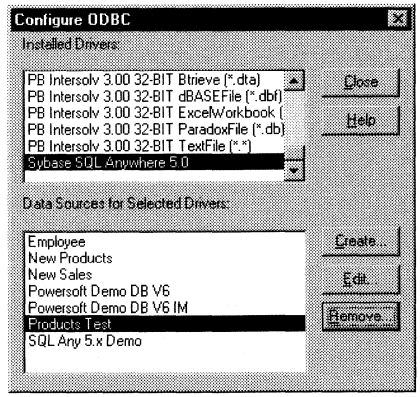
FOR INFO For instructions on customizing toolbars, see the *PowerBuilder User's Guide*.

The Configure ODBC dialog box displays, listing the ODBC drivers installed on your computer and the data sources defined for each driver.

2 Select the ODBC driver that accesses the data source you want to delete.

The names of the data source definitions for that driver display in the Data Sources for Selected Drivers list.

3 Select the name of the data source definition you want to delete.

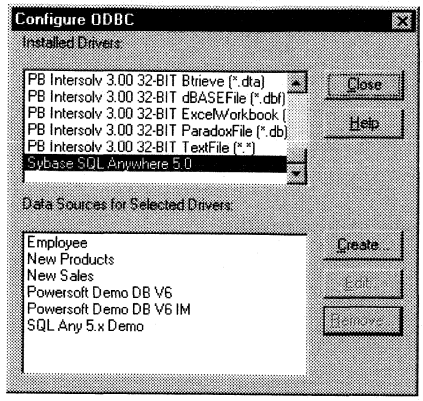


4 Click the Remove button.

A message box asks you to confirm the delete operation.

5 Click OK in the message box to delete the selected data source definition.

The Configure ODBC dialog box displays. The name of the data source definition you deleted is removed from the data source list.



6 Repeat steps 2 through 5 if you want to delete another ODBC data source definition.

7 Click Close when you are finished deleting ODBC data source definitions.

The Configure ODBC dialog box closes and you are returned to the painter workspace.

What happens

When you delete an ODBC data source definition, PowerBuilder:

- Removes the definition from the ODBC initialization file
- Removes the database profile for this data source from the PowerBuilder initialization file

Maintaining database profiles

You can easily edit or delete an existing database profile in PowerBuilder on *all* supported platforms.

Editing a database profile

You can edit a database profile to change one or more of its connection parameters.

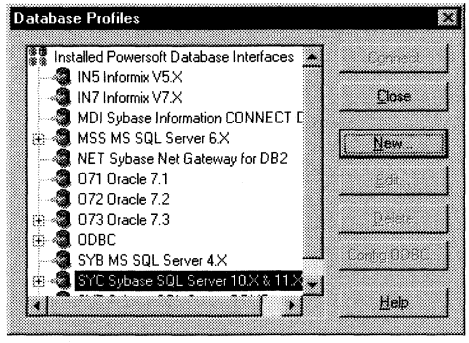
❖ **To edit a database profile:**

1 Click the Database Profile button in the PowerBar.
or
In the Database painter, select File>Connect>Setup from the menu bar.

Database Profile button
If your PowerBar does not include the Database Profile button, use the customize feature to add the button to the PowerBar.

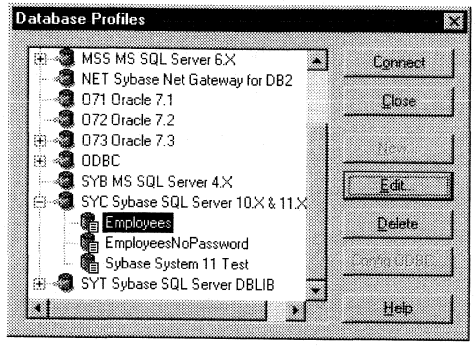
FOR INFO For instructions on customizing toolbars, see the *PowerBuilder User's Guide*.

The Database Profiles dialog box displays, listing your installed Powersoft database interfaces and existing database profiles.



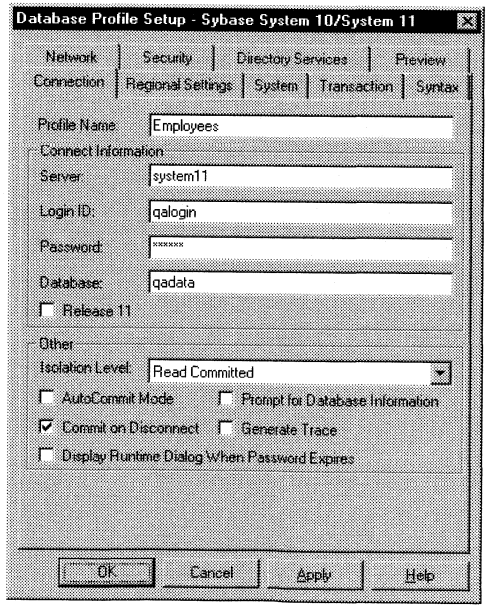
2 Click the plus sign (+) to the left of the interface you are using.
or
If your interface name is preceded by a plus sign, double-click the name.

The list expands to display the database profiles defined for your interface.



3 Select the name of the profile you want and click Edit.
or
Display a profile's popup menu and select Edit.

The Database Profile Setup dialog box for the selected profile displays.



4 Edit the profile as desired for your database connection.

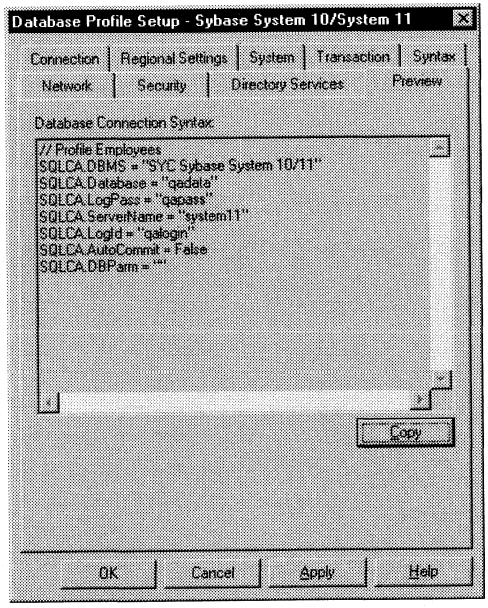
FOR INFO For information about the connection parameters for your interface and the values you should supply, click Help.

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.

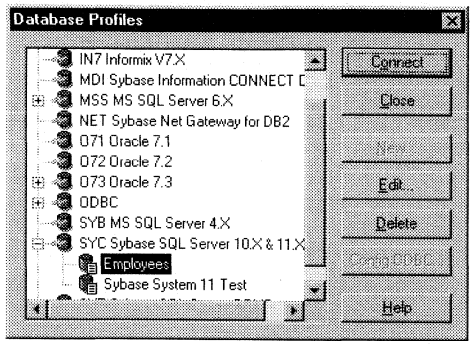
FOR INFO For instructions on using the Preview tab to help you connect in a PowerBuilder application, see the section on using transaction objects in *Application Techniques*.

Since PowerScript connection syntax does not apply to InfoMaker applications, the Preview tab is *not* available in the Database Profile Setup dialog box in InfoMaker.



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

The Database Profiles dialog box displays, with the name of the edited profile highlighted under its interface.



What happens

When you edit a database profile, PowerBuilder updates the database profile entry in the PowerBuilder initialization file.

For more information

For more about the values you should supply when editing a database profile, see the sections listed in the following table:

For more about	See
Defining an ODBC data source	The section for your driver in Chapter 2, "Using ODBC Data Sources and Drivers"
Defining a Powersoft database interface	The section for your Powersoft database interface in Chapter 3, "Using Powersoft Database Interfaces"
Changing the ODBC data source name in the DBParm connect string	"Updating a database profile when you change the ODBC data source name" on page 308
Setting DBParm parameters	Chapter 5, "Setting Additional Connection Parameters"

Deleting a database profile

You can delete a database profile when you no longer need to access its data.

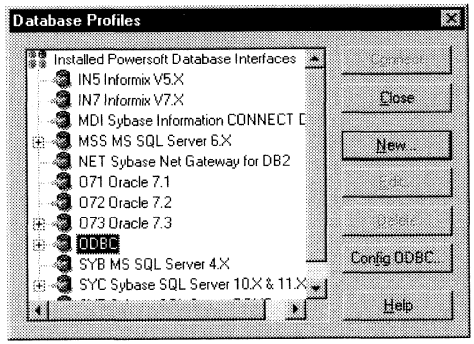
❖ **To delete a database profile:**

1 Click the Database Profile button in the PowerBar.
or
In the Database painter, select File>Connect>Setup from the menu bar.

Database Profile button
If your PowerBar does not include the Database Profile button, use the customize feature to add the button to the PowerBar.

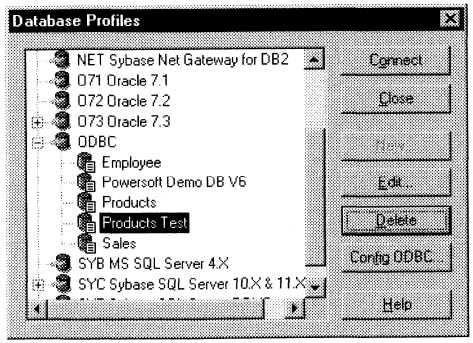
FOR INFO For instructions on customizing toolbars, see the *PowerBuilder User's Guide*.

The Database Profiles dialog box displays, listing your installed Powersoft database interfaces and existing database profiles.



2 Click the plus sign (+) to the left of the interface you are using.
or
If your interface name is preceded by a plus sign, double-click the name.

The list expands to display the database profiles defined for your interface.



3 Select the name of the profile you want and click Delete.
or
Display the popup menu for a database profile and select Delete.

The selected database profile is removed from the list for your interface.

4 Take one of the following actions:

- Click Close to close the Database Profiles dialog box. PowerBuilder remains connected to the current database.
- Select another database profile and click Connect to access that database.

What happens

When you delete a database profile, PowerBuilder removes it from the PowerBuilder initialization file.

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 recreate the database profile later if necessary without having to redefine the data source.

FOR INFO For instructions, see "Creating a profile for an existing ODBC data source" next.

Creating a profile for an existing ODBC data source

In most cases, you do not need to create a database profile for an ODBC data source. When you define the data source in PowerBuilder, the database profile is created for you automatically.

However, you may want to create or replace a database profile for an existing ODBC data source for the following reasons:

- You deleted the profile for this data source and then decide later that you want to access this data in PowerBuilder.
- You defined the database source outside PowerBuilder using a tool such as the ODBC Administrator (on Windows) or the ODBC Setup control panel (on Macintosh). When you define an ODBC data source outside PowerBuilder, you must create the database profile yourself.

When 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. You can therefore recreate the database profile without having to redefine the ODBC data source.

On Windows and Macintosh You *can use this procedure* in PowerBuilder on Windows and Macintosh to create a database profile for an existing ODBC data source.

On UNIX The Select Data Source dialog box (see step 5 below) is unavailable in PowerBuilder on the UNIX platform. Therefore, you *cannot use this procedure* in PowerBuilder for UNIX to create a database profile for an existing ODBC data source.

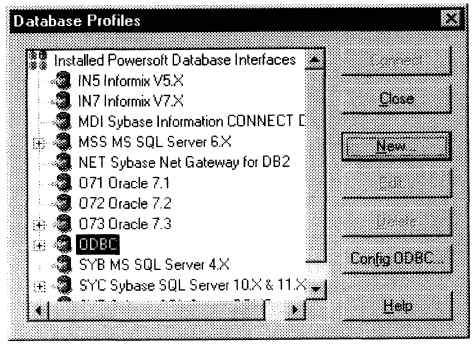
❖ **To create a database profile for an existing ODBC data source:**

1 Click the Database Profile button in the PowerBar.
or
In the Database painter, select File>Connect>Setup from the menu bar.

The Database Profiles dialog box displays, listing your installed Powersoft database interfaces and existing database profiles.

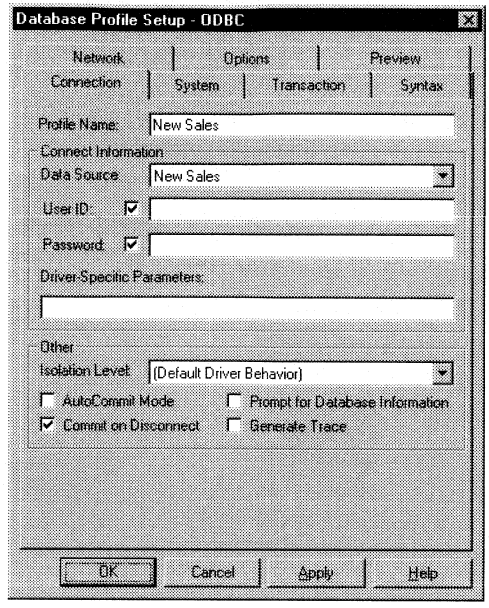
2 Select the ODBC interface and click New.
or
Display the popup menu for the ODBC interface and select New.

The Database Profile Setup - ODBC dialog box displays.



3 Type the name of your profile in the Profile Name box on the Connection tab.

4 Select the name of the ODBC data source you want to access from the Data Source dropdown listbox on the Connection tab.



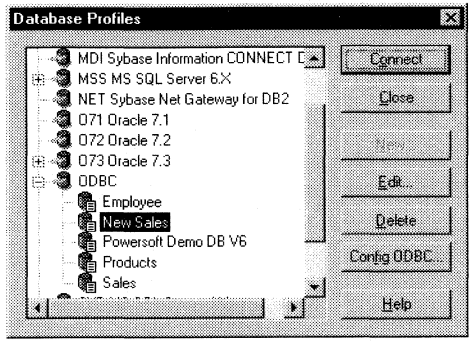
5 (Optional) On the other tab pages, supply values for any additional connection options (DBParm parameters and SQLCA properties) you want to set to take advantage of DBMS-specific features that the ODBC interface supports.

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

6 Click OK.

If you did not supply enough information in your data source definition for PowerBuilder to connect, the driver that accesses this data source prompts you for additional connection parameters.

After you supply additional connection parameters as needed, the Database Profiles dialog box displays with the new profile name highlighted under the ODBC interface. The database profile values are saved in your PowerBuilder initialization file.



7 Click Connect in the Database Profiles dialog box.

PowerBuilder connects to the selected data source.

Sharing database profiles

When you work in the PowerBuilder development environment on *any* supported platform, you can share database profiles among users on the same platform or between PowerBuilder and InfoMaker 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 the PowerBuilder or InfoMaker initialization 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 the initialization file

Where you store the PowerBuilder or InfoMaker initialization file determines how you can share the profiles it contains.

To share database profiles	Store the initialization file
Among all PowerBuilder users at your site	On a network file server accessible to all users
Between PowerBuilder and InfoMaker when both products are running on your computer	In a directory or folder on your computer (typically the PowerBuilder or InfoMaker product directory)

Initialization filename and location

FOR INFO For information about the default names and locations of the PowerBuilder and InfoMaker initialization files on different platforms, see "Initialization files on different platforms" on page 288.

What happens

When you share database profiles, PowerBuilder displays shared database profiles from the initialization file you specify as well as those from your local initialization file. For all other painter preferences, however, PowerBuilder continues to use your local initialization file.

Shared database profiles are read-only. You can select a shared profile to connect to a database but you *cannot* 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 326.)

How to do it

To set up shared database profiles in the PowerBuilder development environment, you specify the location of the initialization file containing shared profiles in the Database painter's Database Preferences property sheet.

FOR INFO For instructions, see "Setting up shared database profiles" next.

Setting up shared database profiles

Platform considerations

Database profiles are usable on any supported PowerBuilder platform. However, the ability to read profile information from the initialization file depends on the PowerBuilder platform you are using.

Sharing database profiles on Windows PowerBuilder on Windows platforms cannot read database profile values from initialization files created on other platforms (Macintosh and UNIX) unless this information contains native Windows line endings.

Therefore, make sure the profile information you are sharing *either*:

- Originates on the Windows platform, *or*
- Contains native Windows line endings converted from another platform

If the database profiles you are sharing come from initialization files created on other platforms, you must first make a copy of the initialization file and convert its line endings to Windows format. You can do this with an appropriate text conversion utility or by opening the file in the PowerBuilder for Windows text editor and saving it in the native format.

After converting the line endings to native Windows format, you can copy the database profiles into the initialization file on the Windows platform and share the profiles among Windows-based PowerBuilder and InfoMaker users.

Sharing database profiles on Macintosh PowerBuilder on the Macintosh platform can read database profile values from initialization files created on *any* supported platform.

Therefore, the profile information you are sharing can originate on the Windows or UNIX platform and contain native line endings for that platform.

Sharing database profiles on UNIX PowerBuilder on the UNIX platform cannot read database profile values from initialization files created on other platforms (Windows and Macintosh) unless this information contains native UNIX line endings.

Therefore, make sure the profile information you are sharing *either*:

- Originates on the UNIX platform, *or*
- Contains native UNIX line endings converted from another platform

If the database profiles you are sharing come from initialization files created on other platforms, you must first make a copy of the initialization file and convert its line endings to UNIX format. You can do this with an appropriate text conversion utility or by opening the file in the PowerBuilder for UNIX text editor and saving it in the native format.

After converting the line endings to native UNIX format, you can copy the database profiles into the initialization file on the UNIX platform and share the profiles among UNIX-based PowerBuilder users.

What you do

❖ **To set up shared database profiles:**

1 In the Database painter, click the Database Preferences button in the PainterBar.
or
Select Design>Options from the menu bar.

Database Preferences button
If your PainterBar does not include the Database Preferences button, use the customize feature to add the button to the PainterBar.

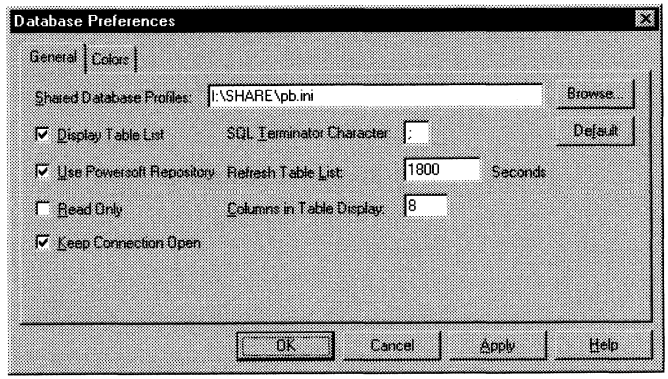
FOR INFO For instructions on customizing toolbars, see the *PowerBuilder User's Guide*.

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 PowerBuilder or InfoMaker initialization 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 initialization file location and display it in the Shared Database Profiles box

In the following example on Windows, I:\SHARE\PB.INI is the location of the initialization 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 [PB] section of your *local* PowerBuilder initialization file.

Using shared database profiles to connect

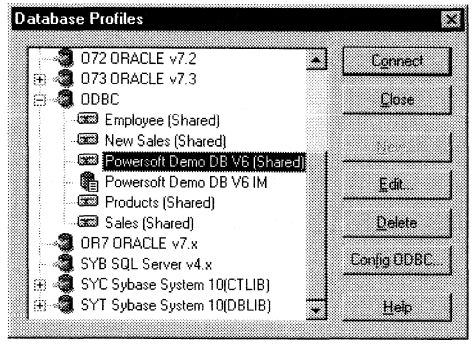
You select a shared database profile to connect to a database in the same way that you select a profile stored in your local initialization file. You can select the shared profile in the Database Profiles dialog box or from the File>Connect menu.

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
In the Database painter, select File>Connect>Setup from the menu bar.

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.

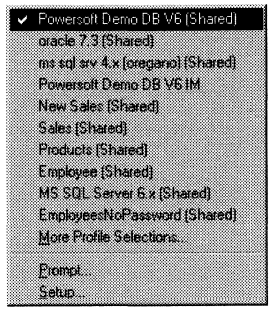
PowerBuilder connects to the selected database and returns you to the painter workspace.

File>Connect menu

❖ **To select a shared database profile from the File>Connect menu:**

1 In the Database painter, select File>Connect from the menu bar.

A cascading menu displays, listing both shared and local database profiles. Shared profiles are denoted by the word *(Shared)*.



2 Select the name of the shared profile you want to access.

PowerBuilder connects to the selected database and returns you to the painter workspace.

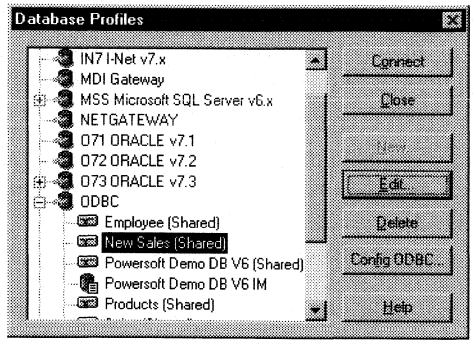
Making local changes to shared database profiles

Because shared database profiles can be accessed by multiple users running PowerBuilder or InfoMaker, 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 the local PowerBuilder initialization file on your computer.

❖ **To save changes to a shared database profile in your local PowerBuilder initialization file:**

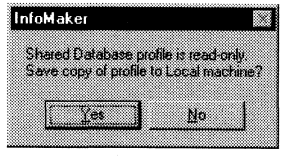
1 In the Database Profiles dialog box, select the shared profile you want to edit and click the Edit button.

Shared profiles are denoted by a network icon and the word *(Shared)*.



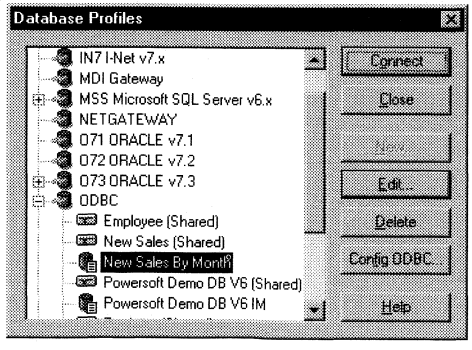
2 In the Database Profile Setup dialog box that displays, edit the profile values as needed and click OK.

A message box displays asking if you want to save a copy of the modified profile to your computer.



3 Click Yes in the message box.

PowerBuilder saves the modified profile in the local PowerBuilder initialization file on your computer.



Maintaining shared database profiles

For administrators
Read this section if you maintain the PowerBuilder or InfoMaker database profiles at your site.

If you maintain the PowerBuilder or InfoMaker database profiles 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 or InfoMaker on different platforms, it is *not* a good idea to make changes to the profiles over a network. Instead, you should make any changes locally and then make the updated profiles available 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 local PowerBuilder or InfoMaker initialization file.

FOR INFO For instructions, see "Making local changes to shared database profiles" on page 326.

2 Copy the [DBMS_PROFILES] section and any updated profile entries from your local PowerBuilder or InfoMaker initialization file to the existing initialization file containing shared profiles.

3 If they have not already done so, have each user specify the location of the new PowerBuilder or InfoMaker initialization file in the Database Preferences property sheet so they can access the updated shared profiles on their computer.

FOR INFO For instructions, see "Setting up shared database profiles" on page 322.

CHAPTER 5

Setting Additional Connection Parameters

Where you are

(*Optional*) Get an introduction to database connections
Prepare to use the data source or database
Install the ODBC driver or Powersoft database interface
Define the data source or database
Connect to the data source or database
> (*Optional*) Set additional connection parameters
(*Optional*) Troubleshoot the data connection

About this chapter

Chapters 2 and 3 describe the basic connection parameters you must supply to define an ODBC data source or Powersoft database interface. To fine-tune your database connection and take advantage of DBMS-specific features that your interface supports, you can set additional connection parameters anytime. These additional connection parameters include:

- DBParm parameters
- Database preferences

This chapter describes how to set DBParm parameters and database preferences in PowerBuilder.

Contents

Basic steps for setting connection parameters

This section gives basic steps for setting DBParm parameters and database preferences in PowerBuilder on *all* supported platforms.

❖ **To set DBParm parameters:**

1 Learn how to set DBParm parameters in the development environment or PowerBuilder application script.

FOR INFO For instructions, see "Setting DBParm parameters" on page 332.

2 Determine the DBParm parameters you can set for your Powersoft database interface.

FOR INFO For a table listing each supported Powersoft database interface (including ODBC) and the DBParm parameters you can use with that interface, see "DBParm parameters and supported database interfaces" on page 392.

3 Read the description of the DBParm parameter you want to set.

FOR INFO See the section for your DBParm parameter in Chapter 7, "DBParm Parameters". The DBParm parameters are described in alphabetical order.

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.

FOR INFO For instructions, see "Setting database preferences" on page 340.

2 Determine the database preferences you can set for your DBMS.

FOR INFO For a table listing each supported Powersoft database interface (including ODBC) and the database preferences you can use with that interface, see "Database preferences and supported database interfaces" on page 564.

3 Read the description of the database preference you want to set.

FOR INFO See the section for your database preference in Chapter 8, "Database Preferences". The database preferences are described in alphabetical order.

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 Powersoft database interface

  Each Powersoft 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, in PowerBuilder, the 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 DBParm parameter and SQLCA property values with easy-to-use checkboxes, dropdown listboxes, and textboxes. PowerBuilder generates the proper syntax automatically when it saves your database profile in the initialization file.

- 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 SQLCA 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.

  Because PowerScript syntax does not apply to InfoMaker applications, the Preview tab is *not available* in the Database Profile Setup dialog box in InfoMaker.

Setting DBParm parameters

In PowerBuilder, you can set DBParm parameters by doing either of the following on *all* supported platforms:

- 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. A database profile is created when you do either of the following:

- Define an ODBC data source in PowerBuilder on the Windows and Macintosh platforms by completing the ODBC setup dialog box for your data source driver. (For instructions, see Chapter 2, "Using ODBC Data Sources and Drivers".)
- Define a Powersoft database interface in PowerBuilder on all platforms by completing the Database Profile Setup dialog box for your connection. (For instructions, see Chapter 3, "Using Powersoft Database Interfaces".)

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

❖ **To set DBParm parameters in a database profile:**

1 Click the Database Profile button in the PowerBar.
or
In the Database painter, select File>Connect>Setup from the menu bar.

Database Profile button

If your PowerBar does not include the Database Profile button, use the customize feature to add the button to the PowerBar.

FOR INFO For instructions on customizing toolbars, see the *PowerBuilder User's Guide*.

The Database Profiles dialog box displays, listing your installed Powersoft database interfaces and existing database profiles.

2 Click the plus sign (+) to the left of the interface you are using.
or
If your interface name is preceded by a plus sign, double-click the name.

The list expands to display the database profiles defined for your interface.

3 Select the name of the profile you want to edit and click Edit.
or
Display the popup menu for a database profile and select Edit.

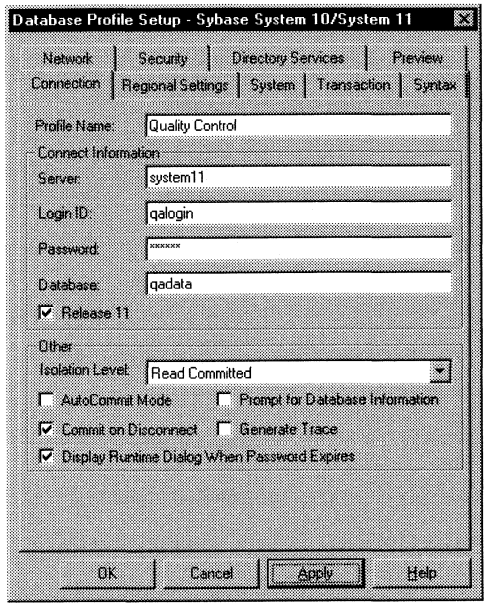
The Database Profile Setup dialog box for the selected profile displays.

4 On one or more tab pages in the Database Profile Setup dialog box, supply values for any DBParm parameters you want to set.

For example, in addition to values for basic connection parameters (Server, Login ID, Password, and Database), the Connection tab for the following Sybase System 11.x profile named Quality Control shows nondefault settings for the following DBParm parameters:

- Release 11 (corresponds to the Release DBParm parameter)

- Display Runtime Dialog When Password Expires (corresponds to the PWEncrypt DBParm parameter)



FOR INFO For information about DBParm parameters for your interface and the values you should supply, click Help.

5 Click Apply to save your settings without closing the Database Profile Setup dialog box.

Clicking Apply is useful if you want to set connection parameters on more than one tab page.

6 (Optional) In PowerBuilder, click the Preview tab if you want to see the PowerScript connection syntax generated for each option.

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 INFO For instructions, see "Copying DBParm syntax from the Preview tab" on page 335.

7 Click OK to close the Database Profile Setup dialog box.

PowerBuilder saves your DBParm settings in the database profile entry in the PowerBuilder initialization file.

For example, here is the database profile entry for Quality Control. The DBParm settings are shown in bold.

```
[Profile Quality Control]
DBMS=SYC Sybase System 10/11
Database=qadata
UserId=
DatabasePassword=
LogPassword=qapass
ServerName=system11
LogId=qalogin
Lock=
DbParm=Release='11',PWDialog=1
Prompt=0
AutoCommit=0
```

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.

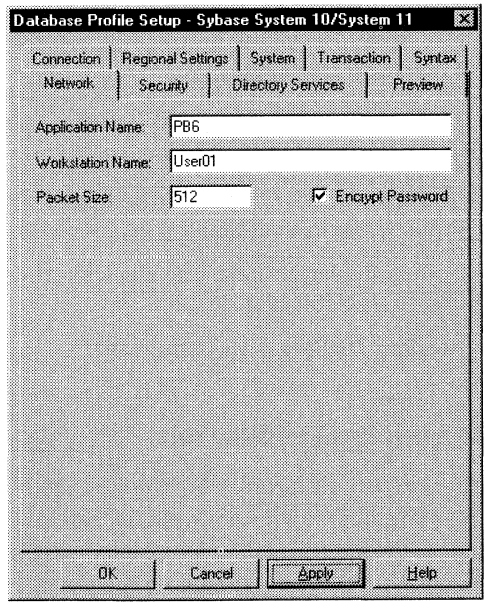
As you set DBParm parameters in the Database Profile Setup dialog box in the development environment, PowerBuilder generates the correct connection syntax on the Preview tab. Therefore, copying the syntax directly from the Preview tab ensures that you use the correct PowerScript DBParm syntax in your 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 INFO For instructions, see "Setting DBParm parameters in the development environment" on page 332.

For example, the Network tab in the Database Profile Setup - Sybase System 10/System 11 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).

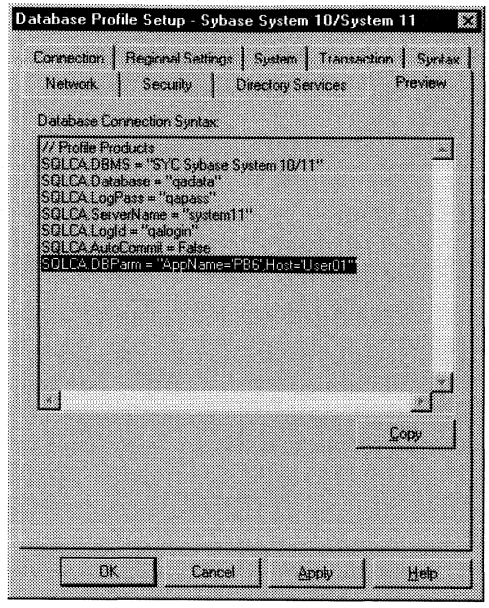


FOR INFO For information about the DBParm parameters for your interface and the values you should supply, click Help.

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

2 Use the following PowerScript syntax to specify the ProfileString function with the SQLCA.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","")
```

3 Compile the script to save your changes.

FOR INFO For instructions, see the *PowerBuilder User's Guide*.

Setting database preferences

How to set

The way you set connection-related database preferences in PowerBuilder depends on the preference, as summarized in the following table: (AutoCommit and Lock are the only database preferences that you can set in a PowerBuilder application script.)

Database preference	Set in development environment by editing	Set in PowerBuilder application by editing
AutoCommit	Database Profile Setup dialog box	Application script
Keep Connection Open	Database Preferences property sheet	—
Lock	Database Profile Setup dialog box	Application script
Read Only	Database Preferences property sheet	—
Shared Database Profiles	Database Preferences property sheet	—
SQL Terminator Character	Database Preferences property sheet	—
Use Powersoft Repository	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 specific information about using each database preference, see Chapter 8, "Database Preferences".

Setting database preferences in the development environment

There are two ways to set database preferences in the PowerBuilder development environment on *all* supported 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 Click the Database Profile button in the PowerBar.
or
In the Database painter, select File>Connect>Setup from the menu bar.

Database Profile button
If your PowerBar does not include the Database Profile button, use the customize feature to add the button to the PowerBar.

FOR INFO For instructions on customizing toolbars, see the *PowerBuilder User's Guide*.

The Database Profiles dialog box displays, listing your installed Powersoft database interfaces and existing database profiles.

2 Click the plus sign (+) to the left of the interface you are using.
or
If your interface name is preceded by a plus sign, double-click the name.

The list expands to display the database profiles defined for your interface.

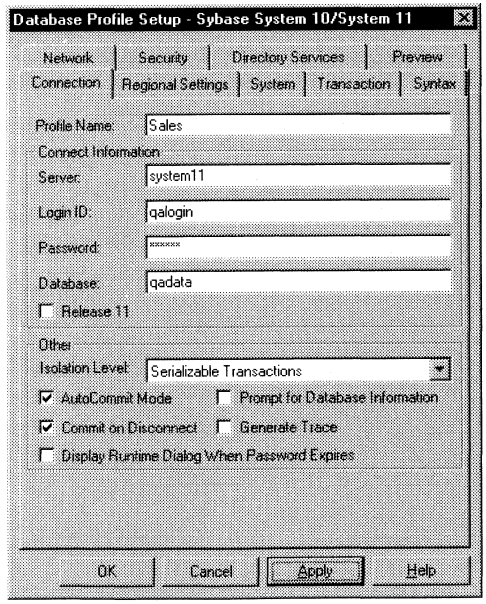
3 Select the name of the profile you want to edit and click Edit.
or
Display the popup menu for a database profile and select Edit.

The Database Profile Setup dialog box for the selected profile displays.

4 On the Connection tab, 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 dropdown listbox. (The Isolation Level dropdown listbox 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 checkbox to set AutoCommit to True or clear the AutoCommit Mode checkbox (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 for the following Sybase System 11.x profile named Sales shows nondefault settings for Isolation Level and AutoCommit Mode.



5 (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 INFO For instructions, see "Copying DBParm syntax from the Preview tab" on page 335.

6 Click OK to close the Database Profile Setup dialog box.

PowerBuilder saves your settings in the database profile entry in the PowerBuilder initialization file.

For example, here is the database profile entry for Sales. The Lock and AutoCommit settings are shown in bold.

```
[Profile Sales]
DBMS=SYC Sybase System 10/11
Database=qadata
UserId=
DatabasePassword=
LogPassword=qapass
ServerName=system11
LogId=qalogin
Lock=3
DbParm=
Prompt=0
AutoCommit=1
```

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
- Use Powersoft Repository
- Read Only
- Keep Connection Open
- 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 INFO For information about the other preferences you can set in the Database Preferences property sheet, see the *PowerBuilder User's Guide*.

❖ **To set connection-related preferences in the Database Preferences property sheet:**

1 Click the Database button in the PowerBar.
or
Select File>PowerPanel from the menu bar and then select Database Painter in the PowerPanel.

The Database painter opens.

2 Click the Database Preferences button.
or
Select Design>Options from the menu bar.

Database Preferences button

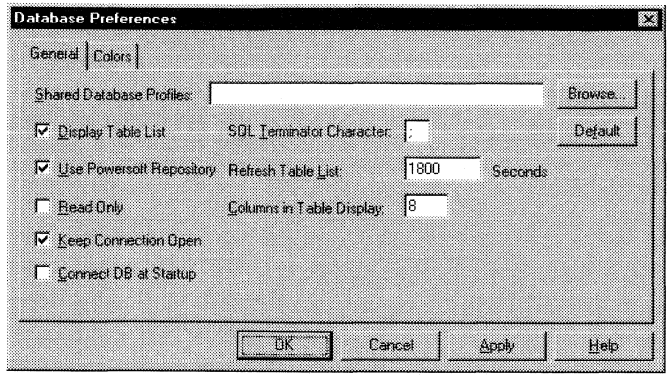
If your PainterBar does not include the Database Preferences button, use the customize feature to add the button to the PainterBar.

FOR INFO For instructions on customizing toolbars, see the *PowerBuilder User's Guide*.

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

Connect DB at Startup in InfoMaker only

The Connect DB at Startup preference is available only in the Database Preferences dialog box in InfoMaker.



3 Specify values for one or more of the following connection-related database preferences:

Preference	Description	For details, see
Shared Database Profiles	Specifies the pathname of the PowerBuilder initialization 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 321
Use Powersoft Repository	Specifies whether PowerBuilder should create and use the Powersoft repository tables. Select or clear the Use Powersoft Repository checkbox as follows: ♦ **Select the checkbox** (Default) Creates and uses the Powersoft repository tables ♦ **Clear the checkbox** Does *not* create the Powersoft repository tables	Use Powersoft Repository on page 578

Preference	Description	For details, see
Read Only	Specifies whether PowerBuilder should update the repository tables and any other tables in your database. Select or clear the Read Only checkbox as follows: ♦ **Select the checkbox** Does not update the repository tables or any other tables in your database. You *cannot* modify (update) information in the repository tables or any other database tables from the DataWindow and Report painters when the Read Only checkbox is selected ♦ **Clear the checkbox** (Default) Updates the repository tables and any other tables in your database	Read Only on page 575

Preference	Description	For details, see
Keep Connection Open	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 checkbox as follows: ♦ **Select the checkbox** (Default) Stays connected to the database throughout your session and closes the connection when you exit ♦ **Clear the checkbox** Opens the connection only when a painter requests it and closes the connection when you close a painter or finish compiling a script **PowerBuilder only** Keep Connection Open has no effect in InfoMaker or when you use a database profile in PowerBuilder to connect.	Keep Connection Open on page 569
SQL Terminator Character	Specifies the SQL statement terminator character used by the Database Administration painter in PowerBuilder The default terminator character is a semicolon (;). If you are creating stored procedures and triggers in the Database Administration 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	SQL Terminator Character on page 576

4 Do one of the following:

- Click Apply to apply the preference settings to the current connection without closing the Database Preferences property sheet.

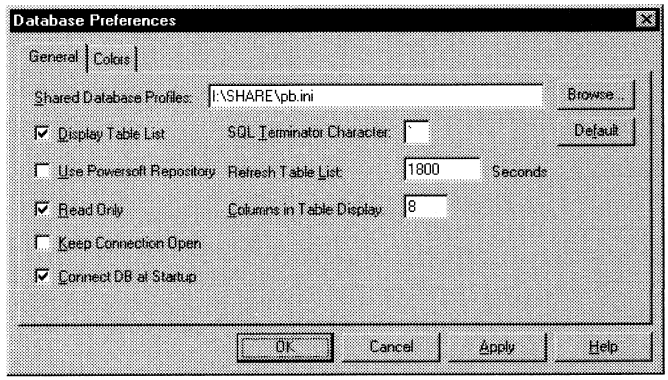
- Click OK to apply the preference settings to the current connection and close the Database Preferences property sheet.

PowerBuilder saves your preference settings in the [Database] section of the PowerBuilder initialization file.

FOR INFO For information about the name and location of the PowerBuilder initialization file on different platforms, see "About database connections" on page 287.

Example

This example for the Windows platform shows the Database Preferences property sheet with nondefault settings for Shared Database Profiles, Use Powersoft Repository, Read Only, Keep Connection Open, Connect DB at Startup (InfoMaker only), and SQL Terminator Character.



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 INFO 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.

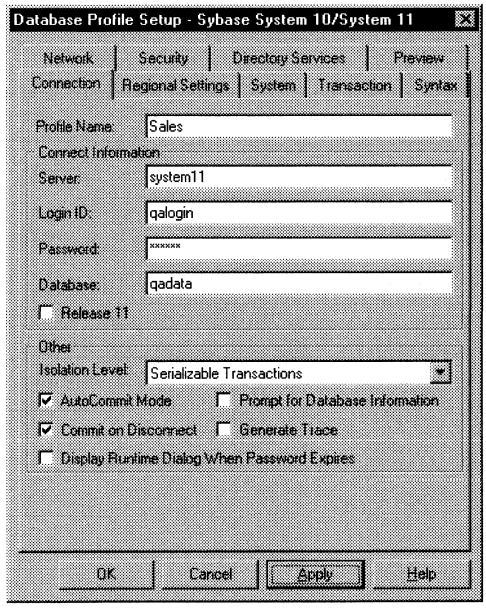
As you complete the Database Profile Setup dialog box in the development environment, PowerBuilder generates the correct connection syntax on the Preview tab for each selected option. Therefore, copying the syntax directly from the Preview tab ensures that you use the correct PowerScript syntax in your 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 INFO For instructions, see "Setting AutoCommit and Lock in the database profile" on page 341.

For example, in addition to values for basic connection parameters (Server, Login ID, Password, and Database), the Connection tab for the following Sybase System 11.x profile named Sales shows nondefault settings for Isolation Level and AutoCommit Mode.

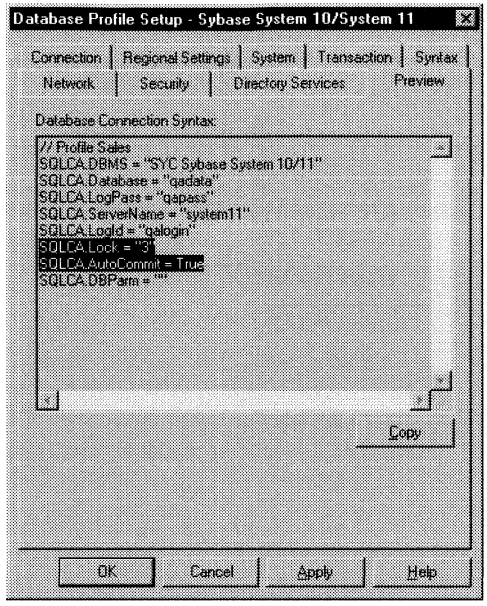


FOR INFO For information about the DBParm parameters for your interface and the values you should supply, click Help.

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 INFO For instructions, see the *PowerBuilder 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 SQL Server System 11.x 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="xxyyzz"
SQLCA.ServerName="SYSTEM11"
SQLCA.AutoCommit="True"
SQLCA.Lock= "3"
```

FOR INFO For more information, see AutoCommit on page 566 or Lock on page 570.

3 Compile the script to save your changes.

FOR INFO For instructions, see the *PowerBuilder 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 INFO For instructions, see the *PowerBuilder User's Guide*.

2 Use the following PowerScript syntax to specify the ProfileString function with the SQLCA.AutoCommit and SQLCA.Lock properties:

SQLCA.AutoCommit = **ProfileString** (*file, section, variable, default_value*)

SQLCA.Lock = **ProfileString** (*file, section, variable, default_value*)

For example, the following statements in a PowerBuilder script read the AutoCommit and Lock values from the [Database] section of the APP.INI file:

```
SQLCA.AutoCommit=ProfileString("APP.INI",
   "Database","AutoCommit","")
SQLCA.Lock=ProfileString("APP.INI","Database",
   "Lock","")
```

3 Compile the script to save your changes.

FOR INFO For instructions, see the *PowerBuilder User's Guide*.

CHAPTER 6

Troubleshooting Your Connection

Where you are

(*Optional*) Get an introduction to database connections
Prepare to use the data source or database
Install the ODBC driver or Powersoft database interface
Define the data source or database
Connect to the data source or database
(*Optional*) Set additional connection parameters
> (*Optional*) Troubleshoot the data 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

Contents

Overview of troubleshooting tools

Tools

When you use PowerBuilder, there are two tools available to trace your database connection in order to troubleshoot problems:

Use this tool	To trace a connection to
Database Trace	*Any* database that PowerBuilder accesses through the Powersoft ODBC interface or one of the Powersoft native database interfaces
ODBC Driver Manager Trace	An ODBC data source only

Availability

Database Trace The Database Trace tool is available on all supported PowerBuilder platforms.

ODBC Driver Manager Trace The ODBC Driver Manager Trace tool is available on all PowerBuilder platforms that support ODBC connectivity. (For information on ODBC connectivity in PowerBuilder, see "Using the Powersoft ODBC interface" on page 19.)

Platform differences

The way you use the Database Trace and ODBC Driver Manager Trace tools is the same on all supported PowerBuilder platforms. In fact, the only difference you'll notice across platforms is the location of the default log file for each tool.

FOR INFO For information about the Database Trace log on different platforms, see "Location of the Database Trace log on different platforms" on page 358.

FOR INFO For information about the ODBC Driver Manager Trace log on different platforms, see "About ODBC Driver Manager Trace" on page 374.

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 only use the Database Trace tool for 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 a SQL Server database, you can trace either the ODBC connection or the SQL Server 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 on different platforms

PBTRACE.LOG

By default, PowerBuilder writes output of the Database Trace tool to a file named PBTRACE.LOG on all supported platforms. The location of PBTRACE.LOG depends on the platform you are using:

Platform	Location
Windows	PBTRACE.LOG in your Windows directory
Macintosh	System Folder:Preferences:pbtrace.log
UNIX	$HOME/pbtrace.log

Nondefault log file

If you prefer, you can specify a nondefault name and location for the log file when you use Database Trace.

FOR INFO For instructions, see "Specifying a nondefault Database Trace log" on page 368.

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 Powersoft database interface

Format of the Database Trace log

The specific contents 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 for *all* supported platforms and databases to display output:

COMMAND: (*time*)

{*additional_information*}

Parameter	Description
COMMAND	The internal command that PowerBuilder executes to perform the database operation
time	The number of milliseconds it takes PowerBuilder to perform the database operation. The precision used depends on your operating system's timing mechanism For example, on Windows, the Database Trace timer rounds down to the nearest 55-millisecond increment. If an operation takes 54 milliseconds or less to perform, the log displays the time as 0 milliseconds. If an operation takes between 55 and 109 milliseconds, the time displays as 55 milliseconds
additional_information	(Optional) Additional information about the command. The information provided depends on the database operation

Example

For example, the following portion of the log file on Windows shows the commands PowerBuilder executes to fetch two rows from a database table:

```
FETCH NEXT: (55 MilliSeconds)
   dept_id=dept_name=Business Services
FETCH NEXT: (0 MilliSeconds)
   dept_id=dept_name=Corporate Management
```

FOR INFO For a more complete example of Database Trace output, see "Sample Database Trace output" on page 372.

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.

To start Database Trace in	Do this
The PowerBuilder development environment	Edit your database profile
A PowerBuilder application	Edit your application script

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. PowerBuilder automatically creates a database profile when you do either of the following:

- **On Windows and Macintosh** Define an ODBC data source by completing the ODBC setup dialog box for your driver. (For instructions, see Chapter 2, "Using ODBC Data Sources and Drivers".)
- **On all platforms** Define a Powersoft database interface by completing the Database Profile Setup dialog box. (For instructions, see Chapter 3, "Using Powersoft Database Interfaces".)

❖ **To start the Database Trace tool by editing a database profile:**

1 Click the Database Profile button in the PowerBar.
or
In the Database painter, select File>Connect>Setup from the menu bar.

Database Profile button
If your PowerBar does not include the Database Profile button, use the customize feature to add the button to the PowerBar.

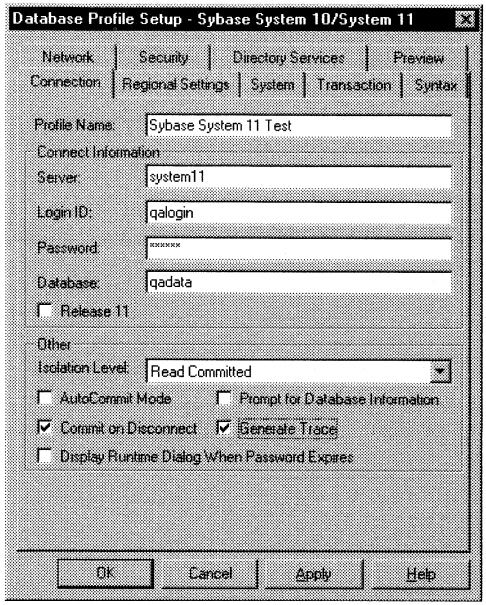
FOR INFO For instructions on customizing toolbars, see the *PowerBuilder User's Guide*.

The Database Profiles dialog box displays listing the names of existing profiles. If you are currently connected to a database profile, its name is highlighted.

2 Select the name of the profile for the database connection you want to trace and click Edit.
or
Display the popup menu for the profile you want to trace and select Edit.

The Database Profile Setup dialog box for the selected profile displays.

3 On the Connection tab, select the Generate Trace checkbox and click OK or Apply.



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 Sybase System 11 Test. The setting that starts Database Trace is shown in bold.

```
[Profile Sybase System 11 Test]
DBMS=TRACE SYC
Database=qadata
UserId=
DatabasePassword=
LogPassword=qapass
ServerName=system11
LogId=qalogin
...
```

4 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 INFO For instructions on specifying your own name and location for the Database Trace log file, see "Specifying a nondefault Database Trace log" on page 368.

5 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 INFO 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:**

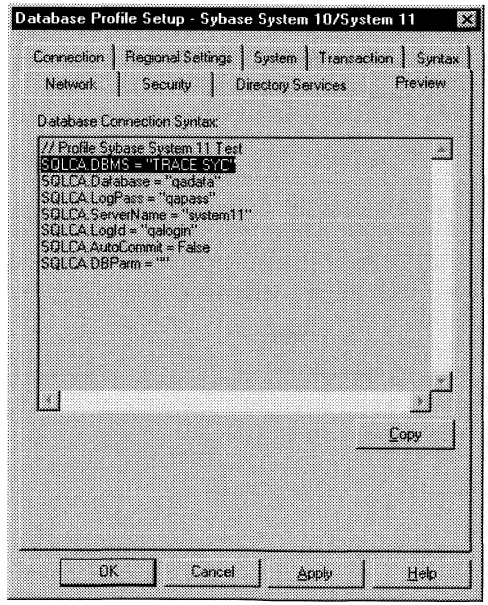
1 On the Connection tab in the Database Profile Setup dialog box for your connection, select the Generate Trace checkbox to turn on Database Trace.

FOR INFO For instructions, see "Starting Database Trace in the development environment" on page 360.

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 a SQL Server 4.x database named Test. The keyword *trace* in the DBMS property indicates that you want to trace the database connection.

```
SQLCA.DBMS        = "trace Sybase"
SQLCA.database    = "Test"
SQLCA.logId       = "Frans"
SQLCA.LogPass     = "xxyyzz"
SQLCA.ServerName  = "Tomlin"
```

Reading the DBMS value from an external text file

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.

The following procedure assumes that the DBMS value read from your initialization file 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","")
```

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 checkbox 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. Click Connect 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

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 Sybase"
```

Here is how the same DBMS connection property should look after you edit it to stop tracing:

```
SQLCA.DBMS = "Sybase"
```

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 profile 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, System Folder:Preferences folder (on Macintosh), or $HOME directory (on UNIX).

You can override this default in the development environment by editing your PowerBuilder initialization file. The name and location of the initialization file depends on your platform:

Platform	Location
Windows	PB.INI in PowerBuilder product directory
Macintosh	System Folder:Preferences:Powersoft 6.0 Preferences:PowerBuilder Preferences
UNIX	$HOME/.pb.ini

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 INFO For instructions on starting Database Trace, see "Starting Database Trace in the development environment" on page 360.

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 INFO For information abut where to find PBTRACE.LOG on different platforms, see "Location of the Database Trace log on different platforms" on page 358.

FOR INFO For instructions about specifying your own Database Trace log to use in the development environment, see "Specifying a nondefault Database Trace log" on page 368.

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 *PowerBuilder 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 *PowerBuilder 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 a Sybase System 11.x 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.

Similar format and content on all platforms

The basic format and content of the Database Trace log is the same on *all* supported PowerBuilder platforms.

However, 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

```
LOGIN: (220 MilliSeconds)
CONNECT TO trace SYC Sybase System 10/11:
DATA=sample
LOGID=sa
SERVER=oregano (55 MilliSeconds)
BEGIN TRANSACTION: (0 MilliSeconds)
```

Prepare SELECT statement

```
PREPARE: (0 MilliSeconds)
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 (0 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):
(0 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: (0 MilliSeconds)
   emp_id=emp_lname=Jonesemp_fname=Alan
   dept_id=
FETCH NEXT: (0 MilliSeconds)
   emp_id=emp_lname=Cicconeemp_fname=Peter
   dept_id=
FETCH NEXT: (0 MilliSeconds)
   emp_id=emp_lname=Houstonemp_fname=Mary
   dept_id=
FETCH NEXT: (0 MilliSeconds)
   emp_id=emp_lname=Smithemp_fname=Susan
   dept_id=
FETCH NEXT: (0 MilliSeconds)
   emp_id=emp_lname=Steinemp_fname=David
   dept_id=
FETCH NEXT: (0 MilliSeconds)
   emp_id=emp_lname=Watsonemp_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: (0 MilliSeconds)
```

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 Powersoft ODBC interface.

Unlike the Database Trace tool, the ODBC Driver Manager Trace tool *cannot* trace connections through one of the Powersoft native database interfaces.

What this tool does

ODBC Driver Manager Trace records information about ODBC API calls (such as SQLDriverConnect, SQLGetInfo, and SQLFetch) made by PowerBuilder while connected to an ODBC data source. It writes this information to a default log file named SQL.LOG or to a log file that you specify.

What both tools do

The information from ODBC Driver Manager Trace, like Database Trace, can help you:

- Understand what PowerBuilder is doing *internally* while connected to an ODBC data source
- Identify and resolve problems with your ODBC connection
- Provide useful information to Technical Support if you call them for help with your database connection

When to use this tool

You should use ODBC Driver Manager Trace *instead* 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, you should use ODBC Driver Manager Trace for debugging purposes only and keep it turned off when you are not debugging.

Platform differences

The way you use ODBC Driver Manager Trace is the same on all PowerBuilder platforms that support ODBC. However, there are some platform-specific considerations you need to know when using it.

On Macintosh In PowerBuilder on Power Macintosh, you can use ODBC Driver Manager Trace to troubleshoot problems with your ODBC connection to Sybase SQL Anywhere and to any data sources that you access with ODBC-compliant drivers obtained from other vendors.

On UNIX In PowerBuilder on UNIX, you can use ODBC Driver Manager Trace to troubleshoot problems with your connection to a data source through one of the INTERSOLV ODBC drivers provided. *These are the only ODBC connection supported in PowerBuilder on UNIX at this time.*

SQL.LOG file On all platforms, PowerBuilder writes ODBC Driver Manager Trace output to a default log file named SQL.LOG or to a log file that you specify. The default location of SQL.LOG depends on the platform you are using.

Platform	Location
Windows	Your root directory
Macintosh	The folder where your current application resides
UNIX	Your $HOME directory

Starting ODBC Driver Manager Trace

By default, ODBC Driver Manager Trace is turned off in PowerBuilder. You can start it in the PowerBuilder development environment or in a PowerBuilder application to trace your ODBC connection.

To start ODBC Driver Manager Trace in	Do this
The PowerBuilder development environment	Edit your database profile
A PowerBuilder application	Edit your application script

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.

On Windows and Macintosh, PowerBuilder automatically creates a database profile when you define an ODBC data source by completing the ODBC setup dialog box for your driver. (For instructions, see Chapter 2, "Using ODBC Data Sources and Drivers".)

❖ **To start ODBC Driver Manager Trace by editing the database profile:**

1 Click the Database Profile button in the PowerBar.
or
In the Database painter, select File>Connect>Setup from the menu bar.

Database Profile button
If your PowerBar does not include the Database Profile button, use the customize feature to add the button to the PowerBar.

FOR INFO For instructions on customizing toolbars, see the *PowerBuilder User's Guide*.

The Database Profiles dialog box displays, listing the names of existing profiles. If you are currently connected to a database profile, its name is highlighted.

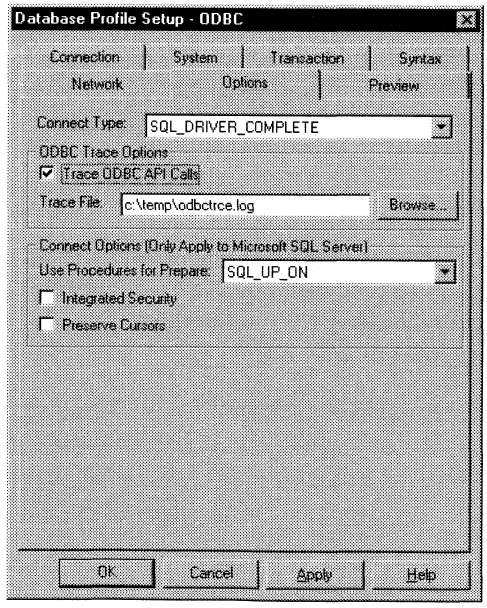
2 Select the name of the ODBC profile for the connection you want to trace and click the Edit button.
or
Display the popup menu for the ODBC profile you want to trace and select Edit.

The Database Profile Setup-ODBC dialog box for the selected profile displays.

3 On the Options tab, select the Trace ODBC API Calls checkbox.

4 (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 checkbox is selected and no trace file is specified, PowerBuilder sends ODBC Driver Manager Trace output to the default SQL.LOG file.



5 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 PowerBuilder initialization file.

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 ConnectString DBParm parameter) are shown in bold.

```
[PROFILE Employee]
DBMS=ODBC
...
DbParm=Connectstring='DSN=Employee',
   ConnectOption='SQL_OPT_TRACE,SQL_OPT_TRACE_ON;
   SQL_OPT_TRACEFILE,c:\temp\odbctrce.log'
...
```

6 Click Connect in the Database Profiles dialog box to connect to the database.

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 INFO 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:

Parameter	Description
SQL_OPT_TRACE	**Purpose** Starts or stops ODBC Driver Manager Trace in PowerBuilder **Values** The values you can specify are: ♦ **SQL_OPT_TRACE_OFF** (Default) Stops ODBC Driver Manager Trace ♦ **SQL_OPT_TRACE_ON** Starts ODBC Driver Manager Trace
SQL_OPT_TRACEFILE	**Purpose** 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) **Values** You can specify any filename for the trace file, *following the naming conventions of your operating system.* 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 SQL.LOG FOR INFO For information about the location of SQL.LOG on different platforms, see "About ODBC Driver Manager Trace" on page 374.

Copying ConnectOption syntax from the Preview tab

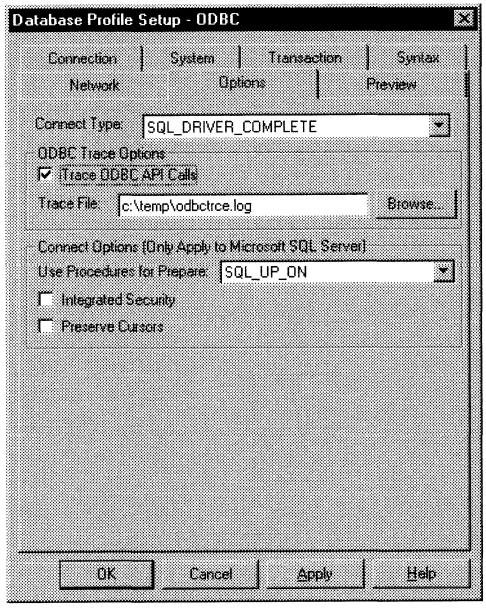
The easiest way to start ODBC Driver Manager Trace in a PowerBuilder application script is to copy the PowerScript ConnectString DBParm syntax from the Preview tab in the Database Profile Setup - ODBC dialog box into your script, modifying the default transaction object name (SQLCA) if necessary.

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 checkbox and (optionally) specify a log file in the Trace File box to start ODBC Driver Manager Trace.

FOR INFO For instructions, see "Starting ODBC Driver Manager Trace in the development environment" on page 376.



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.

In the following example, the PowerScript syntax that starts ODBC Driver Manager Trace and sends output to the file C:\TEMP\ODBCTRCE.LOG is shown in bold.

```
// 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 specify a nondefault trace file, respectively.

For example, the portion of the following statement shown in bold starts ODBC Driver Manager Trace in your application and sends output to a file named MYTRACE.LOG. Insert a comma to separate the ConnectString 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=
   "ConnectString='DSN=Test;UID=PB;PWD=xyz',
   ConnectOption='SQL_OPT_TRACE,SQL_OPT_TRACE_ON;
   SQL_OPT_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 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.

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 INFO For instructions, see "Starting ODBC Driver Manager Trace in the development environment" on page 376.

2 On the Options tab, clear the Trace ODBC API Calls checkbox.

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.

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=
   "ConnectString='DSN=Test;UID=PB;PWD=xyz',
   ConnectOption='SQL_OPT_TRACE,SQL_OPT_TRACE_ON;
   SQL_OPT_TRACEFILE,C:\MYTRACE.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=
   "ConnectString='DSN=Test;UID=PB;PWD=xyz',
   ConnectOption='SQL_OPT_TRACE,SQL_OPT_TRACE_OFF;
   SQL_OPT_TRACEFILE,C:\MYTRACE.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 INFO For information about where to find the default SQL.LOG file on different platforms, see "About ODBC Driver Manager Trace" on page 374.

❖ **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 *PowerBuilder 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 traces an ODBC connection to the Powersoft Demo Database.

FOR INFO For more about a particular ODBC API call, see your ODBC documentation.

Similar format and content on all platforms
The basic format and content of the ODBC Driver Manager Trace log file is the same on *all* PowerBuilder platforms that support it.

```
...
PB60 fffe0f03:fffc5d47EXIT SQLDataSourcesW  with
return code 0 (SQL_SUCCESS)
    HENV 0x01491e30
    UWORD 2 <SQL_FETCH_FIRST>
    WCHAR * 0x01491890 [ 40] "Powersoft Demo DB V6"
    SWORD 66
    SWORD * 0x0136e320 (40)
    WCHAR * 0x0138107c [ 46] "Sybase SQL Anywhere 5.0"
    SWORD 1026
    SWORD * 0x0136e33c (46)
...
PB60 fffe0f03:fffc5d47ENTER SQLGetConnectOption
    HDBC 0x01491410
    UWORD 104
    PTR 0x0136d424
...
PB60 fffe0f03:fffc5d47ENTER SQLSetConnectOption
    HDBC 0x01491410
    UWORD 104
    UDWORD 0
...
PB60 fffe0f03:fffc5d47EXIT  SQLGetInfoW  with
return code 0 (SQL_SUCCESS)
    HDBC 0x01491410
    UWORD 10 <SQL_ODBC_VER>
    PTR 0x0138107c [ 20] "03.00.0000"
    SWORD 512
    SWORD * 0x0136d428 (20)
...
```

```
PB60 fffe0f03:fffc5d47ENTER SQLBindCol
   HSTMT 0x0138107c
   UWORD 5
   SWORD 1 <SQL_C_CHAR>
   PTR 0x02ce08a2
   SDWORD 256
   SDWORD * 0x0136d408
...
PB60 fffe0f03:fffc5d47ENTER SQLFetch
   HSTMT 0x0138107c
PB60 fffe0f03:fffc5d47EXIT  SQLFetch  with return
code 0 (SQL_SUCCESS)
   HSTMT 0x0138107c
...
PB60 fffe0f03:fffc5d47ENTER SQLPrimaryKeysW
   HSTMT 0x0138107c
   WCHAR * 0x00000000
   WCHAR * 0x01491ba0 [ 6] "dba"
   SWORD 6
   WCHAR * 0x01491bb0 [ 16] "customer"
   SWORD 16
...
PB60 fffe0f03:fffc5d47ENTER SQLExtendedFetch
   HSTMT 0x0138107c
   UWORD 1 <SQL_FETCH_NEXT>
   SDWORD 1
   UDWORD * 0x0136cd20
   UWORD * 0x02d10020
PB60 fffe0f03:fffc5d47EXIT SQLExtendedFetch with
return code 0 (SQL_SUCCESS)
   HSTMT 0x0138107c
   UWORD 1 <SQL_FETCH_NEXT>
   SDWORD 1
   UDWORD * 0x0136cd20 (126)
   UWORD * 0x02d10020 (0)
...
PB60 fffe0f03:fffc5d47ENTER SQLFreeStmt
   HSTMT 0x0138107c
   UWORD 1 <SQL_DROP>
PB60 fffe0f03:fffc5d47EXIT  SQLFreeStmt  with
return code 0 (SQL_SUCCESS)
   HSTMT 0x0138107c
   UWORD1 <SQL_DROP>
```

PART 2

Connection Parameter Reference

This part describes the syntax and use of each DBParm parameter and database preference that you can set in a connection to take advantage of DBMS-specific features.

CHAPTER 7

DBParm Parameters

About this chapter

This chapter describes the syntax and use of each DBParm parameter that you can set in PowerBuilder.

Contents

The DBParm parameters are described in alphabetical order.

DBParm parameters and supported database interfaces

The following table lists each supported Powersoft database interface and the DBParm parameters you can use with that interface in PowerBuilder.

Powersoft database interface	DBParm parameters
IBM	CommitOnDisconnect DBAdm DelimitIdentifier FormatArgsAsExp GroupID LoginAttempts NumericFormat PBCatalogOwner SystemOwner TableCriteria (IBM, InformationConnect Gateway)
IN5 INFORMIX (through INFORMIX-NET 5.x)	CommitOnDisconnect DisableBind INET_DBPATH INET_PROTOCOL INET_SERVICE Scroll
IN7 INFORMIX (through INFORMIX ESQL 7.2)	CommitOnDisconnect DisableBind INET_DBPATH INET_PROTOCOL INET_SERVICE Scroll
MSS Microsoft SQL Server 6.x	AppName Async CommitOnDisconnect CursorLock (SQL Server) CursorScroll (SQL Server) DateTimeAllowed DBGetTime DBTextLimit Host Language Log PacketSize (SQL Server) PBCatalogOwner Secure StaticBind SystemProcs

Powersoft database interface	DBParm parameters
ODBC **Using DBParms with ODBC** These DBParm parameters are supported by the Powersoft ODBC interface only if *both* the ODBC driver you are using and the backend DBMS support the feature.	Async Block (ODBC, Oracle) CommitOnDisconnect ConnectOption ConnectString CursorLib CursorLock (ODBC) CursorScroll (ODBC) Date DateTime DBGetTime DecimalSeparator DelimitIdentifier DisableBind FormatArgsAsExp IdentifierQuoteChar InsertBlock LoginTimeOut MsgTerse NumericFormat PacketSize (ODBC) PBCatalogOwner PBUseProcOwner SQLCache StaticBind TableCriteria (ODBC) Time
OR7 Oracle Version 7	Block (ODBC, Oracle) CommitOnDisconnect Date DateTime DecimalSeparator DelimitIdentifier DisableBind FormatArgsAsExp MixedCase PBCatalogOwner PBDBMS SQLCache StaticBind TableCriteria (Oracle) Time

Powersoft database interface	DBParm parameters
O71 Oracle Version 7.1	Block (ODBC, Oracle) CommitOnDisconnect Date DateTime DecimalSeparator DelimitIdentifier DisableBind FormatArgsAsExp MixedCase PBCatalogOwner PBDBMS SQLCache StaticBind TableCriteria (Oracle) Time
O72 Oracle Version 7.2	Async Block (ODBC, Oracle) CommitOnDisconnect Date DateTime DBGetTime DecimalSeparator DelimitIdentifier DisableBind FormatArgsAsExp MixedCase PBCatalogOwner PBDBMS SQLCache StaticBind TableCriteria (Oracle) Time

Powersoft database interface	DBParm parameters
O73 Oracle Version 7.3	Async Block (ODBC, Oracle) CommitOnDisconnect Date DateTime DBGetTime DecimalSeparator DelimitIdentifier DisableBind FormatArgsAsExp MixedCase PBCatalogOwner PBDBMS SQLCache StaticBind TableCriteria (Oracle) ThreadSafe Time
SYB SQL Server 4.x DB-Lib *and* SYT Sybase SQL Server DB-Lib	AppName Async CharSet (SYT only) CommitOnDisconnect CursorLock (SQL Server) CursorScroll (SQL Server) DateTimeAllowed DBGetTime (SYB only) DBTextLimit Host Log PacketSize (SQL Server) Release (SQL Server 4.x) StaticBind
MDI Sybase InformationConnect DB2 Gateway interface (formerly called MDI Database Gateway Interface for DB2)	AppName CommitOnDisconnect ConversionTable DelimitIdentifier FormatArgsAsExp GroupID Host PBCatalogOwner Request StaticBind SystemOwner TableCriteria (IBM, InformationConnect Gateway)

Powersoft database interface	DBParm parameters
NET Sybase Net-Gateway for DB2 interface	AppName CommitOnDisconnect DelimitIdentifier FormatArgsAsExp Host PBCatalogOwner SQLQualifiers StaticBind SystemOwner TableCriteria (Sybase Net-Gateway, Sybase System 10 and System 11)

Powersoft database interface	DBParm parameters
SYC Sybase Systems 10.x and 11.x *and* SYD Sybase Systems 10.x and 11.x distributed application interface on UNIX	AppName Async Block (Sybase System 10 and System 11) CharSet CommitOnDisconnect CursorUpdate DateTimeAllowed DBGetTime DelimitIdentifier DS_Alias DS_Copy DS_DitBase DS_Failover DS_Principal DS_Provider DS_TimeLimit Host Language Locale Log ModifySyntax (SYC only) PacketSize (SQL Server) PBCatalogOwner PWDialog PWEncrypt Release (Sybase System 10 and System 11) Sec_Channel_Bind Sec_Confidential Sec_Cred_Timeout Sec_Data_Integrity Sec_Data_Origin Sec_Delegation Sec_Keytab_File Sec_Mechanism Sec_Mutual_Auth Sec_Network_Auth Sec_Replay_Detection Sec_Seq_Detection Sec_Server_Principal Sec_Sess_Timeout StaticBind SystemProcs TableCriteria (Sybase Net-Gateway, Sybase System 10 and System 11)

AppName

Description

If the DBMS supports it, specifies the application name you want to use when connecting to the database in PowerBuilder.

When to specify AppName
You must specify the AppName DBParm parameter *before* connecting to the database in PowerBuilder.

Applies to

MSS Microsoft SQL Server 6.x
SYB and SYT SQL Server 4.x
MDI Sybase InformationConnect DB2 Gateway interface
NET Sybase Net-Gateway for DB2 interface
SYC and SYD Sybase Systems 10.x and 11.x

Syntax

AppName = '*application_name*'

Default value

All DBMSs except Sybase System 10.x and System 11.x There is no default value for AppName. PowerBuilder does not set the AppName parameter unless you specify a value.

Sybase System 10.x and System 11.x PowerBuilder sets the CS_APPNAME connection property to PowerBuilder, as follows:

```
AppName = 'PowerBuilder'
```

Usage

SQL Server databases It is useful to specify a different AppName value for each of your SQL Server applications. If you are a SQL Server administrator, you can query the MASTER.DBO.SYSPROCESSES table to determine which applications are running on the database server. The value specified for AppName displays in the program_name column of the MASTER.DBO.SYSPROCESSES table, making it easy to identify the applications.

Examples

Example 1 To set the application name to Test:

- **Database profile** Type the following in the Application Name box on the Network tab in the Database Profile Setup dialog box:

```
Test
```

- **PowerBuilder application script** Type the following in a PowerBuilder application script:

```
SQLCA.dbParm = "AppName = 'Test'"
```

Example 2 You can set the AppName and Host parameters in a single DBParm statement to specify both the application name and the host name. To set the application name to Sales and the host name to Fran:

- **Database profile** Type Sales in the Application Name box and Fran in the Workstation Name box on the Network tab in the Database Profile Setup dialog box.
- **PowerBuilder application script** Type the following in a PowerBuilder application script:

```
SQLCA.dbParm = "AppName = 'Sales',Host = 'Fran'"
```

Using the examples in a PowerBuilder application script If you specify DBParm parameters in your database profile, the PowerScript 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.

See also

Host

Async

Description

Allows you to perform asynchronous operations on your database in PowerBuilder. Essentially, this means that if you have coded a RetrieveRow event for a DataWindow object or report, you can cancel the current database retrieval operation or start another nondatabase operation that does not use the same database connection before the current operation completes. You can also switch to another Windows process while the retrieval takes place.

By default, PowerBuilder operates synchronously.

Applies to

MSS Microsoft SQL Server 6.x
ODBC (if driver and backend DBMS support this feature)
O72 Oracle Version 7.2
O73 Oracle Version 7.3
SYB and SYT SQL Server 4.x
SYC and SYD Sybase Systems 10.x and 11.x

Syntax

Async = *value*

Parameter	Description
value	A value specifying synchronous or asynchronous operation. Values are: ♦ **0** (Default) Synchronous operation ♦ **1** Asynchronous operation

Default value

Async = 0

Usage

Enabling asynchronous operation in PowerBuilder is useful when you are executing a complex SQL statement that takes several minutes to return results. If the Async parameter is set to 1, you can do either of the following while the SQL statement is executing:

- Work in another window
- Cancel the statement before it retrieves the first row of data

When to set Async If you are communicating with the database in a PowerBuilder script, you can reset the Async value at any time before or after the transaction object has connected to the database.

How data is retrieved When you retrieve data in a DataWindow object or report, the following steps occur in this order:

1 The database server compiles and executes the SQL statement.

2 PowerBuilder retrieves (fetches) the first row of data.

3 PowerBuilder retrieves each subsequent row of data.

What happens before the first row is retrieved While the server is compiling and executing the SQL statement and before PowerBuilder retrieves the first row of data, you must have done *both* of the following to enable asynchronous operation (allowing you to cancel the current operation before it retrieves the first row of data):

- Coded a RetrieveRow event for the DataWindow object or report (the code can contain only a comment)
- Set the Async DBParm parameter to 1

What happens after the first row is retrieved After the first row of data is retrieved and in between subsequent row fetches, you must have done only the following to enable asynchronous operation:

- Coded a RetrieveRow event for the DataWindow object or report

After the first row is retrieved, PowerBuilder operates asynchronously *without your having to set the Async DBParm to 1*. So you can cancel the current operation anytime after it retrieves the first row of data. Therefore, the Async DBParm parameter has no effect in PowerBuilder after the first row of data is retrieved.

Examples

Example 1 To enable asynchronous operation:

- **Database profile** Select the Asynchronous checkbox on the Transaction tab in the Database Profile Setup dialog box.
- **PowerBuilder application script** Type the following in a PowerBuilder script:

```
SQLCA.dbParm = "Async = 1"
```

Example 2 You can set the Async and DBGetTime parameters in a single DBParm statement. DBGetTime specifies the number of seconds you want PowerBuilder to wait for a response from the DBMS when you retrieve rows in a DataWindow object or report. To enable asynchronous operation and set the DBGetTime parameter to 20 seconds:

- **Database profile** Select the Asynchronous checkbox and type 20 in the Number of Seconds to Wait box on the Transaction tab in the Database Profile Setup dialog box.
- **PowerBuilder application script** Type the following in a PowerBuilder application script:

```
SQLCA.dbParm = "Async = 1, DBGetTime = 20"
```

Using the examples in a PowerBuilder application script If you specify DBParm parameters in your database profile, the PowerScript 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.

See also

DBGetTime

Block (ODBC, Oracle)

Description

Specifies the cursor blocking factor when connecting to an ODBC data source or Oracle database. The blocking factor determines the number of rows that a DataWindow object can fetch from the database at one time.

Using the Block DBParm parameter can improve performance when accessing a database in PowerBuilder.

Applies to

ODBC (if driver and backend DBMS support this feature)
Oracle (all interfaces)

Syntax

Block = *blocking_factor*

Parameter	Description
blocking_factor	The number of rows you want the DataWindow object to fetch from the database at one time. The blocking factor can be a number from 1 to 1000, inclusive To turn off block fetching, set Block to 1

Default value

The default value for the Block parameter depends on the DBMS you are accessing, as summarized in the following table:

DBMS	Block default value

ODBC	For most DataWindow objects, the Block default value is the following, *up to a maximum of 32K per column*:
	`Block = 1000`
	If you specified that the DataWindow object should retrieve only as many rows as needed from the database (Retrieve.AsNeeded property), the Block default value is the following, *up to a maximum of 32K per column*:
	`Block = 100`

Oracle	PowerBuilder sets the blocking factor to the largest value possible to fetch the maximum number of rows at one time, *up to a maximum of 32 K per column*

Using the default blocking factor
You should not have to set a nondefault value for Block. In most cases, the default blocking factor used by PowerBuilder should meet your needs when connecting to an ODBC data source or Oracle database.

Usage

Requirements for ODBC data sources To use the Block DBParm parameter with an ODBC data source, your ODBC driver must:

- Be ODBC Version 2.0-compliant or higher, *and*
- Support the SQLExtendedFetch API call

The SQL Anywhere ODBC driver that comes with PowerBuilder meets both of these requirements.

FOR INFO For information about whether your ODBC driver meets these requirements, see the documentation that comes with your driver.

Determining the Block value for ODBC data sources PowerBuilder searches the following in this order to determine the Block value for ODBC data sources:

1 The section for your database profile in the PowerBuilder initialization file (in the development environment) or the value of the transaction object DBParm property (in a PowerBuilder application)

2 The section for your ODBC driver in the PBODB60 initialization file

If PowerBuilder does not find a Block value in these locations, it uses the default Block value for the DBMS you are accessing.

Turning off block fetching To turn off block fetching for an ODBC data source or Oracle database, set the Block parameter to 1.

Examples

To set the blocking factor for DataWindow objects to 50 rows:

- **Database profile** Type the following in the Retrieve Blocking Factor box on the Transaction tab in the Database Profile Setup dialog box:

```
50
```

- **PowerBuilder application script** Type the following in a PowerBuilder application script:

```
SQLCA.dbParm = "Block = 50"
```

Using the examples in a PowerBuilder application script If you specify DBParm parameters in your database profile, the PowerScript 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.

Block (Sybase System 10 and System 11)

Description

Specifies the internal blocking factor used by the Sybase Client Library (CT-Lib) interface when declaring a cursor. The blocking factor determines the number of rows that are fetched from the database at one time when CT-Lib makes a physical request for data.

When you are connected to a Sybase System 10.x or System 11.x database, the Block DBParm parameter applies only to declared cursors and not to DataWindow objects.

Applies to

SYC and SYD Sybase Systems 10.x and 11.x

Syntax

Block = *blocking_factor*

Parameter	Description
blocking_factor	The number of rows that are fetched from the database at one time when CT-Lib makes a physical request for data. The default blocking factor is 100 rows

Default value

Block = 100

Examples

Example 1 To set the blocking factor to 1000 rows:

- **Database profile** Type the following in the Retrieve Blocking Factor box on the Transaction tab in the Database Profile Setup dialog box:

```
1000
```

- **PowerBuilder application script** Type the following in a PowerBuilder application script:

```
SQLCA.dbParm = "Block = 1000"
```

Example 2 The following embedded SQL statements show how to set the blocking factor in a PowerBuilder application script and use it to declare a cursor. These statements set the blocking factor to 1000 rows and declare a cursor that uses this internal blocking factor.

```
SQLCA.dbParm = "Block = 1000"
DECLARE dept_cursor CURSOR FOR
   SELECT dept_id, dept_name FROM department
   USING SQLCA;
OPEN dept_cursor;
```

Using the examples in a PowerBuilder application script If you specify DBParm parameters in your database profile, the PowerScript 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.

CharSet

Description

Specifies the character set you want the Sybase Open Client software to use when connecting to a Sybase SQL Server database in PowerBuilder.

When to specify CharSet
You must specify the CharSet DBParm parameter *before* connecting to a Sybase SQL Server database in PowerBuilder.

Applies to

SYT Sybase SQL Server (Windows NT only)
SYC and SYD Sybase Systems 10.x and 11.x

Syntax

CharSet = '*character_set*'

Default value

None

Usage

Sybase SQL Server System 10 and System 11 When you specify a value for CharSet, PowerBuilder does the following:

- Allocates a CS_LOCALE structure for this connection
- Sets the CS_SYB_CHARSET value to the character set you specify
- Sets the SQL Server CS_LOC_PROP connection property with the new locale information

Sybase SQL Server on Windows NT When you specify a value for CharSet, PowerBuilder invokes the Sybase Open Client DBSETLCHARSET() function to specify a character set in the LOGINREC structure.

Overriding the Locale DBParm (System 10.x and System 11.x only) If you have previously set a value for the Locale DBParm parameter, which includes settings for the language and character set you want to use, you can override the character set value by specifying a new value for the CharSet DBParm and reconnecting to the database. This applies *only* to Sybase System 10.x and System 11.x connections.

Examples

To set the character set to iso_1:

- **Database profile** Type the following in the Character Set box on the Connection tab or Regional Settings tab in the Database Profile Setup dialog box:

```
iso_1
```

- **PowerBuilder application script** Type the following in a PowerBuilder application script:

```
SQLCA.dbParm = "CharSet = 'iso_1'"
```

Using the examples in a PowerBuilder application script If you specify DBParm parameters in your database profile, the PowerScript 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.

See also

Language
Locale

CommitOnDisconnect

Description

Specifies whether PowerBuilder should commit (the default behavior) or rollback all previously uncommitted database updates before disconnecting from a data source.

When to specify CommitOnDisconnect
You must specify a value for CommitOnDisconnect *before* connecting to the database in PowerBuilder.

Applies to

All Powersoft database interfaces

Syntax

CommitOnDisconnect = '*value*'

Parameter	Description
value	Specifies whether PowerBuilder should commit or rollback all previously uncommitted database updates before disconnecting from a data source. Values are: ♦ **Yes** (Default) PowerBuilder will commit all uncommitted database updates when an application closes or when an explicit DISCONNECT statement is issued in a PowerBuilder script ♦ **No** PowerBuilder will rollback all uncommitted database updates when an application closes or when an explicit DISCONNECT statement is issued in a PowerBuilder script. With this setting, PowerBuilder does not automatically commit updates when you disconnect from the database

Default value

CommitOnDisconnect = 'Yes'

Usage

Set CommitOnDisconnect to No if you want PowerBuilder to rollback uncommitted database updates when you disconnect from the database, instead of automatically committing them.

Examples

To tell PowerBuilder to rollback uncommitted database updates instead of committing them when disconnecting from the database, set CommitOnDisconnect to No as follows:

- **Database profile** Clear the Commit on Disconnect checkbox on the Connection tab in the Database Profile Setup dialog box.
- **PowerBuilder application script** Type the following in a PowerBuilder script:

```
SQLCA.DBParm = "CommitOnDisconnect = 'No'"
```

Using the examples in a PowerBuilder application script If you specify DBParm parameters in your database profile, the PowerScript 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.

ConnectOption

Description

Sets driver-specific connection options when you are accessing an ODBC data source in PowerBuilder. These options specify the following:

- How the ODBC driver prompts for additional connection information
- The type of security for a Microsoft SQL Server 6.x connection
- Whether the ODBC Driver Manager Trace is on or off and what trace file it uses
- Whether cursors are closed or left open on a SQLTransact call
- How temporary stored procedures are treated for a SQLPrepare call

Certain ConnectOption parameters apply to all ODBC drivers, while others apply only to particular ODBC drivers.

FOR INFO For information on each ConnectOption parameter and whether you can use it with your ODBC driver, see the table in the Syntax section.

When to specify ConnectOption

You must specify the ConnectOption DBParm parameter *before* connecting to an ODBC data source in PowerBuilder. The ConnectOption settings take effect when you connect to the database.

Applies to

ODBC (if driver and backend DBMS support this feature)

Syntax

ConnectOption = ' **SQL_DRIVER_CONNECT**,*value*;
SQL_INTEGRATED_SECURITY,*value*;
SQL_OPT_TRACE,*value*;**SQL_OPT_TRACEFILE**,*value*;
SQL_PRESERVE_CURSORS,*value*;
SQL_USE_PROCEDURE_FOR_PREPARE,*value* '

The following table lists the applicable ODBC drivers, purpose, and values for each ConnectOption parameter:

Parameter	Description
SQL_DRIVER_CONNECT	**Driver** Any ODBC driver that supports the SQLDriverConnect API call **Purpose** Specifies how the ODBC driver prompts for additional connection information (such as the user ID and password) when connecting to an ODBC data source **Values** The values you can specify are: ♦ **SQL_DRIVER_COMPLETE** (Default) If the connection string contains correct and sufficient information to connect, the driver connects to the specified data source. If any information is incorrect or missing, the driver displays one or more dialog boxes to prompt for the required connection parameters. The driver then connects to the specified data source ♦ **SQL_DRIVER_COMPLETE_REQUIRED** The driver takes the same actions as it does when SQL_DRIVER_COMPLETE is set. In addition, the driver disables the controls for any information not required to connect to the data source ♦ **SQL_DRIVER_PROMPT** The driver displays one or more dialog boxes to prompt for the required connection parameters. The driver then connects to the specified data source and builds a connection string from the information specified in the dialog boxes ♦ **SQL_DRIVER_NOPROMPT** If the connection string contains correct and sufficient information to connect, the driver connects to the specified data source. If any information is incorrect or missing, the driver returns an error

Parameter	Description
SQL_INTEGRATED_SECURITY	**Driver** Microsoft SQL Server 6.x ODBC driver (not supplied with PowerBuilder) **Purpose** Specifies the type of connection to the Microsoft SQL Server 6.x database server **Values** The values you can specify are: ♦ **SQL_IS_OFF** (Default) Request a normal (nontrusted) connection to SQL Server 6.x using standard security. If you specify SQL_IS_OFF, you cannot request a trusted connection to SQL Server 6.x using integrated security ♦ **SQL_IS_ON** Request a trusted connection to SQL Server 6.x using integrated security regardless of the login security currently in use on the database server FOR INFO For more about security mechanisms in Microsoft SQL Server 6.x, see the Secure DBParm parameter
SQL_OPT_TRACE	**Driver** Any ODBC driver **Purpose** Turns on or turns off the ODBC Driver Manager Trace in PowerBuilder to troubleshoot a connection to an ODBC data source. The ODBC Driver Manager Trace provides detailed information about the ODBC API function calls that PowerBuilder makes when connected to an ODBC data source **Values** The values you can specify are: ♦ **SQL_OPT_TRACE_OFF** (Default) Turns off the ODBC Driver Manager Trace ♦ **SQL_OPT_TRACE_ON** Turns on the ODBC Driver Manager Trace FOR INFO For instructions on using the ODBC Driver Manager Trace, see "About ODBC Driver Manager Trace" on page 374.

Parameter	Description
SQL_OPT_TRACEFILE	**Driver** Any ODBC driver **Purpose** Specifies the name of the trace file where you want PowerBuilder to send the output of the 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) **Values** You can specify any filename for the trace file, following the naming conventions of your operating system. By default, if tracing is on and you have not specified a trace file, PowerBuilder sends ODBC Driver Manager Trace output to the file \SQL.LOG
SQL_PRESERVE_ CURSORS	**Driver** Microsoft SQL Server 6.x ODBC driver (not supplied with PowerBuilder) **Purpose** Specifies whether cursors are closed or left open on a SQLTransact call **Values** The values you can specify are: ♦ **SQL_PC_OFF** (Default) Close all cursors on a SQLTransact call ♦ **SQL_PC_ON** Keep server cursors open on a SQLTransact call
SQL_USE_PROCEDURE_ FOR_PREPARE	**Driver** Microsoft SQL Server 6.x ODBC driver (not supplied with PowerBuilder) **Purpose** Specifies how temporary stored procedures are treated for a SQLPrepare call **Values** The values you can specify are: ♦ **SQL_UP_ON** (Default) Generate temporary stored procedures for a SQLPrepare call ♦ **SQL_UP_OFF** Do not generate temporary stored procedures for a SQLPrepare call. The SQL statement is stored, compiled, and run at execution time. Syntax error checking does not occur until execution time ♦ **SQL_UP_ON_DROP** Explicitly drop temporary stored procedures for a subsequent SQLPrepare call or when a statement handle (hstmt) is freed for reuse

Default value

ConnectOption = 'SQL_DRIVER_CONNECT, SQL_DRIVER_COMPLETE;
SQL_INTEGRATED_SECURITY,SQL_IS_OFF;
SQL_OPT_TRACE,SQL_OPT_TRACE_OFF;
SQL_PRESERVE_CURSORS,SQL_PC_OFF;
SQL_USE_PROCEDURE_FOR_PREPARE,SQL_UP_ON'

Usage

Microsoft SQL Server 6.x ODBC driver The ConnectOption DBParm parameter applies only if you are accessing a SQL Server 6.x database with the Microsoft ODBC SQL Server 6.x driver. It does *not* apply if you are using the Powersoft SQL Server 6.x database interface to access the database.

You must obtain the Microsoft SQL Server 6.x ODBC driver from Microsoft Corporation. This driver is *not* supplied with PowerBuilder.

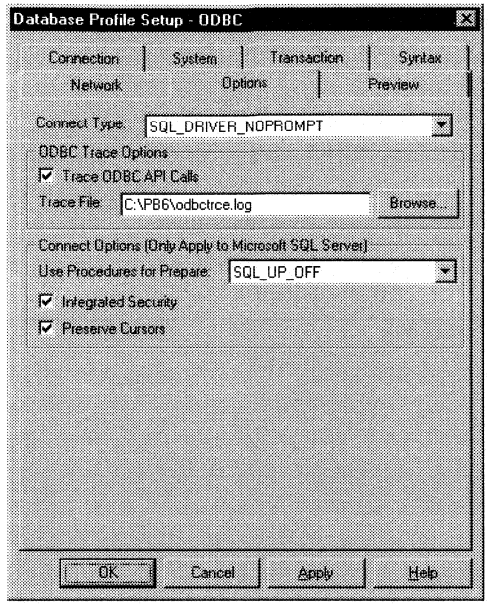
Examples

To specify nondefault options for the ConnectOption DBParm parameter:

- **Database profile** Complete the Options tab in the Database Profile Setup - ODBC dialog box. Each ConnectOption parameter corresponds to an option in the dialog box, as follows:

ConnectOption parameter	Corresponding option
SQL_DRIVER_CONNECT	Connect Type
SQL_INTEGRATED_SECURITY	Integrated Security
SQL_OPT_TRACE	Trace ODBC API Calls
SQL_OPT_TRACEFILE	Trace File
SQL_PRESERVE_CURSORS	Preserve Cursors
SQL_USE_PROCEDURE_FOR_PREPARE	Use Procedure for Prepare

For example:



- **PowerBuilder application script** Type the following in a PowerBuilder application script:

```
SQLCA.dbParm = "ConnectOption =
    'SQL_DRIVER_CONNECT,SQL_DRIVER_NOPROMPT;
    SQL_INTEGRATED_SECURITY,SQL_IS_ON;
    SQL_OPT_TRACE,SQL_OPT_TRACE_ON;
    SQL_OPT_TRACEFILE,C:\PB6\odbctrce.log;
    SQL_PRESERVE_CURSORS,SQL_PC_ON;
    SQL_USE_PROCEDURE_FOR_PREPARE,SQL_UP_OFF'"
```

Using the examples in a PowerBuilder application script If you specify DBParm parameters in your database profile, the PowerScript 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.

See also

Secure

ConnectString

Description

Specifies the parameters required to connect to an ODBC data source. PowerBuilder uses these parameters to connect to the database.

Applies to

ODBC (if driver and backend DBMS support this feature)

Syntax

The ConnectString syntax displays on a single line. You must enclose the entire ConnectString in single quotes and separate parameters within the ConnectString with semicolons.

ConnectString = ' **DSN** = *data_source_name*; {**UID** = *user_ID*; **PWD** = *password*; *driver_specific_parameters*}'

Parameter	Description
data_source_name	A name that identifies the data source
user_ID	(Optional) The user ID required to connect to the data source
password	(Optional) The password required by *user_ID* to connect to the data source
driver_specific_parameters	(Optional) Any other driver-specific parameters required to connect For example, some ODBC drivers specify the data source directory here. If you are using the INTERSOLV Btrieve ODBC driver, you can specify the value **CDB = 1** here to create a new Scalable SQL (formerly NetWare SQL) data dictionary if one does not already exist

Default value

None

Usage

On Windows and Macintosh On the Windows and Macintosh platforms, PowerBuilder generates the ConnectString automatically when you define an ODBC data source and copies it to the DBParm box in the Database Profile Setup dialog box. This happens before you connect to the data source in PowerBuilder.

Therefore, *you do not have to enter the ConnectString yourself* when defining an ODBC data source. However, you may need to edit the ConnectString value in the Database Profile Setup dialog box.

You can change the ConnectString parameter if necessary by editing it in the Database Profile Setup dialog box. For example, if you change the name of an existing ODBC data source, you should edit its database profile to update the connect string with the new DSN (data source name) value.

FOR INFO For instructions, see "Editing an ODBC data source definition" on page 304.

On UNIX When accessing an INFORMIX database in PowerBuilder on UNIX with the INTERSOLV INFORMIX ODBC driver, you must specify certain driver-specific parameters in the Database Profile Setup dialog box to define the INFORMIX data source.

FOR INFO For instructions, see "Defining the INFORMIX data source on UNIX" on page 80.

Examples

Example 1 This example shows a connect string for an ODBC data source that contains the data source name (DSN=Sales), user ID (UID=dba), and password (PWD=sql). Parameters within the connect string are separated by semicolons.

- **Database profile** On the Connection tab in the Database Profile Setup dialog box, select Sales from the Data Source dropdown listbox, select the User ID checkbox and type dba, and select the Password checkbox and type sql.
- **PowerBuilder application script** Type the following in a PowerBuilder application script:

```
SQLCA.dbParm = "ConnectString ='DSN=Sales;UID=dba;
   PWD=sql'"
```

Example 2 This example shows a connect string for a Btrieve data source accessed with the INTERSOLV Btrieve ODBC driver. The connect string consists of two parameters: DSN and CDB (to create a new Scalable SQL data dictionary). Parameters within the connect string are separated by semicolons.

- **Database profile** Type the following in the Driver-Specific Parameters box on the Connection tab in the Database Profile Setup dialog box:

```
DSN=Btrieve;CDB=1
```

- **PowerBuilder application script** To specify this statement in a PowerBuilder application script, type the following:

```
SQLCA.dbParm = "ConnectString ='DSN=Btrieve;CDB=1'"
```

Using the examples in a PowerBuilder application script If you specify DBParm parameters in your database profile, the PowerScript 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.

ConversionTable

Description

When PowerBuilder is connected to an IBM DB2 database through the Sybase InformationConnect DB2 Gateway interface, the ConversionTable DBParm parameter specifies the InformationConnect Gateway conversion profile you want to use to convert DB2 data types to SQL Server data types.

By default, PowerBuilder uses the STD (standard) conversion profile. You should set ConversionTable if you want to use a conversion profile *other* than the standard one provided by the InformationConnect Gateway.

What PowerBuilder does When you specify an InformationConnect Gateway conversion profile, all data types are converted as specified by that profile, *with certain exceptions.*

Regardless of the value you specify for ConversionTable, PowerBuilder *always* converts the date, decimal, time, and timestamp data types as follows:

Data type	Always converted to
Date	ISO
Decimal	CHAR
Time	ISO
Timestamp	ISO

You cannot specify conversions for individual DB2 data types by using the ConversionTable parameter.

FOR INFO For a list of the data type conversions in each of the supported InformationConnect Gateway profiles, see the table in the Usage section.

When to specify ConversionTable
You must specify the ConversionTable DBParm parameter *before* connecting to a DB2 database in PowerBuilder through the Sybase InformationConnect Gateway interface.

Applies to

MDI Sybase InformationConnect DB2 Gateway interface

Syntax

ConversionTable = '*value*'

Parameter	Description
value	Specifies the InformationConnect Gateway conversion profile you want to use to convert DB2 data types to SQL Server data types. When you specify a profile, all data types except date, decimal, time, and timestamp are converted as specified by that profile. Values are: ♦ **STD** (Default) ♦ **COBOL** ♦ **CHAR** ♦ **IXF** Regardless of the value you specify for ConversionTable, PowerBuilder always converts date, time, and timestamp data types to ISO, and decimal data types to CHAR FOR INFO For a list of the data type conversions in each of the InformationConnect profiles, see the table in the Usage section below

Default value

ConversionTable = 'STD '

Usage

All data types except date, decimal, time, and timestamp When you specify a value for ConversionTable, PowerBuilder internally executes the following statement to use the specified data type conversion profile:

```
SET CONVERT ALL = conversion_profile_name
```

If you do not specify a value for ConversionTable, PowerBuilder internally executes the following statement to use the STD (standard) conversion profile provided by the InformationConnect Gateway:

```
SET CONVERT ALL = STD
```

Date, decimal, time, and timestamp data types Regardless of the value you specify for ConversionTable, PowerBuilder internally executes the following statements to convert the date, decimal, time, and timestamp data types:

```
SET CONVERT DATE = ISO
SET CONVERT DECIMAL = CHAR
SET CONVERT TIME = ISO
SET CONVERT TIMESTAMP = ISO
```

No validation performed When you specify a conversion profile with the ConversionTable DBParm, PowerBuilder does not validate the mapping between your data and the profile specifications, and ignores any errors or data type mismatches that may occur.

InformationConnect Gateway conversion profiles The following table lists the data type conversions in each of the InformationConnect Gateway conversion profiles. (See your Sybase InformationConnect DB2 Gateway documentation for the most up-to-date information on DB2-to-SQL Server data type conversion profiles.)

This table does not include the date, decimal, time, and timestamp data types because PowerBuilder handles the conversions for these data types differently than the InformationConnect Gateway. PowerBuilder always converts date, time, and timestamp data types to ISO, and decimal data types to CHAR.

DB2 data type	STD	COBOL	CHAR	IXF
ByteInt	Int2	Int2	Char	IXF
SmallInt	Int2	Int2	Char	IXF
Integer	Int4	Int4	Char	IXF
Char	Char	Char	Char	IXF
Char (for bit data)	Binary	Binary	Binary	IXF *
VarChar	VarChar	VarChar	VarChar	IXF
VarChar (for bit data)	VarBinary	VarBinary	VarBinary	IXF *
LVarChar	Text	Text	Text	IXF
LVarChar (for bit data)	Image	Image	Image	IXF *
Float	Flt8	Flt8	Char	IXF

* Special extensions to the IBM Information eXchange Format (IXF) specification created to handle binary data. See your IBM documentation for information on IXF.

Examples

To specify CHAR as the name of the InformationConnect Gateway conversion profile you want to use:

- **Database profile** Select Character from the Conversion Table dropdown listbox on the Syntax tab in the Database Profile Setup dialog box.
- **PowerBuilder application script** Type the following in a PowerBuilder application script:

```
SQLCA.dbParm = "ConversionTable = 'CHAR'"
```

What happens internally When you connect to the DB2 database after setting ConversionTable with this value, PowerBuilder internally executes the following statement to use the CHAR conversion profile:

```
SET CONVERT ALL = CHAR
```

Using the examples in a PowerBuilder application script If you specify DBParm parameters in your database profile, the PowerScript 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.

CursorLib

Description

Specifies the cursor library to use when connecting to an ODBC data source.

Applies to

ODBC (if driver and backend DBMS support this feature)

Syntax

CursorLib = '*value*'

Parameter	Description
value	The cursor library to use when connecting to an ODBC data source. Values are: ♦ **ODBC_Cur_Lib** Use the ODBC Version 2.0 or higher cursor library ♦ **If_Needed** Use the ODBC Version 2.0 or higher cursor library if your ODBC driver does not support cursors ♦ **Driver_Cursors** (Default) Use your data source's native cursor support

Default value

CursorLib = 'Driver_Cursors'

Examples

To specify use of the ODBC Version 2.0 or higher cursor library when connecting to an ODBC data source:

- **Database profile** Select Cursor Library from the Cursor Library dropdown listbox on the Transaction tab in the Database Profile Setup dialog box.
- **PowerBuilder application script** Type the following in a PowerBuilder application script:

```
SQLCA.dbParm = "CursorLib = 'ODBC_Cur_Lib'"
```

Using the examples in a PowerBuilder application script If you specify DBParm parameters in your database profile, the PowerScript 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.

CursorLock (ODBC)

Description

When used with the CursorScroll parameter, specifies locking options for cursors in ODBC data source.

The values you can set for CursorLock control two aspects of cursor locking:

- **Concurrent access** Ensures that multiple users can simultaneously access data that is accurate and current.
- **Collision detection** Detects collisions that occur when multiple users update the same data at the same time.

Applies to

ODBC (if driver and backend DBMS support this feature)

Syntax

CursorLock = '*lock_value*'

Parameter	Description
lock_value	Specifies the type of locking you want to use for ODBC cursors. Values are: ♦ **Lock** Use the lowest level of locking sufficient to allow updates on table rows ♦ **Opt** Use **optimistic concurrency control**. This means that table rows are not locked against updates by other users. To detect collisions, compare row versions or timestamps ♦ **OptValue** Use optimistic concurrency control. This means that table rows are not locked against updates by other users. To detect collisions, compare selected values with their previous values ♦ **ReadOnly** Prohibit updates on table rows by any user FOR INFO For more about how the ODBC standard defines lock values, see your ODBC documentation

Default value

If you do not specify a value for CursorLock, PowerBuilder defaults to the cursor lock setting specified by your ODBC driver.

Examples

To set scrolling and locking options for cursors in an ODBC data source:

- **Database profile** Select Dynamic Scrolling from the Scrolling Options dropdown listbox, and Optimistic Using Values from the Locking dropdown listbox on the Transaction tab in the Database Profile Setup dialog box.
- **PowerBuilder application script** Type the following in a PowerBuilder application script:

```
SQLCA.dbParm =
   "CursorScroll='Dynamic',CursorLock='OptValue'"
```

Using the examples in a PowerBuilder application script If you specify DBParm parameters in your database profile, the PowerScript 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.

See also

CursorScroll (ODBC)

CursorLock (SQL Server)

Description

When used with the Release and CursorScroll parameters, specifies locking options for cursors that access a SQL Server database Version 4.2 or higher.

The values you can set for CursorLock control two aspects of cursor locking:

- **Concurrent access** Ensures that multiple users can simultaneously access data that is accurate and current.
- **Collision detection** Detects collisions that occur when multiple users update the same data at the same time.

Using CursorLock with SQL Server
To use the CursorLock DBParm parameter with a SQL Server database, you must set the Release DBParm parameter to 4.2 when you create the SQL Server database profile (as shown in the Examples section below). This allows you to use DB-Library cursor support in SQL Server Version 4.2 or higher.

Applies to

MSS Microsoft SQL Server 6.x
SYB and SYT SQL Server 4.x

Syntax

CursorLock = '*lock_value*'

Parameter	Description
lock_value	Specifies the type of locking you want to use for SQL Server cursors. Values are: ♦ **Lock** ♦ **Opt** (Default) ♦ **OptVal** ♦ **ReadOnly** FOR INFO For more about how these values control concurrent access and detect collisions, see the table in the Usage section below

Default value

CursorLock = 'Opt'

Usage

The following table briefly describes how each CursorLock value controls concurrent access and detects collisions in a SQL Server database. (For more information, see your SQL Server documentation.)

CursorLock value	Concurrency control	Collision detection
Lock	Locks table rows when they are fetched inside a SQL transaction. No other user can update or read these rows	Collision detection is unnecessary because locking the rows ensures that any updates made by the owner of the cursor will succeed
Opt	Does not lock table rows. Other users can update or read these rows. This is called **optimistic concurrency control**	Compares timestamps if available. If timestamps are not available, saves and compares the values of all nontext, nonimage columns in the tables with their previous values
OptVal	Does not lock table rows. Other users can update or read these rows. This is called **optimistic concurrency control**	Compares selected values whether or not a timestamp is available
ReadOnly	Prohibits updates on table rows by any user	Collision detection is unnecessary because updates on the rows are prohibited

Examples

To set scrolling and locking options for cursors in a SQL Server database that is Version 4.2 or higher:

- **Database profile** On the Connection tab in the Database Profile Setup dialog box, select the Release 4.2 checkbox. On the Transaction tab, select Forward Scrolling Only from the Scrolling Options dropdown listbox and OptVal from the Locking dropdown listbox.
- **PowerBuilder application script** Type the following in a PowerBuilder application script:

```
SQLCA.dbParm = "Release='4.2',
   CursorScroll='Forward',CursorLock='OptVal'"
```

Using the examples in a PowerBuilder application script If you specify DBParm parameters in your database profile, the PowerScript 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.

See also

CursorScroll (SQL Server)
Release (SQL Server 4.x)

CursorScroll (ODBC)

Description

When used with the CursorLock parameter, specifies scrolling options for cursors in an ODBC data source.

The location of a cursor indicates the current position in the result set produced by a SQL statement. **Scrolling** allows a cursor to move through the data in a result set one row at a time.

Applies to

ODBC (if driver and backend DBMS support this feature)

Syntax

CursorScroll = '*scroll_value*'

Parameter	Description
scroll_value	Specifies the type of scrolling you want to use for ODBC cursors. Values are: ♦ **Forward** The cursor only scrolls forward through the result set ♦ **Static** The data in the result set does not change ♦ **KeySet** Specifies that the cursor is **keyset-driven**. When a keyset-driven cursor is opened, the driver saves keys for the *entire result set.* As the cursor scrolls through the result set, the driver uses the keys in this **keyset** to retrieve the current values for each row ♦ **Dynamic** The driver saves and uses only the keys for the rows specified in the rowset ♦ For large result sets, it may be impractical to use a keyset-driven cursor that requires the driver to save keys for the entire result set. Instead, you can use a **mixed cursor** by specifying a 32-bit integer value that is the number of rows in your keyset (see Example 2). This number is typically smaller than the result set. The default keyset size is 0 A mixed cursor uses KeySet scrolling within the specified keyset and Dynamic scrolling outside the keyset

Default value

If you do not specify a value for CursorScroll, PowerBuilder defaults to the cursor scroll settings specified for your ODBC data source driver.

Usage

Dynamic scrolling in SQL Anywhere 5.x and Watcom SQL 4.0b If you are accessing a SQL Anywhere Version 5.x or Watcom SQL Version 4.0b database and want to execute an embedded SQL FETCH FIRST, FETCH PRIOR, or FETCH LAST statement when you retrieve a row from a cursor, you *must* set CursorScroll to 'Dynamic'. Otherwise, the default for these DBMSs is to do forward scrolling (FETCH NEXT) only.

This is different from the behavior in Watcom SQL Version 4.0, which did dynamic cursor scrolling by default. This default behavior changed as of Version 4.0b when the Watcom SQL ODBC driver was upgraded to be ODBC 2.0-compliant.

Examples

Example 1 To set scrolling and locking options for cursors in an ODBC data source:

- **Database profile** Select Dynamic Scrolling from the Scrolling Options dropdown listbox and Optimistic Using Values from the Locking dropdown listbox on the Transaction tab in the Database Profile Setup dialog box.
- **PowerBuilder application script** Type the following in a PowerBuilder application script:

```
SQLCA.dbParm = "CursorScroll = 'Dynamic',
   CursorLock = 'OptValue'"
```

Example 2 This example sets the number of rows in the keyset to 100. Assume that the entire result set has 1000 rows. When the cursor is opened, the driver saves keys for the first 100 rows of the result set. It then retrieves the next block of 100 keys until the entire result set is retrieved.

- **Database profile** Type the following in the Scrolling Options box on the Transaction tab in the Database Profile Setup dialog box:

```
100
```

- **PowerBuilder application script** Type the following in a PowerBuilder application script:

```
SQLCA.dbParm = "CursorScroll = 100"
```

Using the examples in a PowerBuilder application script If you specify DBParm parameters in your database profile, the PowerScript 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.

See also

CursorLock (ODBC)

CursorScroll (SQL Server)

Description

Specifies scrolling options for cursors in a SQL Server database Version 4.2 or higher or in a Microsoft SQL Server 6.x database. The CursorScroll parameter is often used together with the CursorLock and (for SQL Server 4.x databases) Release parameters.

The location of a cursor indicates the current position in the result set produced by a SQL statement. **Scrolling** allows a cursor to move through the data in a result set one row at a time.

Using CursorScroll with SQL Server 4.x
To use the CursorScroll DBParm parameter with a SQL Server 4.x database, you *must* set the Release DBParm parameter to 4.2 when you create the SQL Server database profile (as shown in the Examples section below). This allows you to use DB-Library cursor support in SQL Server Version 4.2 or higher.

Applies to

MSS Microsoft SQL Server 6.x
SYB and SYT SQL Server 4.x

Syntax

CursorScroll = '*scroll_value*'

Parameter	Description
scroll_value	Specifies the type of scrolling you want to use for SQL Server cursors. Values are: ♦ **Forward** The cursor only scrolls forward through the result set. Values, order, and membership in the result set do not change while the cursor is open ♦ **KeySet** Specifies that the cursor is **keyset-driven**. When a keyset-driven cursor is opened, the driver saves keys for the *entire result set*. As the cursor scrolls through the result set, the driver uses the keys in this **keyset** to retrieve the current values for each row ♦ **Dynamic** (Default) The driver saves and uses only the keys for the rows specified in the rowset. Values, order, and membership in the result set can all change ♦ **Insensitive (Microsoft SQL Server 6.x only)** An **insensitive keyset cursor** makes a temporary copy of the data that the cursor will use. All requests to the cursor are answered from this temporary table. If other users make changes to any rows after the copy is made, the insensitive cursor will not reflect these changes An insensitive cursor prohibits updates on table rows by any user. Therefore, when specifying the CursorScroll DBParm parameter with the CursorLock DBParm parameter, set the value of CursorLock to ReadOnly to prohibit updates (see Example 2 below) ♦ For large result sets, it may be impractical to use a keyset-driven cursor that requires the driver to save keys for the entire result set. Instead, you can use a **mixed cursor** by specifying a 32-bit integer value that is the number of rows in your keyset (see Example 3 below). This number is typically smaller than the result set. The default keyset size is 0 A mixed cursor uses KeySet scrolling within the specified keyset and Dynamic scrolling outside the keyset Using a mixed cursor may be useful if the number of rows in your keyset exceeds the memory capacity of your computer. When you use KeySet scrolling, SQL Server tries to create a buffer on your computer that is large enough to hold all the rows in your keyset. If the number of rows exceeds your computer's memory capacity, you may get an out-of-memory error

Default value CursorScroll = 'Dynamic'

Examples

Example 1 To set scrolling and locking options for cursors in a SQL Server database that is Version 4.2 or higher:

- **Database profile** On the Connection tab in the Database Profile Setup dialog box, select the Release 4.2 checkbox. On the Transaction tab, select Dynamic Scrolling from the Scrolling Options dropdown listbox and Optimistic Using Values from the Locking dropdown listbox.
- **PowerBuilder application script** Type the following in a PowerBuilder application script:

```
SQLCA.dbParm = "Release = '4.2',
   CursorScroll = 'Dynamic',CursorLock = 'OptVal'"
```

Example 2 To specify Insensitive cursor scrolling and ReadOnly cursor locking for a Microsoft SQL Server 6.x database:

- **Database profile** Select Insensitive Keyset from the Scrolling Options dropdown listbox and Read Only from the Locking dropdown listbox on the Transaction tab in the Database Profile Setup dialog box.
- **PowerBuilder application script** Type the following in a PowerBuilder application script:

```
SQLCA.dbParm = "CursorScroll = 'Insensitive',
   CursorLock = 'ReadOnly'"
```

Example 3 This statement sets the number of rows in the keyset to 100 for a SQL Server database that is Version 4.2 or higher. Assume that the entire result set has 1000 rows. When the cursor is opened, the driver saves keys for the first 100 rows of the result set. It then retrieves the next block of 100 keys until the entire result set is retrieved.

- **Database profile** On the Connection tab in the Database Profile Setup dialog box, select the Release 4.2 checkbox. On the Transaction tab, type the following in the Scrolling Options box:

```
100
```

- **PowerBuilder application script** Type the following in a PowerBuilder application script:

```
SQLCA.dbParm = "Release = '4.2',CursorScroll = 100"
```

Using the examples in a PowerBuilder application script If you specify DBParm parameters in your database profile, the PowerScript 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.

See also

CursorLock (SQL Server)
Release (SQL Server 4.x)

CursorUpdate

Description

Specifies whether cursors in a Sybase Systems 10.x or 11.x database are declared read-only or updatable.

By default, cursors are declared read-only.

Applies to

SYC and SYD Sybase Systems 10.x and 11.x

Syntax

CursorUpdate = *value*

Parameter	Description
value	A number that specifies whether Sybase System 10.x or System 11.x cursors are declared read-only or updatable. Values are: ♦ **0** (Default) Cursors are declared read-only. Sybase Client Library cursor declarations include the CS_READ_ONLY option ♦ **1** Cursors are declared updatable. Sybase Client Library cursor declarations include the CS_FOR_UPDATE option. This option applies to all updatable columns in the table

Default value

CursorUpdate = 0

Usage

Set the CursorUpdate parameter to 1 to declare updatable cursors if you plan to use either of the following SQL statements in your application. (In these statements, *table* represents the table name and *cursor* represents the cursor name.)

DELETE FROM *table* **WHERE CURRENT OF** *cursor*

UPDATE *table* **SET** *set_clause* **WHERE CURRENT OF** *cursor*

If you are communicating with the database in a PowerBuilder script, you can reset the CursorUpdate value at any time before or after the transaction object has connected to the database.

Examples

To specify that Sybase System 10.x or System 11.x cursors are declared updatable:

- **Database profile** Select the Cursors Declared Updatable checkbox on the Transaction tab in the Database Profile Setup dialog box.
- **PowerBuilder application script** Type the following in a PowerBuilder application script:

```
SQLCA.dbParm = "CursorUpdate = 1"
```

Using the examples in a PowerBuilder application script If you specify DBParm parameters in your database profile, the PowerScript 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.

Date

Description

When you update data in the DataWindow painter, PowerBuilder builds a SQL UPDATE statement in the background. If you are updating an ODBC data source or Oracle table, the Date DBParm parameter determines how PowerBuilder specifies a date data type when it builds the SQL UPDATE statement.

Applies to

ODBC (if driver and backend DBMS support this feature)
Oracle (all interfaces)

Syntax

The syntax you use to specify the Date DBParm differs slightly for ODBC and Oracle databases.

In the PowerBuilder development environment, the Database Profile Setup dialog box inserts special characters (quotes and backslashes) where needed, so you can specify just the Powersoft date format.

In a PowerBuilder application script, you must use the following syntax:

ODBC syntax If you are accessing an ODBC data source through the Powersoft ODBC interface, use the following syntax for Date. PowerBuilder parses the backslash followed by two single quotes (\") as a single quote when it builds the SQL UPDATE statement.

Date = ' \"*Powersoft_date_format*\" '

Oracle syntax If you are accessing an Oracle database through one of the Powersoft Oracle database interfaces, use the following syntax for Date. PowerBuilder parses each set of four consecutive single quotes ('''') as a single quote when it builds the SQL UPDATE statement.

Date = ' ""*Powersoft_date_format*"" '

Parameter	Description
' \"	**ODBC syntax** Type a single quote, followed by one space, followed by a backslash, followed by two single quotes. There is no space between the two single quotes and the beginning of the Powersoft date format
' ""	**Oracle syntax** Type a single quote, followed by one space, followed by four single quotes. There is no space between the four single quotes and the beginning of the Powersoft date format
Powersoft_date_format	The date format you want PowerBuilder to use when it builds a SQL UPDATE statement to update a data source in the DataWindow painter FOR INFO For more on Powersoft display formats, see the *PowerBuilder User's Guide*
\" '	**ODBC syntax** Type a backslash, followed by two single quotes, followed by one space, followed by a single quote. There is no space between the end of the Powersoft date format and the backslash
"" '	**Oracle syntax** Type four single quotes, followed by one space, followed by a single quote. There is no space between the end of the Powersoft date format and the four single quotes

Default value

The default value for Date depends on the DBMS you are accessing, as summarized in the following table:

DBMS	Date default value
ODBC	If no value is specified for the Date DBParm parameter, PowerBuilder looks for a date format in the section for your ODBC driver in the PBODB60 initialization file. If no date format is found in the initialization file, PowerBuilder uses the ODBC date format escape sequence
Oracle	The default Oracle date format FOR INFO For information, see your Oracle documentation

Examples

About these examples Assume you are updating a table named Employee by setting the Startdate column to 1997-04-23. This date is represented by the following Powersoft date format:

```
yyyy-mm-dd
```

Example 1 (ODBC syntax) To specify that PowerBuilder should use this format for the date data type when it builds the SQL UPDATE statement:

- **Database profile** Type the following in the Date Format box on the Syntax tab in the Database Profile Setup dialog box:

```
yyyy-mm-dd
```

- **PowerBuilder application script** Type the following in a PowerBuilder application script:

```
SQLCA.dbParm = "Date = ' \''yyyy-mm-dd\'' '"
```

What happens PowerBuilder builds the following SQL UPDATE statement to update the table:

```
UPDATE EMPLOYEE
SET STARTDATE = '1997-04-23'
```

Example 2 (Oracle syntax) To specify that PowerBuilder should use this format for the date data type when it builds the SQL UPDATE statement:

- **Database profile** Type the following in the Date format box on the Syntax tab in the Database Profile Setup dialog box:

```
yyyy-mm-dd
```

- **PowerBuilder application script** Type the following in a PowerBuilder application script:

```
SQLCA.dbParm = "Date = ' ''''yyyy-mm-dd'''' '"
```

What happens PowerBuilder builds the following SQL UPDATE statement to update the table:

```
UPDATE EMPLOYEE
SET STARTDATE = '1997-04-23'
```

Using the examples in a PowerBuilder application script If you specify the DBParm parameters in your database profile, the PowerScript 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.

See also

DateTime
Time

DateTime

Description

When you update data in the DataWindow painter, PowerBuilder builds a SQL UPDATE statement in the background. If you are updating an ODBC data source or Oracle table, the DateTime DBParm parameter determines how PowerBuilder specifies a DateTime data type when it builds the SQL UPDATE statement. (A DateTime data type contains both a date value and a time value.)

Applies to

ODBC (if driver and backend DBMS support this feature)
Oracle (all interfaces)

Syntax

The syntax you use to specify the DateTime DBParm differs slightly for ODBC and Oracle databases.

In the PowerBuilder development environment, the Database Profile Setup dialog box inserts special characters (quotes and backslashes) where needed, so you can specify just the Powersoft DateTime format.

In a PowerBuilder application script, you must use the following syntax:

ODBC syntax If you are accessing an ODBC data source through the Powersoft ODBC interface, use the following syntax for DateTime. PowerBuilder parses the backslash followed by two single quotes (\") as a single quote when it builds the SQL UPDATE statement.

DateTime = ' \"*Powersoft_DateTime_format*\" '

Oracle syntax If you are accessing an Oracle database through one of the Powersoft Oracle database interfaces, use the following syntax for DateTime. PowerBuilder parses each set of four consecutive single quotes ("") as a single quote when it builds the SQL UPDATE statement.

DateTime = ' ""*Powersoft_DateTime_format*"" '

Parameter	Description
' \"	**ODBC syntax** Type a single quote, followed by one space, followed by a backslash, followed by two single quotes. There is no space between the two single quotes and the beginning of the Powersoft DateTime format
' ""	**Oracle syntax** Type a single quote, followed by one space, followed by four single quotes. There is no space between the four single quotes and the beginning of the Powersoft date format

Parameter	Description
Powersoft_Date Time_format	The DateTime format you want PowerBuilder to use when it builds a SQL UPDATE statement to update a data source in the DataWindow painter FOR INFO For more on Powersoft display formats, see the *PowerBuilder User's Guide*
\" ' '''' '	**ODBC syntax** Type a backslash, followed by two single quotes, followed by one space, followed by a single quote. There is no space between the end of the Powersoft date format and the backslash **Oracle syntax** Type four single quotes, followed by one space, followed by a single quote. There is no space between the end of the Powersoft DateTime format and the four single quotes

Default value

The default value for DateTime depends on the DBMS you are accessing, as summarized in the following table:

DBMS	Date default value
ODBC	If no value is specified for the DateTime DBParm parameter, PowerBuilder looks for a DateTime format in the section for your ODBC driver in the PBODB60 initialization file. If no DateTime format is found in the initialization file, PowerBuilder uses the ODBC DateTime format escape sequence
Oracle	The default Oracle DateTime format FOR INFO For information, see your Oracle documentation

Examples

About these examples Assume you are updating a table named Files by setting the Timestamp column to 5/2/97 3:45 pm. This DateTime is represented by the following Powersoft DateTime format:

Example 1 (ODBC syntax) To specify that PowerBuilder should use this format for the DateTime data type when it builds the SQL UPDATE statement:

- **Database profile** Type the following in the DateTime Format box on the Syntax tab in the Database Profile Setup dialog box:

```
m/d/yy h:mm am/pm
```

- **PowerBuilder application script** Type the following in a PowerBuilder application script:

```
SQLCA.dbParm = "DateTime =
   ' \''m/d/yy h:mm am/pm\'' '"
```

What happens PowerBuilder builds the following SQL UPDATE statement to update the table:

```
UPDATE FILES
SET TIMESTAMP = '5/2/97 3:45 pm'
```

Example 2 (Oracle syntax) To specify that PowerBuilder should use this format for the DateTime data type when it builds the SQL UPDATE statement:

- **Database profile** Type the following in the DateTime Format box on the Syntax tab in the Database Profile Setup dialog box:

```
m/d/yy h:mm am/pm
```

- **PowerBuilder application script** Type the following in a PowerBuilder application script:

```
SQLCA.dbParm="DateTime =
   ' ''''m/d/yy h:mm am/pm'''' '"
```

What happens PowerBuilder builds the following SQL UPDATE statement to update the table:

```
UPDATE FILES
SET TIMESTAMP = '5/2/97 3:45 pm'
```

Using the examples in a PowerBuilder application script If you specify DBParm parameters in your database profile, the PowerScript 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.

See also

Date
Time

DateTimeAllowed

Description

For those interfaces that support it, DateTimeAllowed controls whether columns having a DateTime data type can appear as unique key columns in the WHERE clause of a SQL UPDATE or DELETE statement. PowerBuilder generates an UPDATE statement, or a DELETE statement followed by an INSERT statement, to update the database from a DataWindow object.

When you are working in the DataWindow painter, you specify which columns to include in the WHERE clause by selecting them from the Unique Key Columns list in the Specify Update Properties dialog box.

By default, DateTimeAllowed is set to 0 to prohibit DateTime columns from displaying in the Unique Key Columns list and, consequently, from appearing in the WHERE clause of an UPDATE or DELETE statement. When you set DateTimeAllowed to 1, any DateTime columns in your database table display in the Unique Key Columns list and can therefore be selected to appear in the WHERE clause of an UPDATE or DELETE statement.

When to specify DateTimeAllowed
You must specify a value for DateTimeAllowed *before* connecting to the database in PowerBuilder.

Applies to

MSS Microsoft SQL Server 6.x
SYB and SYT SQL Server 4.x
SYC and SYD Sybase Systems 10.x and 11.x

Syntax

DateTimeAllowed = *value*

Parameter	Description
value	Specifies whether you can use DateTime columns as unique key columns in a WHERE clause of a SQL UPDATE or DELETE statement generated by PowerBuilder to update the database. Values are: ◆ **0** (Default) Prohibit the use of DateTime columns in the WHERE clause of an UPDATE or DELETE statement. When DateTimeAllowed is set to 0, DateTime columns *do not display* in the Unique Key Columns list in the Specify Update Properties dialog box. You can also specify **'No'** or **'False'** to set this value ◆ **1** Allow the use of DateTime columns in the WHERE clause of an UPDATE or DELETE statement. When DateTimeAllowed is set to 1, DateTime columns *do display* in the Unique Key Columns list in the Specify Update Properties dialog box so you can select one or more to appear in the WHERE clause. You can also specify **'Yes'** or **'True'** to set this value

Default value

DateTimeAllowed = 0

Usage

When to use To allow the use of DateTime columns as unique key columns in the WHERE clause of an UPDATE or DELETE statement when you are updating the database from a DataWindow object, set DateTimeAllowed to 1.

In previous PowerBuilder releases, DateTime columns did not display in the Unique Key Columns list and could not be selected to appear in the WHERE clause.

FOR INFO For instructions on using the Specify Update Properties dialog box to specify update characteristics for a DataWindow object, see the section about controllling updates in the *PowerBuilder User's Guide*.

What happens when you save the DataWindow object When you set DateTimeAllowed to 1, select a DateTime column to appear in the WHERE clause, and then save the DataWindow object, this column will continue to display in the Unique Key Columns list even if you set DateTimeAllowed to 0 on a subsequent connection.

Examples

To allow the use of DateTime columns in the WHERE clause of an UPDATE or DELETE statement:

- **Database profile** Select the DateTime Datatype Allowed checkbox on the Syntax tab in the Database Profile Setup dialog box.
- **PowerBuilder application script** Type the following in a PowerBuilder script:

```
SQLCA.DBParm = "DateTimeAllowed = 1"
```

Using the examples in a PowerBuilder application script If you specify DBParm parameters in your database profile, the PowerScript 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.

DBAdm

Description

Specifies whether you have database administrator (DBAdm) authority to access every table in the IBM database to which you are connected through the IBM interface.

Applies to

IBM

Syntax

DBAdm = '*value*'

Parameter	Description
value	Specifies whether you have database administrator (DBAdm) authority to access every table in your IBM database. Values are: ♦ **Yes** You have DBAdm authority to access tables in the IBM database. If DBAdm is set to Yes, the Select Tables list in PowerBuilder displays all tables in the dataase ♦ **No** (Default) You do not have DBAdm authority to access tables in the IBM database. If DBAdm is set to No, the Select Tables list in PowerBuilder displays only those tables that the user owns, as well as those tables where SELECT permission has been explicitly granted to any of the following: the user, the user's group (as specified with the GroupID DBParm), or PUBLIC

Default value

DBAdm = 'No'

Examples

To specify that you have database administrator authority to access every table in your IBM database:

- **Database profile** Select the Database Administrator Authority checkbox on the System tab in the Database Profile Setup dialog box.
- **PowerBuilder application script** Type the following in a PowerBuilder application script:

```
SQLCA.dbParm = "DBAdm = 'Yes'"
```

Using the examples in a PowerBuilder application script If you specify DBParm parameters in your database profile, the PowerScript 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.

See also

GroupID

DBGetTime

Description

When you set the Async parameter to 1 to enable asynchronous operation, you can also set the DBGetTime parameter for those DBMSs that support it. DBGetTime specifies the number of seconds that you want PowerBuilder to wait for a response from the DBMS when you retrieve rows in a DataWindow object, query, or report.

If DBGetTime is set to 0 (the default), PowerBuilder waits indefinitely for a DBMS response (the request never times out). If the DBGetTime value expires before the first row is retrieved, your request is automatically canceled.

Applies to

MSS Microsoft SQL Server 6.x
ODBC (if driver and backend DBMS support this feature)
O72 Oracle Version 7.2
O73 Oracle Version 7.3
SYB SQL Server 4.x
SYC and SYD Sybase Systems 10.x and 11.x

Syntax

DBGetTime = *value*

Parameter	Description
value	The number of seconds you want PowerBuilder to wait for a DBMS response while waiting to retrieve the first row of a DataWindow object, query, or report (Default = 0)

Default value

DBGetTime = 0

Usage

Requirements for using DBGetTime To use the DBGetTime parameter, you must do *both* of the following:

- Set the Async parameter to 1 to enable asynchronous operation, as shown in the Examples below.
- Code a RetrieveRow event for a DataWindow object or report.

DBGetTime not supported When you use the Powersoft SYT interface to access a Sybase SQL Server database through DB-Library client software on Windows 95 or Windows NT, the DBGetTime DBParm parameter is *not supported.*

Examples

To enable asynchronous operation and set the DBGetTime parameter to 20 seconds:

- **Database profile** Select the Asynchronous checkbox and type 20 in the Number of Seconds to Wait box on the Transaction tab in the Database Profile Setup dialog box.
- **PowerBuilder application script** Type the following in a PowerBuilder application script:

```
SQLCA.dbParm = "Async = 1, DBGetTime = 20"
```

Using the examples in a PowerBuilder application script If you specify DBParm parameters in your database profile, the PowerScript 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.

See also

Async

DBTextLimit

Description

When connecting to a SQL Server database, DBTextLimit specifies the maximum length of a text field that DB-Library will return when you include the text field in a SQL SELECT statement.

You can set the DBTextLimit parameter if you want to include a long text string in a DataWindow object without treating the text as a binary large object (blob) data type.

Applies to

MSS Microsoft SQL Server 6.x
SYB and SYT SQL Server 4.x

Syntax

DBTextLimit = '*value*'

Parameter	Description
value	The maximum length in bytes of a text field that DB-Library will return when you include the text field in a SQL SELECT statement. The range of valid values is from 0 bytes to 32,763 bytes When you set DBTextLimit to 0, DB-Library returns the maximum length text field

Default value

The default value for DBTextLimit is the default specified by SQL Server for the DBTEXTLIMIT DB-Library option. (For information, see your SQL Server documentation.)

Usage

The text field length that DB-Library will return is the lesser of the DBTextLimit value and the setting for the SQL Server global variable @@textsize.

If the setting for @@textsize is less than the value you specify for DBTextLimit, DB-Library will return the @@textsize value.

Examples

To specify that DB-Library will return a text field that is up to 32,000 bytes long when you include the text field in a SQL SELECT statement:

- **Database profile** Type the following in the Text Limit box on the Syntax tab in the Database Profile Setup dialog box:

```
32000
```

- **PowerBuilder application script** Type the following in a PowerBuilder application script:

```
SQLCA.dbParm = "DBTextLimit = '32000'"
```

Using the examples in a PowerBuilder application script If you specify DBParm parameters in your database profile, the PowerScript 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.

DecimalSeparator

Description

Specifies the decimal separator setting used by the backend DBMS that you are accessing in PowerBuilder. If your DBMS uses a decimal separator other than period (.), which is the default, set DecimalSeparator to the value for your DBMS to ensure that PowerBuilder correctly handles numeric strings returned from your database.

Applies to

ODBC (if driver and backend DBMS support this feature)
Oracle (all interfaces)

Syntax

DecimalSeparator = '*value*'

Parameter	Description
value	The decimal separator setting used by the backend DBMS that you are accessing in PowerBuilder. Values are: ♦ '.' (Default) Specifies that your backend DBMS uses a period (.) as the decimal separator. If you do not specify DecimalSeparator or if you specify a value other than period (.) or comma (,), PowerBuilder uses period (.) as the decimal separator ♦ ',' Specifies that your backend DBMS uses a comma (,) as the decimal separator

Default value

DecimalSeparator = '.'

Usage

When to set DecimalSeparator The DecimalSeparator DBParm currently supports period (.) and comma (,) as valid values. Therefore, if the decimal separator setting for your DBMS is a comma, you should set the DecimalSeparator DBParm to ',' (comma) to make sure PowerBuilder correctly handles numeric strings returned from your database.

Example using Oracle Assume you are accessing an Oracle database in PowerBuilder and the decimal separator setting is a comma (,). Oracle returns to PowerBuilder the numeric string '123,50' containing a comma instead of a period as the decimal separator. PowerBuilder then sends this string to its decimal conversion routines.

By default, the PowerBuilder decimal conversion routines expect a period as the decimal separator. If you set the DecimalSeparator DBParm to ',' (comma), PowerBuilder correctly handles this string and returns it as '123,50'.

Examples

To specify that your DBMS uses a comma (,) as the decimal separator setting:

- **Database profile** Type a comma (,) in the Decimal Separator box on the Syntax tab in the Database Profile Setup dialog box.
- **PowerBuilder application script** Type the following in a PowerBuilder application script:

```
SQLCA.dbParm = "DecimalSeparator = ','"
```

Using the examples in a PowerBuilder application script If you specify DBParm parameters in your database profile, the PowerScript 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.

DelimitIdentifier

Description

Specifies whether you want PowerBuilder to enclose the names of tables, columns, indexes, and constraints in double quotes when it generates SQL statements. This affects the behavior of any PowerBuilder painter that generates SQL syntax.

When to specify DelimitIdentifier
You must specify the DelimitIdentifier DBParm parameter *before* connecting to the database in PowerBuilder.

Applies to

IBM
ODBC (if driver and backend DBMS support this feature)
Oracle (all interfaces)
MDI Sybase InformationConnect DB2 Gateway interface
NET Sybase Net-Gateway for DB2 interface
SYC and SYD Sybase Systems 10.x and 11.x

Syntax

DelimitIdentifier = '*value*'

Parameter	Description
value	Specifies whether you want PowerBuilder to enclose table and column names in double quotes. Values are: ♦ **Yes** Enclose table and column names in double quotes ♦ **No** Do not enclose table and column names in double quotes

Default value

The default value for the DelimitIdentifier parameter depends on the DBMS you are accessing, as summarized in the following table:

DBMS	DelimitIdentifer default value
IBM	`DelimitIdentifier = 'Yes'`
ODBC	Depends on the DelimitIdentifer setting in the PBODB60 initialization file
Oracle	`DelimitIdentifier = 'Yes'`
Sybase InformationConnect DB2 Gateway interface	`DelimitIdentifier = 'No'`
Sybase Net-Gateway for DB2 interface	`DelimitIdentifier = 'No'`
Sybase SQL Server System 10 and System 11	`DelimitIdentifier = 'No'`

Usage

IBM databases By default, PowerBuilder encloses IBM identifiers in double quotes to preserve case sensitivity. However, the established convention at many DB2/MVS sites is to use only uppercase identifier names. At these sites, specifying DelimitIdentifer = 'No' is recommended. The DB2 family of databases automatically converts identifiers to uppercase if they are not enclosed in double quotes.

ODBC data sources The DelimitIdentifier DBParm setting overrides the DelimitIdentifier setting specified for your ODBC driver in the PBODB60 initialization file.

Examples

To specify that PowerBuilder should not enclose table and column names in double quotes when it generates SQL statements:

- **Database profile** Clear the Enclose Table and Column Names in Quotes checkbox on the Syntax tab in the Database Profile Setup dialog box.
- **PowerBuilder application script** Type the following in a PowerBuilder application script:

```
SQLCA.dbParm = "Delimitidentifier = 'No'"
```

Using the examples in a PowerBuilder application script If you specify DBParm parameters in your database profile, the PowerScript 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.

See also

IdentifierQuoteChar

DisableBind

Description

For those DBMSs that support bind variables, PowerBuilder binds input parameters to a compiled SQL statement by default. The DisableBind parameter allows you to specify whether you want to disable this default binding.

When you set DisableBind to 1 to disable the binding, PowerBuilder replaces the input variable with the value entered by the application user or specified in a PowerBuilder script.

Applies to

IN5 and IN7 INFORMIX
ODBC (if driver and backend DBMS support this feature)
Oracle (all interfaces)

Syntax

DisableBind = *value*

Parameter	Description
value	Specifies whether you want to disable the default binding of input parameters to a compiled SQL statement. Values are: ♦ **0** (Default) PowerBuilder binds input parameters to a compiled SQL statement. This value enables the default binding ♦ **1** PowerBuilder does *not* bind input parameters to a compiled SQL statement. This value disables the default binding

Default value

DisableBind = 0

Usage

Bind variables In a SQL statement, a **bind variable** is a placeholder for a column value. By default, PowerBuilder associates (binds) data from a variable defined in your application to the bind variable each time the SQL statement executes.

Using bind variables in SQL statements For example, the following SQL statement retrieves those rows in the Books table about books written by Hemingway:

```
SELECT * FROM books WHERE author = "Hemingway"
```

Suppose that you want to execute this statement to get information about books written by other authors. Instead of compiling and executing a new statement for each author, you can define a bind variable that represents the author's name. The user then supplies the author's actual name when the application executes. By using bind variables, the statement is compiled only once and executed repeatedly with new values supplied by the user.

If your database supports bind variables and DisableBind is set to 0 (the default) to enable binding, PowerBuilder generates the statement with parameter markers (:bind_param) and passes the actual parameter value at execution time. For example:

```
SELECT * FROM books WHERE author = :bind_param
```

Bind variables and cached statements Using bind variables in conjunction with cached statements can improve the performance of most applications. The amount of improvement depends on the application. In general, applications that perform a large amount of transaction processing benefit the most from using bind variables and cached statements.

In order to use cached statements, make sure that DisableBind is set to 0. This enables the default binding of input variables to SQL statements in PowerBuilder. (For more about using cached statements, see the description of the SQLCache parameter.)

Performance improvements For SQL Anywhere and Oracle databases, bind variables improve performance by allowing PowerBuilder to insert and modify strings that exceed 255 characters.

Bind variables and default column values When DisableBind is set to 0 to enable the default use of bind variables, the DataWindow painter does both of the following to get maximum performance improvement from using bind variables when you add rows to a DataWindow object:

- Generates a SQL INSERT statement that includes all columns (except identity and SQL Server timestamp)
- Reuses this SQL INSERT statement for each row you add to the DataWindow object

For example, if a table named Order_T contains three columns named Order_ID, Order_Date, and Customer_ID, the DataWindow painter generates the following SQL INSERT statement when DisableBind is set to 0 (default binding enabled):

```
INSERT INTO Order_T(Order_ID, Order_Date, Customer_ID)
   VALUES(:bind_param, :bind_param, :bind_param)
```

If one of these columns is null, the DataWindow painter sets a null value indicator for this column parameter and executes the statement. This behavior is important to understand if you want your backend DBMS to set a default value for any columns in your DataWindow object.

To illustrate, suppose that your application users do not enter a value for the Order_Date column because they expect the backend DBMS to set this column to a default value of TODAY. They then retrieve the row and find that a null value has been set for Order_Date instead of its default value. This happens because the SQL INSERT statement generated by the DataWindowpainter specified a null value indicator, so the DBMS set the column value to null instead of to its default value as expected.

Setting a default column value when binding is enabled If you are using bind variables (DisableBind set to 0) and want the backend DBMS to set a column to its default value when your application user does not explicitly enter a value in a new row, you should set an initial value for the DataWindow object column that mirrors the DBMS default value for this column'

In the DataWindow painter, you can set or modify a column's initial value in the Column Specifications dialog box.

FOR INFO For more about the Column Specifications dialog box, see the *PowerBuilder User's Guide*.)

Setting a default column value when binding is disabled If you are *not* using bind variables (DisableBind set to 1) and want the backend DBMS to set a column to its default value when your application user does not explicitly enter a value in a new row, you do *not* need to set an initial value for the DataWindow column.

This is because with bind variables disabled, the DataWindow painter generates a SQL INSERT statement for each row added to the DataWindow. If a column does not contain an explicit value, it is not included in the SQL INSERT statement.

Using the Order_T table example, if your application user enters 123 as the value for the Order_ID column and A-123 as the value for the Customer_ID column, the DataWindow painter generates the following SQL INSERT statement when DisableBind is set to 1 (default binding disabled):

```
INSERT INTO Order_T(Order_ID, Customer_ID)
   VALUES(123, 'A-123')
```

Your backend DBMS would then set the Order_Date column to its default value as expected since a value for Order_Date is not explicitly set in the SQL INSERT statement generated by the DataWindow painter.

Examples

To specify that PowerBuilder should disable the default binding of input parameters to a compiled SQL statement:

- **Database profile** Select the Disable Bind checkbox on the Transaction tab in the Database Profile Setup dialog box.
- **PowerBuilder application script** Type the following in a PowerBuilder application script:

```
SQLCA.dbParm = "DisableBind = 1"
```

Using the examples in a PowerBuilder application script If you specify DBParm parameters in your database profile, the PowerScript 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.

See also

SQLCache

DS_Alias

Description

When you access a Sybase Systems 10.x or 11.x database in PowerBuilder through Open Client 11.1 software, DS_Alias is one of several DBParm parameters that you can set to enable network-based directory services in your application. (For other directory services DBParms, see the See Also section.)

Some directory service providers and drivers support the creation of alias entries. An **alias entry** provides a link to a primary directory entry in a hierarchy, thereby giving users multiple ways to access the primary entry while searching the directory structure for a particular network entity.

For those directory service providers and drivers that support aliases, DS_Alias specifies whether the provider is allowed to follow links for (expand) alias entries while searching the directory hierarchy. The default behavior is to allow expansion of alias entries for providers that support this feature.

You must specify a value for DS_Alias *before* connecting to the database in PowerBuilder.

Using third-party directory service providers

FOR INFO For information about the third-party directory service providers and operating system platforms that Powersoft has tested with Open Client 11.1 directory services, see the Release Notes.

Applies to

SYC and SYD Sybase Systems 10.x and 11.x

Syntax

DS_Alias = *value*

Parameter	Description
value	For those directory services providers and drivers that support aliases, specifies whether the provider is allowed to expand alias entries while searching a directory hierarchy. Values are: ♦ **0** Prohibit provider from expanding alias entries during a directory search. You can also specify **'No'** or **'False'** to set this value ♦ **1** (Default) Allow provider to expand alias entries during a directory search. You can also specify **'Yes'** or **'True'** to set this value

Default value

DS_Alias = 1

Usage

When to use To prevent access to your data through directory alias entries, set DS_Alias to 0. This prohibits directory service providers that support aliases from expanding alias entries during a directory search.

Set Release DBParm to '11' For DS_Alias to take effect, you *must* also set the Release DBParm parameter to '11' to specify that your application should use Sybase Open Client Client-Library (CT-Lib) 11.x behavior. (The default for Release is to use CT-Lib 10.x behavior, which does not support directory services.)

Requirements for use To use DS_Alias or any other DBParm parameter supporting Open Client 11.1 directory services, you must meet certain requirements for using directory services in your PowerBuilder application.

FOR INFO For details, see "Requirements for using Open Client directory services" on page 266.

Corresponding CT-Lib connection property Specifying a value for DS_Alias sets the corresponding Sybase CT-Lib 11.1 connection property named CS_DS_EXPANDALIAS.

Examples

To prohibit directory service providers that support aliases from expanding alias entries during a directory search:

- **Database profile** Clear the Directory Alias Entries checkbox on the Directory Services tab in the Database Profile Setup dialog box.
- **PowerBuilder application script** Type the following in a PowerBuilder script:

```
SQLCA.DBParm = "DS_Alias = 0"
```

Using the examples in a PowerBuilder application script If you specify DBParm parameters in your database profile, the PowerScript 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.

See also

DS_Copy
DS_DitBase
DS_Failover
DS_Principal
DS_Provider
DS_TimeLimit
Release (Sybase System 10 and System 11)

DS_Copy

Description

When you access a Sybase SQL Server 10.x or 11.x database in PowerBuilder through Open Client 11.1 software, DS_Copy is one of several DBParm parameters that you can set to enable network-based directory services in your application. (For other directory services DBParms, see the See Also section.)

Some directory service providers and drivers support the use of caching. **Caching** allows a directory service provider to use cached information while searching a directory instead of making a request to the directory server agent for information.

For those directory service providers and drives that support caching, DS_Copy specifies whether the provider is allowed to use cached information during a directory search. The default behavior is to allow providers that support this feature to use cached information.

You must specify a value for DS_Copy *before* connecting to the database in PowerBuilder.

Using third-party directory service providers

FOR INFO For information about the third-party directory service providers and operating system platforms that Powersoft has tested with Open Client 11.1 directory services, see the Release Notes.

Applies to

SYC and SYD Sybase Systems 10.x and 11.x

Syntax

DS_Copy = *value*

Parameter	Description
value	For those directory services providers and drivers that support caching, specifies whether the provider is allowed to use cached information when making a directory search. Values are: ♦ **0** Prohibit provider from using cached information during a directory search. You can also specify **'No'** or **'False'** to set this value ♦ **1** (Default) Allow provider to use cached information when making a directory search. You can also specify **'Yes'** or **'True'** to set this value

Default value

DS_Copy = 1

Usage

When to use Allowing providers to use cached information during directory searches makes the searches faster, but does not ensure that the provider is using the most up-to-date directory information.

To ensure that the application gets the most recent changes to directory entries when it requests directory information, set DS_Copy to 0 to prohibit providers that support caching from using cached information during a directory search.

Set Release DBParm to '11' For DS_Copy to take effect, you *must* also set the Release DBParm parameter to '11' to specify that your application should use Sybase Open Client Client-Library (CT-Lib) 11.x behavior. (The default for Release is to use CT-Lib 10.x behavior, which does not support directory services.)

Requirements for use To use DS_Copy or any other DBParm parameter supporting Open Client 11.1 directory services, you must meet certain requirements for using directory services in your PowerBuilder application.

FOR INFO For details, see "Requirements for using Open Client directory services" on page 266.

Corresponding CT-Lib connection property Specifying a value for DS_Copy sets the corresponding Sybase CT-Lib 11.1 connection property named CS_DS_COPY.

Examples

To prohibit directory service providers that support caching from using cached information during a directory search:

- **Database profile** Clear the Use Caching checkbox on the Directory Services tab in the Database Profile Setup dialog box.
- **PowerBuilder application script** Type the following in a PowerBuilder script:

```
SQLCA.DBParm = "DS_Copy = 0"
```

Using the examples in a PowerBuilder application script If you specify DBParm parameters in your database profile, the PowerScript 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.

See also

DS_Alias
DS_DitBase
DS_Failover
DS_Principal
DS_Provider
DS_TimeLimit
Release (Sybase System 10 and System 11)

DS_DitBase

Description

When you access a Sybase SQL Server 10.x or 11.x database in PowerBuilder through Open Client 11.1 software, DS_DitBase is one of several DBParm parameters that you can set to enable network-based directory services in your application. (For other directory services DBParms, see the See Also section.)

When you use Open Client 11.1 directory services, a default (active) directory information tree base (DIT base) is specified in the Open Client/Open Server Configuration utility. The **DIT base** is the directory node where directory searches start. This is analogous to the current working directory in UNIX or MS-DOS file systems.

DS_DitBase lets you specify the name of the directory node where you want searches for directory entries to start. The DS_DitBase value you specify must be a fully qualified name that uses the syntax required by your directory service provider and driver (see the Examples section for illustrations).

The default value for DS_DitBase is the DIT base currently specified as active in the Open Client/Open Server Configuration utility.

You must specify a value for DS_DitBase *before* connecting to the database in PowerBuilder.

Using third-party directory service providers

FOR INFO For information about the third-party directory service providers and operating system platforms that Powersoft has tested with Open Client 11.1 directory services, see the Release Notes.

Applies to

SYC and SYD Sybase Systems 10.x and 11.x

Syntax

DS_DitBase = '*dit_base*'

Parameter	Description
dit_base	The name of the directory node where you want directory searches to start. By default, this is the DIT base currently specified as active in the Open Client/Open Server Configuration utility The value for *dit_base* must be a fully qualified name that uses the syntax required by your directory service provider and driver. The syntax for specifying the DIT base varies for different providers; see your provider's documentation for details FOR INFO For examples of how to specify *dit_base* for different directory service providers, see the Examples section

Default value

The default value for DS_DitBase is the DIT base currently specified as active in the Open Client/Open Server Configuration utility.

Usage

When to use Set DS_DitBase to specify a starting node for directory searches *other than* the DIT base node specified as active in the Open Client/Open Server Configuration utility.

FOR INFO For instructions on using the Open Client/Open Server Configuration utility, see your Sybase Open Client/Server configuration guide.

Set Release DBParm to '11' For DS_DitBase to take effect, you *must* also set the Release DBParm parameter to '11' to specify that your application should use Sybase Open Client Client-Library (CT-Lib) 11.x behavior. (The default for Release is to use CT-Lib 10.x behavior, which does not support directory services.)

Requirements for use To use DS_DitBase or any other DBParm parameter supporting Open Client 11.1 directory services, you must meet certain requirements for using directory services in your PowerBuilder application.

FOR INFO For details, see "Requirements for using Open Client directory services" on page 266.

Corresponding CT-Lib connection property Specifying a value for DS_DitBase sets the corresponding Sybase CT-Lib 11.1 connection property named CS_DS_DITBASE.

Examples

About these examples The examples that follow show how to specify a DS_DitBase value for different directory service providers.

FOR INFO See your directory service provider's documentation for complete information about the format your provider requires for specifying the DIT base.

Example 1 (Windows NT Registry) This example shows the syntax for DS_DitBase if your directory service provider is the Windows NT Registry.

```
Node name:SALES:software\sybase\server\SYS11NT
DS_DitBase:SALES:software\sybase\server
```

To set DS_DitBase:

- **Database profile** Type the following in the DIT Base box on the Directory Services tab in the Database Profile Setup dialog box. Do *not* end the DS_DitBase value with a backslash (\).

```
SALES:software\sybase\server
```

- **PowerBuilder application script** Type the following in a PowerBuilder script. Do *not* end the DS_DitBase value with a backslash (\).

```
SQLCA.DBParm =
   "DS_DitBase='SALES:software\sybase\server'"
```

Example 2 (DCE/CDS) This example shows the syntax for DS_DitBase if your directory service provider is Distributed Computing Environment Cell Directory Services (DCE/CDS).

Node name:/.../boston.sales/dataservers/sybase/SYS11
DS_DitBase:/.../boston.sales/dataservers

To set DS_DitBase:

- **Database profile** Type the following in the DIT Base box on the Directory Services tab in the Database Profile Setup dialog box. Do *not* end the DS_DitBase value with a slash (/).

```
/.../boston.sales/dataservers
```

- **PowerBuilder application script** Type the following in a PowerBuilder script. Do *not* end the DS_DitBase value with a slash (/).

```
SQLCA.DBParm =
   "DS_DitBase = '/.../boston.sales/dataservers'"
```

Example 3 (Banyan STDA) This example shows the syntax for DS_DitBase if your directory service provider is Banyan StreetTalk Directory Assistance (STDA).

Node name:SYS11@sales@chicago
DS_DitBase:chicago

To set DS_DitBase:

- **Database profile** Type the following in the DIT Base box on the Directory Services tab in the Database Profile Setup dialog box. Do *not* start the DS_DitBase value with @.

```
chicago
```

- **PowerBuilder application script** Type the following. Do *not* start the DS_DitBase value with @.

```
SQLCA.DBParm = "DS_DitBase = 'chicago'"
```

Example 4 (Novell NDS) This example shows the syntax for DS_DitBase if your directory service provider is Novell NetWare Directory Services (NDS).

Node name:CN=SYS11.OU=miami.OU=sales.O=sybase
DS_DitBase:OU=miami.OU=sales.O=sybase

To set DS_DitBase:

- **Database profile** Type the following in the DIT Base box on the Directory Services tab in the Database Profile Setup dialog box:

```
OU=miami.OU=sales.O=sybase
```

- **PowerBuilder application script** To specify DS_DitBase in a PowerBuilder application script, type the following:

```
SQLCA.DBParm =
   "DS_DitBase = 'OU=miami.OU=sales.O=sybase'"
```

Using the examples in a PowerBuilder application script If you specify DBParm parameters in your database profile, the PowerScript 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.

See also

DS_Alias
DS_Copy
DS_Failover
DS_Principal
DS_Provider
DS_TimeLimit
Release (Sybase System 10 and System 11)

DS_Failover

Description

When you access a Sybase Systems 10.x or 11.x database in PowerBuilder through Open Client 11.1 software, DS_Failover is one of several DBParm parameters that you can set to enable network-based directory services in your application. (For other directory services DBParms, see the See Also section.)

Sybase Open Client Client-Library (CT-Lib) requires a directory to map logical server names to network addresses. The source for this directory can either be the Sybase Interfaces file or a network-based directory service provider (such as DCE Cell Directory Services or the Windows NT Registry).

If you want an application to use a directory source *other than* the Interfaces file, CT-Lib must be able to load the appropriate directory driver. If CT-Lib cannot load the required driver, you can set DS_Failover to specify whether CT-Lib should silently default (fail over) to using the Interfaces file as the directory source.

By default, DS_Failover specifies that CT-Lib should use the Interfaces file as the directory source if it cannot load the requested directory driver.

You must specify a value for DS_Failover *before* connecting to the database in PowerBuilder.

Using third-party directory service providers

FOR INFO For information about the third-party directory service providers and operating system platforms that Powersoft has tested with Open Client 11.1 directory services, see the Release Notes.

Applies to

SYC and SYD Sybase Systems 10.x and 11.x

Syntax

DS_Failover = *value*

Parameter	Description
value	Specifies whether Sybase CT-Lib should silently default (fail over) to using the Interfaces file as the directory source if it cannot load the requested directory driver. Values are: ♦ **0** Prohibit CT-Lib from using the Interfaces file as the directory source if it cannot load the requested directory driver. You can also specify **'No'** or **'False'** to set this value ♦ **1** (Default) Allow CT-Lib to use the Interfaces file as the directory source if it cannot load the requested directory driver. You can also specify **'Yes'** or **'True'** to set this value

Default value

DS_Failover = 1

Usage

When to use To prevent CT-Lib from using the Interfaces file as the directory source if it cannot load the requested directory driver, set DS_Failover to 0.

If DS_Failover is set to 0 to prevent use of the Interfaces file and CT-Lib cannot load the requested directory driver, the connection's directory source is undefined. This will cause certain operations requiring directory access to fail.

Set Release DBParm to '11' For DS_Failover to take effect, you *must* also set the Release DBParm parameter to '11' to specify that your application should use Sybase CT-Lib 11.x behavior. (The default for Release is to use CT-Lib 10.x behavior, which does not support directory services.)

Requirements for use To use DS_Failover or any other DBParm parameter supporting Open Client 11.1 directory services, you must meet certain requirements for using directory services in your PowerBuilder application.

FOR INFO For details, see "Requirements for using Open Client directory services" on page 266.

Corresponding CT-Lib connection property Specifying a value for DS_Failover sets the corresponding Sybase CT-Lib 11.1 connection property named CS_DS_FAILOVER.

Examples

To prohibit CT-Lib from using the Interfaces file as the directory source if it cannot load the requested directory driver:

- **Database profile** Clear the Enable Failover checkbox on the Directory Services tab in the Database Profile Setup dialog box.
- **PowerBuilder application script** Type the following in a PowerBuilder script:

```
SQLCA.DBParm = "DS_Failover = 0"
```

Using the examples in a PowerBuilder application script If you specify DBParm parameters in your database profile, the PowerScript 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.

See also

DS_Alias
DS_Copy
DS_DitBase
DS_Principal
DS_Provider
DS_TimeLimit
Release (Sybase System 10 and System 11)

DS_Principal

Description

When you access a Sybase Systems 10.x or 11.x database in PowerBuilder through Open Client 11.1 software, DS_Principal is one of several DBParm parameters that you can set to enable network-based directory services in your application. (For other directory services DBParms, see the See Also section.)

Some directory service providers and drivers require an authenticated principal (user ID) name to control an application's access to directory entries. For those providers and drivers, DS_Principal specifies the principal name your application should use to identify you to the directory service provider.

You must specify a value for DS_Principal *before* connecting to the database in PowerBuilder.

Using third-party directory service providers

FOR INFO For information about the third-party directory service providers and operating system platforms that Powersoft has tested with Open Client 11.1 directory services, see the Release Notes.

Applies to

SYC and SYD Sybase Systems 10.x and 11.x

Syntax

DS_Principal = '*principal_name*'

Parameter	Description
principal_name	The principal (user ID) name your application should use to identify you to the directory service provider

Default value

None

PowerBuilder does not set DS_Principal or the corresponding Sybase Open Client Client-Library (CT-Lib) 11.1 connection parameter CS_DS_PRINCIPAL if you do not specify a value.

Usage

When to use If your directory service provider requires an authenticated principal name for directory access, set DS_Principal to the principal (user ID) name that goes with your directory service password.

Set Release DBParm to '11' For DS_Principal to take effect, you *must* also set the Release DBParm parameter to '11' to specify that your application should use Sybase CT-Lib 11.x behavior. (The default for Release is to use CT-Lib 10.x behavior, which does not support directory services.)

Requirements for use To use DS_Principal or any other DBParm parameter supporting Open Client 11.1 directory services, you must meet certain requirements for using directory services in your PowerBuilder application.

FOR INFO For details, see "Requirements for using Open Client directory services" on page 266.

Corresponding CT-Lib connection property Specifying a value for DS_Principal sets the corresponding Sybase CT-Lib 11.1 connection property named CS_DS_PRINCIPAL.

Examples

To specify JSMITH as your application's principal name:

- **Database profile** Type the following in the Principal Name box on the Directory Services tab in the Database Profile Setup dialog box:

```
JSMITH
```

- **PowerBuilder application script** Type the following in a PowerBuilder script:

```
SQLCA.DBParm = "DS_Principal = 'JSMITH'"
```

Using the examples in a PowerBuilder application script If you specify DBParm parameters in your database profile, the PowerScript 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.

See also

DS_Alias
DS_Copy
DS_DitBase
DS_Failover
DS_Provider
DS_TimeLimit
Release (Sybase System 10 and System 11)

DS_Provider

Description

When you access a Sybase SQL Server 10.x or 11.x database in PowerBuilder through Open Client 11.1 software, DS_Provider is one of several DBParm parameters that you can set to enable network-based directory services in your application. (For other directory services DBParms, see the See Also section.)

When you use Open Client 11.1 directory services, you must specify your directory service provider names in the Open Client/Open Server Configuration utility so the required drivers can be loaded for each provider. The default directory service provider is the one currently specified as active in the Configuration utility.

DS_Provider lets you specify a directory service provider name listed in the Open Client/Open Server Configuration utility *other than* the default (active) provider. The default value for DS_Provider is the provider name currently specified as active in the Configuration utility.

You must specify a value for DS_Provider *before* connecting to the database in PowerBuilder.

Using third-party directory service providers

FOR INFO For information about the third-party directory service providers and operating system platforms that Powersoft has tested with Open Client 11.1 directory services, see the Release Notes.

Applies to

SYC and SYD Sybase Systems 10.x and 11.x

Syntax

DS_Provider = '*provider_name*'

Parameter	Description
provider_name	The directory service provider name you want to use for directory services The provider name is case-sensitive. You must specify it *exactly as it appears* in the Open Client/Open Server Configuration utility

Default value

The default value for DS_Provider is the provider name currently specified as active in the Open Client/Open Server Configuration utility.

Usage

When to use Set DS_Provider to use a directory service provider specified in the Open Client/Open Server Configuration utility *other than* the default (active) provider.

FOR INFO For instructions on using the Open Client/Open Server Configuration utility, see your Sybase Open Client/Server configuration guide.

Set Release DBParm to '11' For DS_Provider to take effect, you *must* also set the Release DBParm parameter to '11' to specify that your application should use Sybase Open Client Client-Library (CT-Lib) 11.x behavior. (The default for Release is to use CT-Lib 10.x behavior, which does not support directory services.)

Requirements for use To use DS_Provider or any other DBParm parameter supporting Open Client 11.1 directory services, you must meet certain requirements for using directory services in your PowerBuilder application.

FOR INFO For details, see "Requirements for using Open Client directory services" on page 266.

Corresponding CT-Lib connection property Specifying a value for DS_Provider sets the corresponding Sybase CT-Lib 11.1 connection property named CS_DS_PROVIDER.

Examples

To specify NTREGISTRY as the directory service provider name:

- **Database profile** Type the following in the Provider box on the Directory Services tab in the Database Profile Setup dialog box:

```
NTREGISTRY
```

- **PowerBuilder application script** Type the following in a PowerBuilder script:

```
SQLCA.DBParm = "DS_Provider = 'NTREGISTRY'"
```

Using the examples in a PowerBuilder application script If you specify DBParm parameters in your database profile, the PowerScript 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.

See also

DS_Alias
DS_Copy
DS_DitBase
DS_Failover
DS_Principal
DS_TimeLimit
Release (Sybase System 10 and System 11)

DS_TimeLimit

Description

When you access a Sybase Systems 10.x or 11.x database in PowerBuilder through Open Client 11.1 software, DS_TimeLimit is one of several DBParm parameters that you can set to enable network-based directory services in your application. (For other directory services DBParms, see the See Also section.)

Some directory service providers and drivers support the use of time limits for a directory search. For those providers and drivers, DS_TimeLimit specifies the maximum number of seconds that a directory search will last.

By default, DS_TimeLimit specifies that there is no time limit for a directory search.

You must specify a value for DS_TimeLimit *before* connecting to the database in PowerBuilder.

Using third-party directory service providers

FOR INFO For information about the third-party directory service providers and operating system platforms that Powersoft has tested with Open Client 11.1 directory services, see the Release Notes.

Applies to

SYC and SYD Sybase Systems 10.x and 11.x

Syntax

DS_TimeLimit = *value*

Parameter	Description
value	Specifies the maximum number of seconds that you want a directory search to last. You can also specify **'no_limit'** (the default) to indicate that there is no time limit for the directory search If the specified time limit expires, the directory search is unsuccessful and the PowerBuilder connection fails

Default value

DS_TimeLimit = 'no_limit'

Usage

Set Release DBParm to '11' For DS_TimeLimit to take effect, you *must* also set the Release DBParm parameter to '11' to specify that your application should use Sybase Open Client Client-Library (CT-Lib) 11.x behavior. (The default for Release is to use CT-Lib 10.x behavior, which does not support directory services.)

Requirements for use To use DS_TimeLimit or any other DBParm parameter supporting Open Client 11.1 directory services, you must meet certain requirements for using directory services in your PowerBuilder application.

FOR INFO For details, see "Requirements for using Open Client directory services" on page 266.

Corresponding CT-Lib connection property Specifying a value for DS_TimeLimit sets the corresponding Sybase CT-Lib 11.1 connection property named CS_DS_TIMELIMIT.

Examples

To specify that you want the directory search to last a maximum of 120 seconds (2 minutes):

- **Database profile** Type the following in the Directory Search Time Limit box on the Directory Services tab in the Database Profile Setup dialog box:

```
120
```

- **PowerBuilder application script** Type the following in a PowerBuilder script:

```
SQLCA.DBParm = "DS_TimeLimit = 120"
```

Using the examples in a PowerBuilder application script If you specify DBParm parameters in your database profile, the PowerScript 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.

See also

DS_Alias
DS_Copy
DS_DitBase
DS_Failover
DS_Principal
DS_Provider
Release (Sybase System 10 and System 11)

FormatArgsAsExp

Description

For those database interfaces that support it, FormatArgsAsExp controls whether PowerBuilder converts a DataWindow or report retrieval argument of decimal data type to scientific (exponential) notation if the argument exceeds 12 digits. If FormatArgsAsExp is set to Yes (the default), PowerBuilder performs this conversion.

When to specify FormatArgsAsExp
You must specify a value for FormatArgsAsExp *before* connecting to the database in PowerBuilder.

Applies to

IBM
ODBC interface (if driver and backend DBMS support this feature)
Oracle (all interfaces)
MDI Sybase InformationConnect DB2 Gateway interface
NET Sybase Net-Gateway for DB2 interface

Syntax

FormatArgsAsExp= '*value*'

Parameter	Description
value	Specifies whether you want PowerBuilder to convert a DataWindow or report retrieval argument of decimal data type to scientific (exponential) notation if the argument exceeds 12 digits. Values are: ♦ **Yes** (Default) PowerBuilder converts a retrieval argument of decimal data type to scientific notation if it exceeds 12 digits ♦ **No** PowerBuilder leaves the retrieval argument as a decimal and does not perform the default conversion to scientific notation if it exceeds 12 digits

Default value

FormatArgsAsExp = 'Yes'

Usage

When to use The setting of FormatArgsAsExp may affect the speed of data retrieval in your DataWindows and reports, especially if you are accessing large databases.

If FormatArgsAsExp is set to Yes (the default), PowerBuilder converts retrieval arguments of type decimal to scientific notation if the argument exceeds 12 digits. Some DBMS optimizers may interpret the resulting scientific notation as a different data type and scan all rows in the table to find it. This can slow data retrieval if (for example) you are accessing a DB2 database with many large tables.

Setting FormatArgsAsExp to No tells PowerBuilder to leave the retrieval argument as a decimal and not convert it to scientific notation. This speeds data retrieval for large databases.

Data type precision The precision for PowerScript decimal data types is 18 digits, while the precision for scientific notation is 32 digits.

Thus, if a retrieval argument of type decimal exceeds 18 digits and FormatArgsAsExp is set to No to prevent conversion to scientific notation, PowerBuilder truncates the retrieval argument value after 18 digits.

Examples

To tell PowerBuilder to leave a retrieval argument exceeding 12 digits as a decimal and not convert it to scientific notation:

- **Database profile** Clear the Format Arguments in Scientific Notation checkbox on the Syntax tab in the Database Profile Setup dialog box.
- **PowerBuilder application script** Type the following in a PowerBuilder script:

```
SQLCA.DBParm = "FormatArgsAsExp = 'No'"
```

Using the examples in a PowerBuilder application script If you specify DBParm parameters in your database profile, the PowerScript 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.

GroupID

Description

At some sites, administrators grant permission to access database tables to groups of users instead of or in addition to individual users. The GroupID DBParm allows you to specify an additional group authorization ID that gives a group of users permission to access tables in the database.

If you specify a group ID, tables accessible to this group as well as to individuals display in the Select Tables lists in PowerBuilder.

Applies to

IBM
MDI Sybase InformationConnect DB2 Gateway interface

Syntax

GroupID = '*group_authorization_ID*'

Parameter	Description
group_authorization_ID	The ID that grants authority to a group of users to access tables in your database

Default value

None

Examples

Example 1 This example sets the GroupID to PB1. Tables accessible to this group will display in the Select Tables list in PowerBuilder.

- **Database profile** Type the following in the Group Authorization ID box on the System tab in the Database Profile Setup dialog box:

```
PB1
```

- **PowerBuilder application script** Type the following in a PowerBuilder application script:

```
SQLCA.dbParm = "GroupID = 'PB1'"
```

Example 2 You can set the GroupID, PBCatalogOwner, and SystemOwner parameters together. To set the group ID to PB3, PowerBuilder catalog owner to POWERBLD, and system owner to PBOWNER:

- **Database profile** Type PB3 in the Group Authorization ID box, POWERBLD in the PowerBuilder Catalog Table Owner box, and PBOWNER in the System Owner box on the System tab in the Database Profile Setup dialog box.
- **PowerBuilder application script** Type the following on a single line in a PowerBuilder application script:

```
SQLCA.dbParm = "GroupID = 'PB3',
PBCatalogOwner ='POWERBLD',SystemOwner = 'PBOWNER'"
```

Using the examples in a PowerBuilder application script If you specify DBParm parameters in your database profile, the PowerScript 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.

See also

PBCatalogOwner
SystemOwner

Host

Description

If your DBMS supports it, specifies the workstation name when connecting to the database in PowerBuilder. The Host DBParm parameter lets you assign any 10-character label to identify the process you are about to create when you connect to the database. This label helps you distinguish your process from others running on the database server.

When to specify Host
You must specify the Host DBParm parameter *before* connecting to the database in PowerBuilder.

Applies to

MSS Microsoft SQL Server 6.x
SYB and SYT SQL Server 4.x
MDI Sybase InformationConnect DB2 Gateway interface
NET Sybase Net-Gateway for DB2 interface
SYC and SYD Sybase Systems 10.x and 11.x

Syntax

Host = '*workstation_name*'

Default value

None

Usage

SQL Server databases When you specify a value for Host, PowerBuilder sets the SQL Server CS_HOSTNAME connection property to the workstation name you specify.

The value you specify for the Host parameter displays in the hostname column of the MASTER.DBO.SYSPROCESSES table in a SQL Server database. How you use the Host parameter depends on the design of your PowerBuilder application.

For example, many sites want to secure their production tables so that updates are possible only through a specific application. To do this, you can grant explicit authority to the PowerBuilder application but *not* to users.

The application prompts the user for an authorization ID and password, verifies it, and then connects to the SQL Server database through a single application login ID. Only this application login ID has authorization to update production tables.

In this scenario, you could use the Host DBParm parameter to store the name of the user running the application.

Examples

Example 1 To set the host name to Alan:

- **Database profile** Type the following in the Workstation Name box on the Network tab in the Database Profile Setup dialog box:

```
Alan
```

- **PowerBuilder application script** Type the following in a PowerBuilder application script:

```
SQLCA.dbParm = "Host = 'Alan'"
```

Example 2 You can use the Host and AppName parameters together to specify both the host name and the application name. To set the host name to Jane and the application name to Sales:

- **Database profile** Type Jane in the Workstation Name box and Sales in the Application Name box on the Network tab in the Database Profile Setup dialog box.
- **PowerBuilder application script** Type the following in a PowerBuilder application script:

```
SQLCA.dbParm = "Host = 'Jane',AppName = 'Sales'"
```

Using the examples in a PowerBuilder application script If you specify DBParm parameters in your database profile, the PowerScript 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.

See also

AppName

IdentifierQuoteChar

Description

Specifies the single **quote character** you want PowerBuilder to use to delimit the names of identifiers (tables, columns, indexes, and constraints) when it generates SQL statements. PowerBuilder uses the quote character you specify instead of the default quote character returned by your ODBC driver.

DelimitIdentifier must be set to Yes
In order for IdentifierQuoteChar to take effect, the DelimitIdentifier parameter must be set to Yes in either your database profile or the PBODB60 initialization file. Otherwise, PowerBuilder's default behavior is *not* to delimit identifiers in SQL statements and to ignore any value specified for IdentifierQuoteChar.

Applies to

ODBC (if driver and backend DBMS support this feature)

Syntax

IdentifierQuoteChar = '*quote_character*'

Parameter	Description
quote_character	The single character you want PowerBuilder to use instead of your ODBC driver's default quote character to delimit the names of identifiers in SQL statements

Default value

None

PowerBuilder searches the following in this order to determine the IdentifierQuoteChar value:

1 The section for your database profile in the PowerBuilder initialization file (in the development environment) or the value of the transaction object DBParm property (in a PowerBuilder application)

2 The section for your ODBC driver in the PBODB60 initialization file

If PowerBuilder does not find an IdentifierQuoteChar value in these locations, it makes a SQLGetInfo call to your ODBC driver to return the default SQL_IDENTIFIER_QUOTE_CHAR value.

Usage

By default, some ODBC drivers return quote characters that do not work with PowerBuilder's parsing routines, such as the backquote character (`). As a result, delimiting is turned off for these drivers in PowerBuilder.

However, if you paint SQL statements containing identifiers that require delimiters, syntax errors can occur if you are using an ODBC driver for which delimiting is turned off. To avoid such errors, set IdentifierQuoteChar to override the driver's default quote character.

Examples

To specify *c* as the quote character you want PowerBuilder to use to delimit identifiers in SQL statements:

- **Database profile** Type the following in the Identifier Quote Character box on the Syntax tab in the Database Profile Setup dialog box:

```
c
```

- **PowerBuilder application script** Type the following in a PowerBuilder application script:

```
SQLCA.dbParm = "IdentifierQuoteChar = 'c'"
```

Using the examples in a PowerBuilder application script If you specify DBParm parameters in your database profile, the PowerScript 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.

See also

DelimitIdentifier

INET_DBPATH

Description

Specifies the INFORMIX DBPATH setting. The DBPATH environment variable identifies a list of directories that contain INFORMIX databases. INET_DBPATH typically specifies the location of INFORMIX databases if this is *other* than in a directory on the database server.

INET_DBPATH not applicable on UNIX
INET_DBPATH is *not applicable* when accessing an INFORMIX database through the INFORMIX ODBC driver supplied in UNIX versions of PowerBuilder.

Applies to

IN5 and IN7 INFORMIX

Syntax

INET_DBPATH = '*server_db_path*'

Parameter	Description
server_db_path	The name of the directory containing INFORMIX databases

Default value

By default, PowerBuilder uses the value specified for DBPATH in the INFORMIX.INI configuration file.

Examples

Example 1 To specify that the directory /HOME/INFORMIX contains INFORMIX databases:

- **Database profile** Type the following in the Database Path box on the Network tab in the Database Profile Setup dialog box:

```
/home/informix
```

- **PowerBuilder application script** Type the following in a PowerBuilder application script:

```
SQLCA.dbParm = "INET_DBPATH = '/home/informix'"
```

Example 2 You can specify values for INET_DBPATH, INET_PROTOCOL, and INET_SERVICE together. To specify that the directory /INFORMIX contains INFORMIX databases and that you want to connect using the SE5 service and the TCP/IP network protocol:

- **Database profile** Type /informix in the Database Path box, SE5 in the Service Name box, and tcp-ip in the Protocol Type box on the Network tab in the Database Profile Setup dialog box.
- **PowerBuilder application script** Type the following on a single line in a PowerBuilder application script:

```
SQLCA.dbParm = "INET_DBPATH = '/informix',
   INET_SERVICE = 'se5',INET_PROTOCOL = 'tcp-ip'"
```

Using the examples in a PowerBuilder application script If you specify DBParm parameters in your database profile, the PowerScript 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.

See also

INET_PROTOCOL
INET_SERVICE

INET_PROTOCOL

Description

Specifies the network protocol that the INFORMIX client software uses to communicate with a remote INFORMIX Version 5.x, 6.x, or 7.x database server.

INET_PROTOCOL not applicable on UNIX
INET_PROTOCOL is *not applicable* when accessing an INFORMIX database through the INFORMIX ODBC driver supplied in UNIX versions of PowerBuilder.

Applies to

IN5 and IN7 INFORMIX

Syntax

INET_PROTOCOL = '*network_protocol*'

Parameter	Description
network_protocol	A string that specifies the name of the network protocol used by the INFORMIX client software FOR INFO For information about the correct network protocol for your site, see your INFORMIX system administrator

Default value

By default, PowerBuilder uses the network protocol specified in the INFORMIX.INI configuration file.

Examples

Example 1 To specify that INFORMIX client software uses the Novell IPX/SPX network protocol:

- **Database profile** Type the following in the Protocol Type box on the Network tab in the Database Profile Setup dialog box:

```
ipx
```

- **PowerBuilder application script** Type the following in a PowerBuilder application script:

```
SQLCA.dbParm = "INET_PROTOCOL = 'ipx'"
```

Example 2 You can specify values for INET_DBPATH, INET_PROTOCOL, and INET_SERVICE together. To specify that the directory /INFORMIX contains INFORMIX databases, and that you want to connect using the SE5 service and the TCP/IP network protocol:

- **Database profile** Type /informix in the Database Path box, SE5 in the Service Name box, and tcp-ip in the Protocol Type box on the Network tab in the Database Profile Setup dialog box.
- **PowerBuilder application script** Type the following on a single line in a PowerBuilder application script:

```
SQLCA.dbParm = "INET_DBPATH = '/informix',
   INET_SERVICE = 'se5',INET_PROTOCOL = 'tcp-ip'"
```

Using the examples in a PowerBuilder application script If you specify DBParm parameters in your database profile, the PowerScript 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.

See also

INET_DBPATH
INET_SERVICE

INET_SERVICE

Description

Specifies the name of the service that a remote INFORMIX database server uses to listen to all incoming requests from client applications.

INET_SERVICE not applicable on UNIX
INET_SERVICE is *not applicable* when accessing an INFORMIX database through the INFORMIX ODBC driver supplied in UNIX versions of PowerBuilder.

Applies to

IN5 and IN7 INFORMIX

Syntax

INET_SERVICE = '*service_name*'

Parameter	Description
service_name	A string that specifies the name of the service that a remote INFORMIX database server uses to listen to incoming requests FOR INFO For information about the correct service name for your site, see your INFORMIX system administrator

Default value

By default, PowerBuilder uses the service name specified in the INFORMIX.INI configuration file.

Examples

Example 1 To specify that your INFORMIX database server uses the sqlexec service name:

- **Database profile** Type the following in the Service Name box on the Network tab in the Database Profile Setup dialog box:

```
sqlexec
```

- **PowerBuilder application script** Type the following in a PowerBuilder application script:

```
SQLCA.dbParm = "INET_SERVICE = 'sqlexec'"
```

Example 2 You can specify values for INET_DBPATH, INET_PROTOCOL, and INET_SERVICE together. To specify that the directory /INFORMIX contains INFORMIX databases, and that you want to connect using the SE5 service and the TCP/IP network protocol:

- **Database profile** Type /informix in the Database Path box, SE5 in the Service Name box, and tcp-ip in the Protocol Type box on the Network tab in the Database Profile Setup dialog box.

- **PowerBuilder application script** Type the following on a single line in a PowerBuilder application script:

```
SQLCA.dbParm = "INET_DBPATH = '/informix',
   INET_SERVICE = 'se5',INET_PROTOCOL = 'tcp-ip'"
```

Using the examples in a PowerBuilder application script If you specify DBParm parameters in your database profile, the PowerScript 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.

See also

INET_DBPATH
INET_PROTOCOL

InsertBlock

Description

Specifies the number of rows that you want the Data Pipeline in PowerBuilder to insert at one time into a table in the destination database.

FOR INFO For instructions on using the Data Pipeline, see the *PowerBuilder User's Guide*.

Applies to

ODBC (only in Data Pipeline if driver and backend DBMS support this feature)

Syntax

InsertBlock = *insert_blocking_factor*

Parameter	Description
insert_blocking_factor	The number of rows that you want the Data Pipeline to insert at one time into a table in the destination database, up to a maximum of 100 rows (Default = 100 rows) To turn off block inserting for an ODBC data source in the Data Pipeline, set InsertBlock to 1 or DisableBind to 1 in the database profile of the destination database

Default value

InsertBlock = 100

Usage

Requirements for using InsertBlock To use the InsertBlock parameter, *all* of the following must be true:

- You are using an ODBC driver to access the destination database in the Data Pipeline.
- The destination database supports the use of bind variables. (For more about bind variables, see DisableBind on page 444.)
- The DisableBind parameter is set to 0 in the database profile of the destination database. This enables the default binding of input parameters to a compiled SQL statement in PowerBuilder.
- Maximum Errors is set to 1 in the Data Pipeline.

The SQL Anywhere ODBC driver and most INTERSOLV ODBC drivers meet the first two requirements.

FOR INFO To determine whether your ODBC driver meets these requirements, see the documentation that comes with your driver.

Determining the InsertBlock value PowerBuilder searches the following in this order to determine the value for InsertBlock:

1 The section for your database profile in the PowerBuilder initialization file

2 The section for your ODBC driver in the PBODB60 initialization file

If PowerBuilder does not find an InsertBlock value in these locations, it defaults to an insert blocking factor of 100 rows.

What happens When PowerBuilder finds a value for InsertBlock, the Data Pipeline batches the specified number of rows and inserts them with a single call to the ODBC driver you are using to access the destination database.

If you specify an InsertBlock value or Data Pipeline commit factor of less than 100 rows, the Data Pipeline batches and inserts the specified number of rows into the destination database. If you specify more than 100 rows, the Data Pipeline batches and inserts at most only 100 rows at one time.

The insert blocking factor that the Data Pipeline actually uses depends on the size of the data in each column inserted in the destination database. In addition, the Data Pipeline does not exceed 64K of data in the buffer for any one column.

Turning off block inserting To turn off block inserting for an ODBC data source in the Data Pipeline, you can do any of the following in the database profile of the destination database:

- Set the InsertBlock parameter to 1
- Set the DisableBind parameter to 1 (to disable default binding of input parameters to a compiled SQL statement)
- In the Data Pipeline, set Maximum Errors to a value other than 1

Examples

To set the insert blocking factor in the Data Pipeline to 50 rows:

- **Database profile** Type the following in the Insert Blocking Factor box on the Transaction tab in the Database Profile Setup dialog box:

```
50
```

- **PowerBuilder application script** Type the following in a PowerBuilder application script:

```
SQLCA.dbParm = "InsertBlock = 50"
```

Using the examples in a PowerBuilder application script If you specify DBParm parameters in your database profile, the PowerScript 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.

See also

DisableBind

Language

Description

Specifies the language you want to use when connecting to a Microsoft SQL Server 6.x, Sybase System 10.x, or Sybase System 11.x database in PowerBuilder.

When to specify Language
You must specify the Language DBParm parameter *before* connecting to the database in PowerBuilder. The Language setting takes effect when you connect to the database.

Applies to

MSS Microsoft SQL Server 6.x
SYC and SYD Sybase Systems 10.x and 11.x

Syntax

Language = '*language_name*'

Default value

None

Usage

Microsoft SQL Server 6.x Set the Language DBParm parameter to specify the language you want to use when displaying error messages and date formats.

In order for the Language DBParm to take effect, you or your database administrator must have already set up the specified language on the database server.

FOR INFO For information on setting up languages on the database server, see the administrator's guide in your Microsoft SQL Server 6.x documentation.

Sybase Systems 10.x and 11.x When you specify a value for Language, PowerBuilder does the following:

- Allocates a CS_LOCALE structure for this connection
- Sets the CS_SYB_LANG value to the language you specify
- Sets the SQL Server CS_LOC_PROP connection property with the new locale information

If you have previously set a value for the Locale DBParm parameter, which includes settings for the language and character set you want the Open Client software to use, you can override the language value by specifying a new value for the Language DBParm and reconnecting to the database.

Examples

To set the Language DBParm parameter to French:

- **Database profile** Type the following in the Language box on the Connection tab or Regional Settings tab in the Database Profile Setup dialog box:

```
French
```

- **PowerBuilder application script** Type the following in a PowerBuilder application script:

```
SQLCA.dbParm = "Language = 'French'"
```

Using the examples in a PowerBuilder application script If you specify DBParm parameters in your database profile, the PowerScript 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.

See also

CharSet
Locale

Locale

Description

Specifies the locale name that you want the Sybase Open Client software to use when connecting to a Sybase Systems 10.x or 11.x database in PowerBuilder.

When to specify Locale
You must specify the Locale DBParm parameter *before* connecting to the database in PowerBuilder.

Applies to

SYC and SYD Sybase Systems 10.x and 11.x

Syntax

Locale = '*locale_name*'

Default value

If you do not specify a value for the Locale DBParm parameter, Sybase Open Client uses the default locale defined in your LOCALES.DAT file.

Usage

Locales Locales are stored as entries in a file named LOCALES.DAT. The LOCALES.DAT file contains information about the languages and character sets you are using with the Sybase Open Client software. The Sybase Open Client installation places the LOCALES.DAT file in the $SYBASE\LOCALES directory.

An entry in the LOCALES.DAT file has the following format:

locale = *locale_name, language_name, character_set_name*

For example, the following are sample entries in the LOCALES.DAT file:

```
locale = default, us_english, cp850
locale = enu, us_english, cp850
locale = fra, french, cp850
```

Why set Locale DBParm Setting a value for the Locale DBParm parameter lets you use a locale *other than the default locale* when accessing a Sybase System 10 or System 11 database. If you do not set a value for Locale, Sybase Open Client uses the default locale defined in your LOCALES.DAT file.

What happens When you specify a value for the Locale DBParm parameter, PowerBuilder does the following:

- Allocates a CS_LOCALE structure for this connection
- Sets the CS_LC_ALL value to the locale name you specify
- Sets the SQL Server CS_LOC_PROP connection property with the new locale information

Overriding Locale DBParm If you have previously set a value for the Locale DBParm parameter, which includes settings for the language and character set you want to use, you can override the language or character set values by specifying new values for the Language or CharSet DBParm and reconnecting to the database.

Examples

To set the locale to *fra* in a Sybase Systems 10.x or 11.x connection:

- **Database profile** Type the following in the in the Locale box on the Regional Settings tab in the Database Profile Setup dialog box:

```
fra
```

- **PowerBuilder application script** Type the following in a PowerBuilder application script:

```
SQLCA.dbParm = "Locale = 'fra'"
```

What happens Setting the Locale DBParm parameter to *fra* has the same effect as individually setting both the Language and CharSet DBParm parameters as follows:

```
Language = 'French'
CharSet = 'cp850'
```

Using the examples in a PowerBuilder application script If you specify DBParm parameters in your database profile, the PowerScript 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.

See also

CharSet
Language

Log

Description

Specifies whether the database server should log updates of text and image data in the SQL Server transaction log. By default, the database server logs updates of text and image data in the SQL Server transaction log.

Applies to

MSS Microsoft SQL Server 6.x
SYB and SYT SQL Server 4.x
SYC and SYD Sybase Systems 10.x and 11.x

Syntax

Log = *value*

Parameter	Description
value	A value that specifies whether the database server should log updates of text and image data in the SQL Server transaction log. Values are: ♦ **0** Do not log text and image updates in the SQL Server transaction log. Specify this value only if your database server allows you to disable logging ♦ **1** (Default) Log text and image updates in the SQL Server transaction log

Default value

Log = 1

Usage

You should set the Log parameter to 0 only if your database server allows you to disable logging.

Examples

To specify that PowerBuilder should *not* log text and image updates in the SQL Server transaction log:

- **Database profile** Clear the Log Text and Image Updates checkbox on the System tab or Transaction tab in the Database Profile Setup dialog box.
- **PowerBuilder application script** Type the following in a PowerBuilder application script:

```
SQLCA.dbParm = "Log = 0"
```

Using the examples in a PowerBuilder application script If you specify DBParm parameters in your database profile, the PowerScript 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.

LoginAttempts

Description

When PowerBuilder is connected to an IBM database through the IBM interface, LoginAttempts specifies the number of attempts you (the developer) or an application user can make to log in to the database by using the Powersoft IBM Interface dialog box. The Powersoft IBM Interface dialog box prompts for the user ID, password, and database required to log in to the database.

The value you specify for LoginAttempts determines the number of times the Powersoft IBM interface dialog box will display in the development environment or your application to prompt you for login information. To prevent the Powersoft IBM interface dialog box from ever displaying, set LoginAttempts to 0.

If you do not specify a value for LoginAttempts, the Powersoft IBM interface dialog box will display an unlimited number of times (up to 999) to prompt for login information.

Applies to

IBM

Syntax

LoginAttempts = *value*

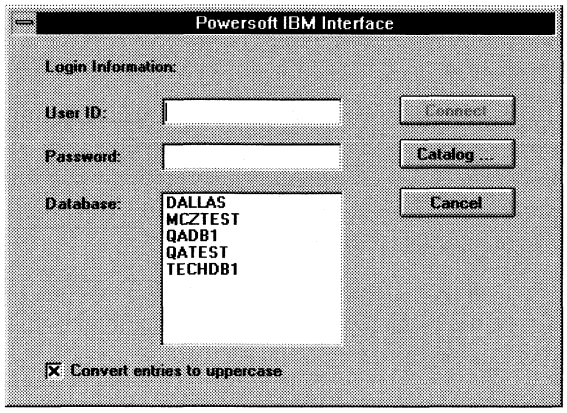
Parameter	Description
value	The number of attempts you or an application user can make to log in to an IBM database by using the Powersoft IBM Interface dialog box. This is the number of times that the Powersoft IBM Interface dialog box will display in the development environment or your application to prompt for login information (Default = 999 times) To prevent the Powersoft IBM Interface dialog box from ever displaying, set LoginAttempts to 0

Default value

LoginAttempts = 999

Usage

Powersoft IBM Interface dialog box If you set LoginAttempts to a value greater than 0, the Powersoft IBM Interface dialog box displays if necessary in the development environment or your application to prompt for login information.



FOR INFO For instructions on using the Powersoft IBM interface dialog box, see the section on IBM databases in *Connecting to Your Database (for InfoMaker)*.

Setting LoginAttempts in an application Setting LoginAttempts in an application is useful in the following situations:

- **Setting LoginAttempts to 0** If your application has its own dialog box to prompt the user for login information and you want to prevent the Powersoft IBM Interface dialog box from ever displaying, set LoginAttempts to 0.
- **Setting LoginAttempts to a small value** To control the number of times users can try to log in to an IBM database, set LoginAttempts to a small value (such as 3).

Examples

To set LoginAttempts to 0:

- **Database profile** Type the following in the Login Attempts box on the Connection tab in the Database Profile Setup dialog box:

```
0
```

- **PowerBuilder application script** Type the following:

```
SQLCA.dbParm = "LoginAttempts = 0"
```

Using the examples in a PowerBuilder application script If you specify DBParm parameters in your database profile, the PowerScript 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.

LoginTimeOut

Description

Specifies the number of seconds the ODBC driver should wait for a login request to an ODBC data source. If you set LoginTimeOut to 0, PowerBuilder waits the number of seconds specified by the ODBC driver's client software.

Applies to

ODBC (if driver and backend DBMS support this feature)

Syntax

LoginTimeOut = *value*

Parameter	Description
value	The number of seconds you want the ODBC driver to wait for an ODBC login request (Default = 15 seconds) To wait for the number of seconds specified by the ODBC driver's client software, set the value to 0

Default value

LoginTimeOut = 15

Examples

To set the LoginTimeOut value to wait 60 seconds for an ODBC login request:

- **Database profile** Type the following in the Login Timeout box on the Network tab in the Database Profile Setup dialog box:

```
60
```

- **PowerBuilder application script** Type the following in a PowerBuilder application script:

```
SQLCA.dbParm = "LoginTimeOut = 60"
```

Using the examples in a PowerBuilder application script If you specify DBParm parameters in your database profile, the PowerScript 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.

MixedCase

Description

Specifies whether you want connections to an Oracle database to be case sensitive or case insensitive.

By default, MixedCase is set to 0. This setting specifies a case insensitive connection, and assumes that all identifiers are uppercase. To make the Oracle connection case sensitive, set the MixedCase parameter to 1.

Applies to

Oracle (all interfaces)

Syntax

MixedCase = *value*

Parameter	Description
value	Specifies whether an Oracle database connection is case sensitive or case insensitive. Values are: ♦ **0** (Default) The Oracle database connection is case insensitive. It assumes that all identifiers are uppercase ♦ **1** The Oracle database connection is case sensitive. It supports mixed case, uppercase, and lowercase identifiers

Default value

MixedCase = 0

Usage

When you set the MixedCase parameter to 1 and define a primary key for a table in an Oracle database, all of the following must contain only uppercase letters:

- The name of the primary key
- The name of the table containing the primary key
- The names of any foreign keys that reference the primary key

Examples

To make an Oracle database connection case sensitive:

- **Database profile** Select the Case Sensitive checkbox on the Connection tab in the Database Profile Setup dialog box.
- **PowerBuilder application script** Type the following in a PowerBuilder application script:

```
SQLCA.dbParm = "MixedCase = 1"
```

Using the examples in a PowerBuilder application script If you specify DBParm parameters in your database profile, the PowerScript 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.

ModifySyntax

Description

When you access a database with the Sybase Systems 10.x and 11.x interface, PowerBuilder prepares SQL statements by submitting a query to Sybase SQL Server to get a description of the columns in the result set. To optimize performance, PowerBuilder modifies the WHERE clause of this query by default.

For some complex SQL queries containing nested SUBSELECT statements, modifying the WHERE clause generates illegal SQL syntax. If you are issuing a complex query, you can turn off the default WHERE clause modification by setting ModifySyntax to 0.

Applies to

SYC Sybase Systems 10.x. and 11.x

Syntax

ModifySyntax = *value*

Parameter	Description
value	Specifies whether PowerBuilder modifies the WHERE clause to optimize peformance when it queries Sybase SQL Server for result set descriptions. Values are: ♦ **0** PowerBuilder does not modify the WHERE clause when querying Sybase SQL Server for a result set description. Use this setting to avoid syntax errors when you are issuing complex SQL queries containing nested SUBSELECT statements. You can also specify **'N'** or **'False'** to set this value ♦ **1** (Default) PowerBuilder modifies the WHERE clause when querying Sybase SQL Server for a result set description. You can also specify 'Yes' or **'True'** to set this value

Default value

ModifySyntax = 1

Usage

If ModifySyntax is set to 1 and you issue a complex SQL query containing a nested SUBSELECT statement, an error message may display indicating that the SQL syntax is incorrect. To help you avoid such syntax errors, set ModifySyntax to 0 to turn off PowerBuilder's default modification of the WHERE clause.

Examples

To turn off PowerBuilder's default modification of the WHERE clause:

- **Database profile** Clear the Optimize Syntax for Prepare checkbox on the Syntax tab in the Database Profile Setup dialog box.

- **PowerBuilder application script** Type the following in a PowerBuilder script:

```
SQLCA.DBParm = "ModifySyntax = 0"
```

Using the examples in a PowerBuilder application script If you specify DBParm parameters in your database profile, the PowerScript 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.

MsgTerse

Description

Specifies whether PowerBuilder should display terse error messages for ODBC data source drivers. A terse ODBC error message is one without the SQLSTATE = *nnnnn* prefix, where *nnnn* is the number of the error message.

By default, PowerBuilder displays ODBC error messages with the SQLSTATE prefix. To display ODBC error messages without the SQLSTATE prefix, set MsgTerse to 'Yes'.

Applies to

ODBC (if driver and backend DBMS support this feature)

Syntax

MsgTerse = '*value*'

Parameter	Description
value	Specifies whether PowerBuilder should display ODBC error messages without the SQLSTATE prefix. Values are: ♦ **Yes** Display ODBC error messages *without* the SQLSTATE prefix ♦ **No** (Default) Display ODBC error messages *with* the SQLSTATE prefix

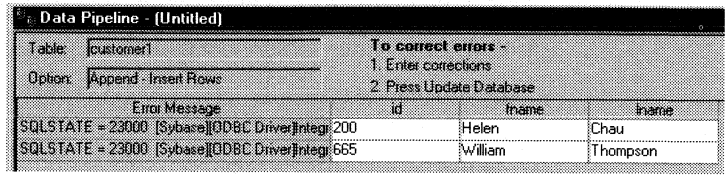
Default value

MsgTerse = 'No'

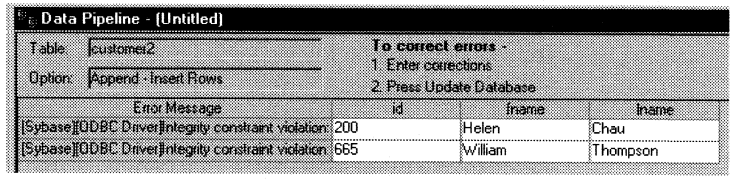
Usage

You can set the MsgTerse DBParm parameter to 'Yes' to display shorter ODBC error messages in PowerBuilder. This may be useful if space on your screen is limited.

For example, suppose you are using the Data Pipeline in PowerBuilder to pipe data to a SQL Anywhere ODBC database, and errors occur while you are executing the pipeline. If MsgTerse is set to 'No' (the default value), pipeline errors display in an Error dialog box *with* the SQLSTATE prefix (for example, SQLSTATE = 23000).



If you specify MsgTerse = 'Yes' in the database profile of the SQL Anywhere destination database, the Data Pipeline displays terse ODBC error messages *without* the SQLSTATE prefix.



FOR INFO For instructions on using the Data Pipeline, see the *PowerBuilder User's Guide.*

Examples

To specify that PowerBuilder should display terse ODBC error messages without the SQLSTATE prefix:

- **Database profile** Select the Display Terse Error Messages checkbox on the System tab in the Database Profile Setup dialog box.
- **PowerBuilder application script** Type the following in a PowerBuilder application script:

```
SQLCA.dbParm = "MsgTerse = 'Yes'"
```

Using the examples in a PowerBuilder application script If you specify DBParm parameters in your database profile, the PowerScript 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.

NumericFormat

Description

If supported by the DBMS or backend database, setting NumericFormat tells the driver to do special formatting of numeric strings in SQL syntax. This formatting affects how PowerBuilder generates numeric values in the SQL syntax it internally builds in DataWindow objects and sends to your database.

Currently, this feature is supported in PowerBuilder when you access:

- IBM databases (such as DB2/MVS) through the Powersoft IBM interface
- ODBC data sources (such as Oracle) through the Powersoft ODBC interface

Applies to

IBM
ODBC (if driver and backend DBMS support this feature)

Syntax

There is no single syntax that covers all uses of NumericFormat. The syntax you use depends on the backend DBMS you are accessing and how you want to format the numeric string.

The following are typical syntax examples for IBM DB2 and Oracle databases that format a numeric string with a comma as the decimal separator. (See the Examples section below for information about how PowerBuilder generates numeric values in the SQL syntax it builds and sends to the database.)

In the PowerBuilder development environment, the Database Profile Setup dialog box inserts special characters (quotes) where needed, so you can specify just the NumericFormat value (%s in this example).

In a PowerBuilder application script, you must use the following syntax:

IBM DB2 syntax If you are accessing an IBM DB2 database through the Powersoft IBM or ODBC interface, use the following syntax for NumericFormat. Note the use of *one single quote* at the beginning and end of the string.

```
NumericFormat = '%s,%s'
```

Oracle ODBC syntax If you are accessing an Oracle database through the Powersoft ODBC interface, use the following syntax for NumericFormat. Note the use of *three single quotes* at the beginning and end of the string.

```
NumericFormat = '''%s,%s'''
```

Parameter	Description
'	**IBM DB2 syntax** Type a single open quote. PowerBuilder returns no open quote in the SQL syntax it builds and sends to the database, as required by IBM DB2 databases
'''	**Oracle ODBC syntax** Type three single open quotes. PowerBuilder parses the second and third quotes as one single open quote in the SQL syntax it builds and sends to the database
%s	Represents one or more digits to the *left of the decimal* in the numeric string. PowerBuilder substitutes this value with the digits to the left of the decimal when it builds the SQL syntax
,	Represents the decimal separator character (in this case, a comma)
%s	Represents one or more digits to the *right of the decimal* in the numeric string. PowerBuilder substitutes this value with the digits to the right of the decimal when it builds the SQL syntax
'	**IBM DB2 syntax** Type one single closed quote. PowerBuilder returns no closed quote in the SQL syntax it builds and sends to the database, as required by IBM DB2 databases
'''	**Oracle ODBC syntax** Type three single closed quotes. PowerBuilder parses the first and second quotes as one single closed quote in the SQL syntax it builds and sends to the database

Default value

None

Usage

When to set NumericFormat In general, you should *not* need to set the NumericFormat DBParm parameter. Most backend DBMSs do not require that the driver do special formatting of numeric strings in SQL syntax. However, some databases may require special formatting, such as an IBM DB2/MVS database server configured to use a comma as the decimal separator.

In these cases, setting NumericFormat allows you to generate numeric values with special formatting in the SQL syntax that PowerBuilder builds in DataWindow objects and sends to your database. For example, if the decimal separator for your DBMS is a comma, you may want to set NumericFormat as shown in the Examples section below to use a comma as the decimal delimiter in the SQL syntax sent to your database.

An alternative to NumericFormat for IBM DB2 databases If you are connecting to an IBM DB2 database with IBM Client Application Enabler (CAE) software Version 2.x or higher, you can specify a comma as the decimal delimiter (separator) when you bind PowerBuilder to your database.

To do so, include the optional DECDEL COMMA clause in the BIND command to set the COMMA column in the SYSIBM.SYSPACKAGE table to Y (Yes).

FOR INFO For instructions on using the DECDEL COMMA clause when you bind PowerBuilder to your IBM database, see your IBM documentation.

Examples

Example 1 (IBM DB2 syntax) This example shows how to specify that you want PowerBuilder to generate two numeric values in the format *125,50* and *4,0*. PowerBuilder uses the comma as a decimal separator in the SQL syntax it builds in DataWindow objects and sends to an IBM DB2 database.

- **Database profile** Type the following in the Numeric Format box on the Syntax tab in the Database Profile Setup dialog box:

```
%s,%s
```

- **PowerBuilder application script** Type the following in a PowerBuilder application script:

```
SQLCA.dbParm = "NumericFormat = '%s,%s'"
```

What happens PowerBuilder internally builds the following SQL INSERT statement in the DataWindow object and sends the syntax to your database. PowerBuilder returns no quotes in the SQL syntax.

```
INSERT INTO MYTABLE (a, b)
VALUES (125,50, 4,0)
```

Example 2 (Oracle ODBC syntax) This example shows how to specify that you want PowerBuilder to generate two numeric values in the format *'125,50'* and *'4,0'*. PowerBuilder uses the comma as a decimal separator in the SQL syntax it builds in DataWindow objects and sends to an Oracle database.

- **Database profile** Type the following in the Numeric Format box on the Syntax tab in the Database Profile Setup dialog box:

```
%s,%s
```

- **PowerBuilder application script** Type the following in a PowerBuilder application script:

```
SQLCA.dbParm = "NumericFormat = '''%s,%s'''"
```

What happens PowerBuilder internally builds the following SQL INSERT statement in the DataWindow object and sends the syntax to your database. PowerBuilder returns single quotes in the SQL syntax.

```
INSERT INTO MYTABLE (a, b)
VALUES ('125,50', '4,0')
```

Using the examples in a PowerBuilder application script If you specify DBParm parameters in your database profile, the PowerScript 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.

PacketSize (ODBC)

Description

Specifies the network packet size in bytes when you access an ODBC data source in PowerBuilder.

Many backend DBMSs either do not support the PacketSize option or can only return the current network packet size. For information about whether the DBMS you are accessing supports PacketSize, see your DBMS documentation.

When to specify PacketSize
If your backend DBMS supports it, you must specify the PacketSize DBParm parameter *before* connecting to the database in PowerBuilder.

Applies to

ODBC (if ODBC 2.0 or higher driver and backend DBMS support this feature)

Syntax

PacketSize = *value*

Parameter	Description
value	A 32-bit integer value that specifies the network packet size in bytes

Default value

The default value for PacketSize is the default for your backend DBMS.

Usage

If the PacketSize value you specify is larger than the maximum network packet size or smaller than the minimum network packet size, your ODBC driver substitutes the maximum or minimum value for the value you specified.

Examples

To set the network packet size for an ODBC data source to 2048 bytes:

- **Database profile** Type the following in the Packet Size box on the Network tab in the Database Profile Setup dialog box:

```
2048
```

- **PowerBuilder application script** Type the following in a PowerBuilder application script:

```
SQLCA.dbParm = "PacketSize = 2048"
```

Using the examples in a PowerBuilder application script If you specify DBParm parameters in your database profile, the PowerScript 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.

PacketSize (SQL Server)

Description

When connecting to a SQL Server database, specifies the packet size in bytes that you want the server to set for transferring data to and from your PowerBuilder application. A **packet** is a fixed-size chunk of data for sending information over a network.

The server sets the actual packet size to a value less than or equal to the value specified with PacketSize. If the server has space limitations, it sets the packet size less than the specified PacketSize value. Otherwise, it sets the size equal to the PacketSize value.

When to specify PacketSize
You must specify the PacketSize DBParm parameter *before* connecting to the database in PowerBuilder.

Applies to

MSS Microsoft SQL Server 6.x
SYB and SYT SQL Server 4.x
SYC and SYD Sybase Systems 10.x and 11.x

Syntax

PacketSize = *value*

Parameter	Description
value	A value specifying the packet size in bytes that a SQL Server database server sets for transferring data to and from your application. The value must be a multiple of 512 bytes (Default = 512 bytes)

Default value

PacketSize = 512

Usage

When to use If your PowerBuilder application sends or receives large amounts of text or image data from the server, setting the PacketSize value larger than the default 512 bytes may speed performance since it results in fewer network read and write operations.

Microsoft SQL Server The PacketSize DBParm parameter is supported in Microsoft SQL Server 4.x and 6.x on both the Windows and Windows NT platforms.

Sybase SQL Server 4.x The PacketSize DBParm parameter is supported in Sybase SQL Server 4.x on the Windows NT platform but *not* on the Windows platform.

Sybase System 10.x or System 11.x Before setting PacketSize for use with a Sybase System 10.x or System 11.x database, you or your system administrator must set the following configuration variables on the server for PacketSize to take effect:

- **Additional netmem** This variable sets the maximum size of additional memory that can be used for network packets larger than the default size.
- **Maximum network packet size** This variable sets the maximum network packet size for all SQL Server users.

FOR INFO For instructions on setting these configuration variables, see your Sybase SQL Server documentation.

Examples

To specify that the SQL Server database server should set the packet size equal to or less than 2048 bytes:

- **Database profile** Type the following in the Packet Size box on the Network tab in the Database Profile Setup dialog box:

```
2048
```

- **PowerBuilder application script** Type the following in a PowerBuilder application script:

```
SQLCA.dbParm = "PacketSize = 2048"
```

Using the examples in a PowerBuilder application script If you specify DBParm parameters in your database profile, the PowerScript 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.

PBCatalogOwner

Description

Specifies a nondefault owner for the tables in the Powersoft repository. The Powersoft **repository** (also known as the Powersoft extended catalog or Powersoft system tables) consists of five tables that contain default extended attribute information for your database.

When you specify a PBCatalogOwner name that is different from the default repository owner for your DBMS, PowerBuilder creates a new set of repository tables with the owner name you specify.

FOR INFO For more about the Powersoft repository, see "About the Powersoft repository" on page 289.

When to specify PBCatalogOwner
You must specify the PBCatalogOwner DBParm parameter *before* connecting to the database in PowerBuilder.

Applies to

IBM
MSS Microsoft SQL Server 6.x
ODBC (if driver and backend DBMS support this feature)
Oracle (all interfaces)
MDI Sybase InformationConnect DB2 Gateway interface
NET Sybase Net-Gateway for DB2 interface
SYC and SYD Sybase Systems 10.x and 11.x

Syntax

PBCatalogOwner = '*owner_name*'

Parameter	Description
owner_name	Specifies the owner of the tables in the Powersoft repository **For DB2 databases** If you use the DB2SYSPB.SQL script to create the repository in a DB2 database and you replace all instances of PBOwner in the script with the name of a nondefault table owner, *owner_name* must be the same as the owner specified in the DB2SYSPB.SQL script

Default value

The default value for PBCatalogOwner depends on the DBMS you are accessing, as summarized in the following table:

DBMS	PBCatalogOwner default value
IBM	`PBCatalogOwner = 'PBCATOWN'`
Microsoft SQL Server 6.x	`PBCatalogOwner = 'dbo'`

DBMS	PBCatalogOwner default value
ODBC	If a value for PBCatalogOwner is not specified in the database profile or in the PBODB60 initialization file, the default value is the user ID specified in the database profile
Oracle	`PBCatalogOwner = 'SYSTEM'`
Sybase InformationConnect DB2 Gateway interface	If you do not specify a value for PBCatalogOwner, PowerBuilder prompts you to supply it
Sybase Net-Gateway for DB2 interface	If you do not specify a value for PBCatalogOwner, PowerBuilder prompts you to supply it
Sybase SQL Server System 10 and System 11	`PBCatalogOwner = 'dbo'`

Usage

By specifying a nondefault owner for the Powersoft repository tables, you are in effect creating an alternative repository. This is useful if you want to test new validation rules or display formats without overwriting the extended attributes currently in the default repository.

ODBC data sources When you connect to an ODBC data source and a value for PBCatalogOwner is set in both the database profile and the PBODB60 initialization file, the setting in the profile overrides the setting in the PBODB60 initialization file.

DB2 databases When you connect to a DB2 database, you can use the DB2SYSPB.SQL script to create the Powersoft repository tables. If you use the DB2SYSPB.SQL script, keep the following in mind:

- You can edit the script to change all instances of PBOwner to another name. Or you can leave the table owner as PBOwner in the script, which is the default.

  Specifying SYSIBM is prohibited
  You cannot specify SYSIBM as the table owner. This is prohibited by DB2.

- If you changed PBOwner to another name in the DB2SYSPB.SQL script, set the PBCatalogOwner parameter to the owner you specified in this script.

FOR INFO For instructions on using the DB2SYSPB.SQL script, see "Creating the Powersoft repository in DB2 databases" on page 271.

Examples

This example shows how to create a new set of Powersoft repository tables with the owner TEST. The names of the new tables will have the prefix TEST, such as TEST.pbcatcol, TEST.pbcatedt, and so on.

- **Database profile** Type the following in the PowerBuilder Catalog Table Owner box on the System tab in the Database Profile Setup dialog box:

```
TEST
```

- **PowerBuilder application script** Type the following in a PowerBuilder application script:

```
SQLCA.dbParm = "PBCatalogOwner = 'TEST'"
```

Using the examples in a PowerBuilder application script If you specify DBParm parameters in your database profile, the PowerScript 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.

PBDBMS

Description

PowerBuilder lets you use two kinds of Oracle stored procedures as data sources for DataWindow objects, depending on the version of your Oracle database server and how you set the PBDBMS DBParm parameter:

If your Oracle database server is	You can use as the data source	With this PBDBMS setting
Oracle Version 7.2 or higher	An Oracle stored procedure that has a result set as an IN OUT (reference) parameter	PBDBMS = 0
Oracle Version 7.x	An Oracle stored procedure that uses PBDBMS.Put_Line function calls to build the SQL SELECT statement	PBDBMS = 1 (default)

Applies to

Oracle (all interfaces)

Syntax

PBDBMS = *value*

Parameter	Description
value	Specifies what kind of Oracle stored procedure you want to use as the data source for a DataWindow object in PowerBuilder. Values are: ♦ **0** If your database server is Oracle Version 7.2 or higher, use as the data source an Oracle stored procedure that has a result set as an IN OUT (reference) parameter ♦ **1** (Default) If your database server is Oracle Version 7.x, use as the data source an Oracle stored procedure containing PBDBMS.Put_Line function calls

Default value

PBDBMS = 1

Usage

Using an Oracle stored procedure as the data source for DataWindow objects in PowerBuilder involves the following general steps:

1 Make sure you have installed the required version of the Oracle database server and client software.

2 Make sure you are using the correct Oracle database interface for your PowerBuilder platform and version of the Oracle database server.

FOR INFO For information, see "Supported versions and platforms for Oracle" on page 173.

3 If you are using an Oracle stored procedure with PBDBMS.Put_Line function calls, set up the database server.

4 Create the Oracle stored procedure.

5 Set the PBDBMS DBParm parameter to support the kind of Oracle stored procedure you want to use.

6 Create DataWindow objects that use the stored procedure as a data source.

FOR INFO For instructions and examples of how to use the different kinds of Oracle stored procedures, see "Using Oracle stored procedures as a data source" on page 191.

Examples

To specify that you want to use an Oracle stored procedure with a result set as an IN OUT (reference) parameter as the data source for a DataWindow in PowerBuilder:

♦ **Database profile** Clear the PBDBMS Checkbox on the Connection tab in the Database Profile Setup dialog box.

- **PowerBuilder application script** Type the following in a PowerBuilder application script:

```
SQLCA.dbParm = "PBDBMS = 0"
```

Using the examples in a PowerBuilder application script If you specify DBParm parameters in your database profile, the PowerScript 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.

PBUseProcOwner

Description

When you access a database through the Powersoft ODBC interface and define a DataWindow that uses a stored procedure as its data source, PBUseProcOwner specifies whether PowerBuilder should qualify the stored procedure with the owner name in the SQL EXECUTE statement passed to the ODBC driver.

PowerBuilder qualifies the stored procedure with an owner only if the owner associated with the stored procedure is different from the ID of the current user (the developer building the DataWindow or the user running the application containing the DataWindow).

Applies to

ODBC (if driver and backend DBMS support this feature)

Syntax

PBUseProcOwner = '*value*'

Parameter	Description

value	Specifies whether PowerBuilder should qualify the stored procedure with its owner name in the SQL EXECUTE statement built by the DataWindow and passed to the ODBC driver. Values are:
	♦ **Yes** If the owner associated with the stored procedure is different from the current user ID, PowerBuilder qualifies the stored procedure with its owner name in the SQL EXECUTE statement and passes this information to the ODBC driver. This allows users to execute stored procedures that they do not own. For example:
	`EXECUTE FRAN.MYPROCEDURE`
	♦ **No** (Default) PowerBuilder does not qualify the stored procedure with its owner name in the SQL EXECUTE statement passed to the ODBC driver. For example:
	`EXECUTE MYPROCEDURE`

Default value

PBUseProcOwner = 'No'

Usage

Determining the PBUseProcOwner value PowerBuilder searches the following in this order to determine the PBUseProcOwner value:

1 The section for your database profile in the PowerBuilder initialization file (in the development environment) or the value of the transaction object DBParm property (in a PowerBuilder application)

2 The section for your ODBC driver in the PBODB60 initialization file

If PowerBuilder does not find a PBUseProcOwner value in these locations, it defaults to a value of 'No'.

If DBA owns the SQL Anywhere stored procedure DBA (database administrator) is a reserved word in SQL Anywhere syntax. If you define a DataWindow object with a SQL Anywhere stored procedure as its data source and DBA owns the stored procedure, the painter passes the following SQL EXECUTE statement to the SQL Anywhere ODBC driver if PBUseProcOwner is set to Yes:

```
EXECUTE DBA.MYPROCEDURE
```

This statement generates a syntax error because it includes the DBA reserved word.

If DBA owns the SQL Anywhere stored procedure you are using, you can avoid this syntax error by setting PBUseProcOwner to No so that PowerBuilder does not qualify the stored procedure with DBA.

In some situations, however, you *must* qualify the stored procedure with the DBA owner. For example, the DBA may want to grant execute permission to another user ID. In this case, you can avoid errors by editing the SQL EXECUTE syntax to enclose DBA in quotes, like this:

```
EXECUTE "DBA".MYPROCEDURE
```

Examples

To specify that PowerBuilder should qualify the stored procedure with its owner name in the SQL EXECUTE statement:

- **Database profile** Select the Qualify Stored Procedures with Owner Name checkbox on the Transaction tab in the Database Profile Setup dialog box.
- **PowerBuilder application script** Type the following in a PowerBuilder application script:

```
SQLCA.dbParm = "PBUseProcOwner = 'Yes'"
```

Using the examples in a PowerBuilder application script If you specify DBParm parameters in your database profile, the PowerScript 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.

PWDialog

Description

For those database interfaces that support it, PWDialog controls whether a Password Expired dialog box displays if necessary in a PowerBuilder application at execution time.

When PWDialog is set to 1, the Password Expired dialog box prompts users to change their password if they attempt to log in to the database with an expired password. By default, PWDialog is set to 0 to specify that the Password Expired dialog box will not display in your application at execution time.

The setting of PWDialog affects PowerBuilder applications only at execution time. It has no effect in the PowerBuilder development environment because, regardless of the PWDialog setting, the Change Password dialog box displays in the development environment to prompt users to change an expired password.

When to specify PWDialog
You must specify a value for PWDialog *before* connecting to the database in PowerBuilder.

Applies to

SYC and SYD Sybase Systems 10.x and 11.x

Syntax

PWDialog = *value*

Parameter	Description
value	Specifies whether the Password Expired dialog box displays in a PowerBuilder application at execution time to prompt the user to change an expired login password. Values are: ♦ **0** (Default) Do not display the Password Expired dialog box at execution time ♦ **1** Display the Password Expired dialog box at execution time to prompt the user to change an expired password

Default value

PWDialog = 0

Usage

When to use Setting PWDialog to 1 to display the Password Expired dialog box in your executable application provides a convenient way for you to notify your users that their password has expired and allow them to change it.

What happens When the Password Expired dialog box displays in your PowerBuilder application at execution time, it notifies users that the password for their login ID has expired and prompts them to supply a new password. It then runs the sp_password system stored procedure to set the new password. Once the password has been changed, the database connection succeeds.

If the user clicks Cancel to close the Password Expired dialog box without changing the password, the database connection fails and a message displays indicating that the password has expired.

Examples

To display the Password Expired dialog box when needed in your PowerBuilder application at execution time:

- **Database profile** Although the setting of PWDialog has no effect in the development environment, you may want to set it in your database profile to generate the corresponding PowerScript syntax on the Preview tab that you can copy into an application script. Select the Display Runtime Dialog When Password Expires checkbox on the Connection tab in the Database Profile Setup dialog box.

- **PowerBuilder application script** Type the following in a PowerBuilder script:

```
SQLCA.DBParm = "PWDialog = 1"
```

Using the examples in a PowerBuilder application script If you specify DBParm parameters in your database profile, the PowerScript 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.

PWEncrypt

Description

If you are using Sybase Open Client software Version 10.0.2 P2 or higher, PWEncrypt specifies whether you want Open Client to automatically encrypt your password when connecting to a Sybase System 10.x or System 11.x database in PowerBuilder.

You must install Sybase Open Client software Version 10.0.2 P2 or higher in order to use the PWEncrypt DBParm parameter.

When to specify PWEncrypt
You must specify the PWEncrypt DBParm parameter *before* connecting to the database in PowerBuilder.

Applies to

SYC and SYD Sybase Systems 10.x and 11.x

Syntax

PWEncrypt = '*value*'

Parameter	Description
value	Specifies whether you want the Open Client software to encrypt your password. Values are: ♦ **Yes** (Default) Tells Open Client to *encrypt the password* by setting the CS_SEC_ENCRYPTION connection property to CS_TRUE ♦ **No** Tells Open Client *not to encrypt the password* by setting the CS_SEC_ENCRYPTION connection property to CS_FALSE

Default value

PWEncrypt = 'Yes'

Usage

On UNIX In PowerBuilder on the UNIX platforms, you can use either Release 10.0.1 or Release 10.0.3 or higher of the Sybase Open Client/Server client software to connect to a System 10.x or System 11.x database. Release 10.0.3 or higher *supports* password encryption, but Release 10.0.1 *does not support* password encryption. By default, PowerBuilder tells the Open Client software to encrypt your password when connecting to a System 10.x or System 11.x database.

Since password encryption is supported in Open Client/Server Release 10.0.3 or higher but not in Release 10.0.01, you must set PWEncrypt as follows in PowerBuilder depending on the release of Open Client/Server you are using:

Sybase Open Client/Server	How to set PWEncrypt in PowerBuilder for UNIX
Release 10.0.1	`PWEncrypt='No'` If you do not set PWEncrypt to No to turn off PowerBuilder's default password encryption behavior, an invalid login message displays when PowerBuilder tries to connect
Release 10.0.3 or higher	You do not need to set PWEncrypt because the default value (Yes) is supported by Sybase Open Client/Server Release 10.0.3 or higher

Examples

To tell Open Client *not to encrypt your password* when connecting to a Sybase Systems 10.x or 11.x database in PowerBuilder:

- **Database profile** Clear the Encrypt Password checkbox on the Network tab in the Database Profile Setup dialog box.
- **PowerBuilder application script** Type the following in a PowerBuilder application script:

```
SQLCA.dbParm = "PWEncrypt='No'"
```

Using the examples in a PowerBuilder application script If you specify DBParm parameters in your database profile, the PowerScript 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.

Release (SQL Server 4.x)

Description

You can set the Release DBParm parameter to '4.2' when connecting to a SQL Server Version 4.x database in PowerBuilder. This specifies that you want to use SQL Server DB-Library cursor processing instead of the default PowerBuilder cursor processing.

Applies to

SYB and SYT SQL Server 4.x

Syntax

Release = '4.2'

Default value

By default, the Release parameter is not set, and PowerBuilder uses its own default cursor processing.

Usage

You *must* set the Release parameter to '4.2' in any of the following situations:

- You are using the CursorLock and CursorScroll DBParm parameters with a SQL Server database.
- You want to use features of SQL Server DB-Library cursor processing, such as the ability to create scroll cursors.
- You want to use SQL UNION statements.
- You want to use any of the following SQL Server data types:
  - Real
  - SmallDateTime
  - SmallMoney

In general, SQL Server cursor processing is slower than default cursor processing in PowerBuilder. Therefore, using SQL Server cursor processing may slow the performance of your application.

Examples

Example 1 To specify that you want to use SQL Server DB-Library cursor processing when connecting to a SQL Server Version 4.x database:

- **Database profile** Select the Release 4.2 checkbox on the Connection tab in the Database Profile Setup dialog box.
- **PowerBuilder application script** Type the following in a PowerBuilder application script:

```
SQLCA.dbParm = "Release = '4.2'"
```

Example 2 You can set the Release, CursorScroll, and CursorLock parameters together for a SQL Server database. To set keyset-driven scrolling and OptVal locking options for cursors in a SQL Server Version 4.x database:

- **Database profile** On the Connection tab in the Database Profile Setup dialog box, select the Release 4.2 checkbox. On the Transaction tab, select Keyset-Driven from the Scrolling Options dropdown listbox and OptVal from the Locking dropdown listbox.
- **PowerBuilder application script** Type the following in a PowerBuilder application script:

```
SQLCA.dbParm = "Release='4.2',
   CursorScroll='KeySet',CursorLock='OptVal'"
```

Using the examples in a PowerBuilder application script If you specify DBParm parameters in your database profile, the PowerScript 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.

See also

CursorLock (SQL Server)
CursorScroll (SQL Server)

Release (Sybase System 10 and System 11)

Description

When you are accessing a Sybase System 10.x or 11.x database on the Windows or UNIX platform, Release specifies whether your PowerBuilder application should use Sybase Open Client Client-Library (CT-Lib) 10.x or 11.x behavior. This is *different* from the Release (SQL Server 4.x) on page 504.

By default, Release is set to '10' to indicate that you want your application to use Open Client CT-Lib 10.x behavior. If you want your application to take advantage of Open Client 11.1 features such as network-based security or directory services, you *must* set Release to '11' to specify that your application use Open Client CT-Lib 11.x behavior. (Open Client CT-Lib 10.x behavior does not support security or directory services.)

At this time, the only valid values for Release are '10' or '11'; no other values are permitted.

When to specify Release
You must specify a value for Release *before* connecting to the database in PowerBuilder.

Applies to

SYC and SYD Sybase Systems 10.x and 11.x

Syntax

Release = '*value*'

Parameter	Description
value	Specifies whether your PowerBuilder application should use Sybase Open Client CT-Lib 10.x or 11.x behavior. Values are: ♦ **10** (Default) Specifies that your PowerBuilder application should use CT-Lib 10.x behavior. This sets the CT-Lib version variable to CS_VERSION_100 ♦ **11** Specifies that your PowerBuilder application should use CT-Lib 11.x behavior. This sets the CT-Lib version variable to CS_VERSION_110

Default value

Release = '10'

Usage

When to use Certain features are supported only when you use a specified version of Sybase Open Client to access a SQL Server 10.x or 11.x database. For example, you *must* be using Open Client 11.1 or higher to take advantage of network-based security and directory services in your application.

Therefore, to use any of the DBParm parameters in PowerBuilder that support Open Client 11.1 security and directory services, *you must set Release to '11'* to specify CT-Lib 11.x behavior. Otherwise, you can leave Release set to '10' to specify the default CT-Lib 10.x behavior.

Examples

To specify that your PowerBuilder application should use Sybase Open Client CT-Lib 11.x behavior:

- **Database profile** Select the Release 11 checkbox on the Connection tab in the Database Profile Setup dialog box.
- **PowerBuilder application script** Type the following in a PowerBuilder script:

```
SQLCA.DBParm = "Release = '11'"
```

Using the examples in a PowerBuilder application script If you specify DBParm parameters in your database profile, the PowerScript 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.

Request

Description

When you connect to an IBM DB2 database through the Sybase InformationConnect DB2 Gateway interface, the Request parameter specifies whether you want to release IBM Customer Information Control System (CICS) transaction resources after each request. To do so, set the value of Request to 1.

By default, Request is set to 0, which maintains the CICS resources for the duration of the request.

Applies to

MDI Sybase InformationConnect DB2 Gateway interface

Syntax

Request = *value*

Parameter	Description
value	Specifies whether you want to release CICS transaction resources after each request. Values are: ♦ **0** (Default) Maintain CICS resources for the duration of the request ♦ **1** Release CICS resources after each request

Default value

Request = 0

Usage

Requirements for using the Request parameter Setting the Request parameter to 1 to release CICS resources has an effect only when you do both of the following:

- Set the AutoCommit database preference to True to specify that PowerBuilder should issue SQL statements outside the scope of a transaction. (See the description of AutoCommit on page 566.)
- Specify the value for Request *before* connecting to a DB2 database through the Sybase InformationConnect Gateway interface.

What happens When you set the Request parameter to 1, a CICS transaction is started for each request. This may slow the performance of your application.

Examples

To specify that you want to release CICS transaction resources after each request:

- **Database profile** Select the Release CICS Transaction Resources After Each Request checkbox on the Transaction tab in the Database Profile Setup dialog box.

- **PowerBuilder application script** Type the following in a PowerBuilder application script:

```
SQLCA.dbParm = "Request = 1"
```

Using the examples in a PowerBuilder application script If you specify DBParm parameters in your database profile, the PowerScript 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.

See also

AutoCommit

Scroll

Description

Specifies whether you want to use a scroll cursor when connecting to an INFORMIX database in PowerBuilder. When you fetch rows in an INFORMIX table, using a **scroll cursor** enables you to fetch the next row, previous row, first row, or last row.

By default, PowerBuilder does not use scroll cursors in an INFORMIX database connection.

Scroll not applicable on UNIX
Scroll is *not applicable* when accessing an INFORMIX database through the INFORMIX ODBC driver supplied in UNIX versions of PowerBuilder.

Applies to

IN5 and IN7 INFORMIX

Syntax

Scroll = *value*

Parameter	Description
value	Specifies whether you want to use a scroll cursor when connecting to an INFORMIX database in PowerBuilder. Values are: ♦ **0** (Default) Do not use a scroll cursor ♦ **1** Use a scroll cursor

Default value

Scroll = 0

Examples

To specify that you want to use a scroll cursor when connecting to an INFORMIX database in PowerBuilder:

- **Database profile** Select the Use a Scroll Cursor checkbox on the Transaction tab in the Database Profile Setup dialog box.
- **PowerBuilder application script** Type the following in a PowerBuilder application script:

```
SQLCA.dbParm = "Scroll = 1"
```

Using the examples in a PowerBuilder application script If you specify DBParm parameters in your database profile, the PowerScript 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.

Sec_Channel_Bind

Description

When you access a Sybase Sytems 10.x or 11.x database in PowerBuilder through Open Client 11.1 software, Sec_Channel_Bind is one of several DBParm parameters that supports login authentication for network-based security services. (For other login authentication DBParms, see the See Also section.)

Sec_Channel_Bind controls whether your connection's security mechanism performs channel binding. When Sec_Channel_Bind is set to 1, both Sybase Open Client Client-Library (CT-Lib) and the server provide a network channel identifier to the security mechanism before connecting. The channel identifier contains the network addresses of the client and server.

When Sec_Channel_Bind is set to 0 (the default), no channel binding is performed.

You must specify a value for Sec_Channel_Bind *before* connecting to the database in PowerBuilder.

Using third-party security mechanisms

FOR INFO For information about the third-party security mechanisms and operating system platforms that Powersoft has tested with Open Client 11.1 security services, see the Release Notes.

Applies to

SYC and SYD Sybase Systems 10.x and 11.x

Syntax

Sec_Channel_Bind = *value*

Parameter	Description
value	Specifies whether your connection's security mechanism performs channel binding. Values are: ♦ **0** (Default) Do *not* perform channel binding. You can also specify **'No'** or **'False'** to set this value ♦ **1** Perform channel binding. Both CT-Lib and the server provide a channel identifier to the connection's security mechanism. You can also specify **'Yes'** or **'True'** to set this value

Default value

Sec_Channel_Bind = 0

Usage

Not supported with CyberSafe Kerberos Sec_Channel_Bind is *not supported* if your security mechanism is CyberSafe Kerberos.

Set Release DBParm to '11' For Sec_Channel_Bind to take effect, you *must* also set the Release DBParm parameter to '11' to specify that your application should use CT-Lib 11.x behavior. (The default for Release is to use CT-Lib 10.x behavior, which does not support security services.)

Requirements for use To use Sec_Channel_Bind or any other DBParm parameter supporting Open Client 11.1 security services, you must meet certain requirements for using security services in your PowerBuilder application.

FOR INFO For details, see "Requirements for using Open Client security services" on page 263.

Corresponding CT-Lib connection property Specifying a value for Sec_Channel_Bind sets the corresponding Sybase CT-Lib 11.1 connection property named CS_SEC_CHANBIND.

Examples

To specify that your connection's security mechanism performs channel binding:

- **Database profile** Select the Enable Channel Binding checkbox on the Security tab in the Database Profile Setup dialog box.
- **PowerBuilder application script** Type the following in a PowerBuilder script:

```
SQLCA.DBParm = "Sec_Channel_Bind = 1"
```

Using the examples in a PowerBuilder application script If you specify DBParm parameters in your database profile, the PowerScript 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.

See also

Release (Sybase System 10 and System 11)
Sec_Cred_Timeout
Sec_Delegation
Sec_Keytab_File
Sec_Mechanism
Sec_Mutual_Auth
Sec_Network_Auth
Sec_Server_Principal
Sec_Sess_Timeout

Sec_Confidential

Description

When you access a Sybase Systems 10.x or 11.x database in PowerBuilder through Open Client 11.1 software, Sec_Confidential is one of several DBParm parameters that supports per-packet security for network-based security services. (For other per-packet security DBParms, see the See Also section.)

Sec_Confidential controls whether transmitted data is encrypted. When Sec_Confidential is set to 1, all requests sent to the server and all results returned by the server are encrypted.

When Sec_Confidential is set to 0 (the default), transmitted data is not encrypted.

You must specify a value for Sec_Confidential *before* connecting to the database in PowerBuilder.

Using third-party security mechanisms

FOR INFO For information about the third-party security mechanisms and operating system platforms that Powersoft has tested with Open Client 11.1 security services, see the Release Notes.

Applies to

SYC and SYD Sybase Systems 10.x and 11.x

Syntax

Sec_Confidential = *value*

Parameter	Description
value	Specifies whether transmitted data is encrypted. Values are: ♦ **0** (Default) Do *not* encrypt transmitted data. You can also specify **'No'** or **'False'** to set this value ♦ **1** Encrypt transmitted data. Requests sent to the server and results returned by the server are encrypted. You can also specify **'Yes'** or **'True'** to set this value

Default value

Sec_Confidential = 0

Usage

When to use Encryption can protect your data if you are sending it over a public network to a nonsecure server. In a networked environment, you may want to set Sec_Confidential to 1 to ensure that all requests sent to the server and all results returned by the server are encrypted.

Set Release DBParm to '11' For Sec_Confidential to take effect, you *must* also set the Release DBParm parameter to '11' to specify that your application should use Sybase Open Client Client-Library (CT-Lib) 11.x behavior. (The default for Release is to use CT-Lib 10.x behavior, which does not support security services.)

Requirements for use To use Sec_Confidential or any other DBParm parameter supporting Open Client 11.1 security services, you must meet certain requirements for using security services in your PowerBuilder application.

FOR INFO For details, see "Requirements for using Open Client security services" on page 263.

Corresponding CT-Lib connection property Specifying a value for Sec_Confidential sets the corresponding Sybase CT-Lib 11.1 connection property named CS_SEC_CONFIDENTIALITY.

Examples

To specify that transmitted data is encrypted:

- **Database profile** Select the Encrypt All Results checkbox on the Security tab in the Database Profile Setup dialog box.
- **PowerBuilder application script** Type the following in a PowerBuilder script:

```
SQLCA.DBParm = "Sec_Confidential = 1"
```

Using the examples in a PowerBuilder application script If you specify DBParm parameters in your database profile, the PowerScript 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.

See also

Release (Sybase System 10 and System 11)
Sec_Data_Integrity
Sec_Data_Origin
Sec_Replay_Detection
Sec_Seq_Detection

Sec_Cred_Timeout

Description

When you access a Sybase Systems 10.x or 11.x database in PowerBuilder through Open Client 11.1 software, Sec_Cred_Timeout is one of several DBParm parameters that supports login authentication for network-based security services. (For other login authentication DBParms, see the See Also section.)

Some security mechanisms allow applications to set credential timeout values for connections that use network-based login authentication. Sec_Cred_Timeout specifies the number of seconds remaining before a user's network credentials expire and become invalid. Users obtain network credentials when they log into the network.

By default, Sec_Cred_Timeout specifies that there is no credential timeout limit—the credentials will not expire.

You must specify a value for Sec_Cred_Timeout *before* connecting to the database in PowerBuilder.

Using third-party security mechanisms

FOR INFO For information about the third-party security mechanisms and operating system platforms that Powersoft has tested with Open Client 11.1 security services, see the Release Notes.

Applies to

SYC and SYD Sybase Systems 10.x and 11.x

Syntax

Sec_Cred_Timeout = *value*

Parameter	Description
value	Specifies the number of seconds remaining before a user's network credentials expire and become invalid. You can also specify **'no_limit'** (the default) to indicate that the credentials will not expire A credential timeout value set by the security system's administrator supersedes any value you specify for Sec_Cred_Timeout

Default value

Sec_Cred_Timeout = 'no_limit'

Usage

CyberSafe Kerberos If your security mechanism is CyberSafe Kerberos, Sec_Cred_Timeout cannot override the installation default value set for credential timeout.

Set Release DBParm to '11' For Sec_Cred_Timeout to take effect, you *must* also set the Release DBParm parameter to '11' to specify that your application should use Sybase Open Client Client-Library (CT-Lib) 11.x behavior. (The default for Release is to use CT-Lib 10.x behavior, which does not support security services.)

Requirements for use To use Sec_Cred_Timeout or any other DBParm parameter supporting Open Client 11.1 security services, you must meet certain requirements for using security services in your PowerBuilder application.

FOR INFO For details, see "Requirements for using Open Client security services" on page 263.

Corresponding CT-Lib connection property Specifying a value for Sec_Cred_Timeout sets the corresponding Sybase CT-Lib 11.1 connection property named CS_SEC_CREDTIMEOUT.

Examples

To specify that there are 120 seconds (2 minutes) remaining before a user's network credentials expire:

- **Database profile** Type the following in the Credential Timeout box on the Security tab in the Database Profile Setup dialog box:

```
120
```

- **PowerBuilder application script** Type the following in a PowerBuilder script:

```
SQLCA.DBParm = "Sec_Cred_Timeout = 120"
```

Using the examples in a PowerBuilder application script If you specify DBParm parameters in your database profile, the PowerScript 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.

See also

Release (Sybase System 10 and System 11)
Sec_Channel_Bind
Sec_Delegation
Sec_Keytab_File
Sec_Mechanism
Sec_Mutual_Auth
Sec_Network_Auth
Sec_Server_Principal
Sec_Sess_Timeout

Sec_Data_Integrity

Description

When you access a Sybase Systems 10.x or 11.x database in PowerBuilder through Open Client 11.1 software, Sec_Data_Integrity is one of several DBParm parameters that supports per-packet security for network-based security services. (For other per-packet security DBParms, see the See Also section.)'

Sec_Data_Integrity controls whether your connection's security mechanism checks the integrity of data transmitted to and from the server. When Sec_Data_Integrity is set to 1, the security mechanism analyzes all packets to ensure that their contents was not modified during transmission.

When Sec_Data_Integrity is set to 0 (the default), no integrity checking is performed.

You must specify a value for Sec_Data_Integrity *before* connecting to the database in PowerBuilder.

Using third-party security mechanisms

FOR INFO For information about the third-party security mechanisms and operating system platforms that Powersoft has tested with Open Client 11.1 security services, see the Release Notes.

Applies to

SYC and SYD Sybase Systems 10.x and 11.x

Syntax

Sec_Data_Integrity = *value*

Parameter	Description
value	Specifies whether your connection's security mechanism performs integrity checking on data transmitted to and from the server. Values are: ♦ **0** (Default) Do *not* check data integrity. You can also specify **'No'** or **'False'** to set this value ♦ **1** Check data integrity by analyzing all packets to ensure that their contents was not modified during transmission. You can also specify **'Yes'** or **'True'** to set this value

Default value

Sec_Data_Integrity = 0

Usage

When to use Your connection's security mechanism can check data integrity only when your connection is also using network-based login authentication.

FOR INFO For information, see your Sybase Open Client/Server documentation.

Set Release DBParm to '11' For Sec_Data_Integrity to take effect, you *must* also set the Release DBParm parameter to '11' to specify that your application should use Sybase Open Client Client-Library (CT-Lib) 11.x behavior. (The default for Release is to use CT-Lib 10.x behavior, which does not support security services.)

Requirements for use To use Sec_Data_Integrity or any other DBParm parameter supporting Open Client 11.1 security services, you must meet certain requirements for using security services in your PowerBuilder application.

FOR INFO For details, see "Requirements for using Open Client security services" on page 263.

Corresponding CT-Lib connection property Specifying a value for Sec_Data_Integrity sets the corresponding Sybase CT-Lib 11.1 connection property named CS_SEC_INTEGRITY.

Examples

To specify that your connection's security mechanism checks data integrity:

- **Database profile** Select the Ensure Data Integrity checkbox on the Security tab in the Database Profile Setup dialog box.
- **PowerBuilder application script** Type the following in a PowerBuilder script:

```
SQLCA.DBParm = "Sec_Data_Integrity = 1"
```

Using the examples in a PowerBuilder application script If you specify DBParm parameters in your database profile, the PowerScript 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.

See also

Release (Sybase System 10 and System 11)
Sec_Confidential
Sec_Data_Origin
Sec_Replay_Detection
Sec_Seq_Detection

Sec_Data_Origin

Description

When you access a Sybase Systems 10.x or 11.x database in PowerBuilder through Open Client 11.1 software, Sec_Data_Origin is one of several DBParm parameters that supports per-packet security for network-based security services. (For other per-packet security DBParms, see the See Also section.)'

Sec_Data_Origin controls whether your connection's security mechanism performs data origin stamping. When Sec_Data_Origin is set to 1, the security mechanism attaches a digital signature to each packet that verifies the packet's origin and contents.

When Sec_Data_Origin is set to 0 (the default), no data origin stamping is performed.

You must specify a value for Sec_Data_Origin *before* connecting to the database in PowerBuilder.

Using third-party security mechanisms

FOR INFO For information about the third-party security mechanisms and operating system platforms that Powersoft has tested with Open Client 11.1 security services, see the Release Notes.

Applies to

SYC and SYD Sybase Systems 10.x and 11.x

Syntax

Sec_Data_Origin = *value*

Parameter	Description
value	Specifies whether your connection's security mechanism performs data origin stamping. Values are: ♦ **0** (Default) Do *not* perform data origin stamping. You can also specify **'No'** or **'False'** to set this value ♦ **1** Perform data origin stamping by attaching a digital signature to each packet that verifies the packet's origin and contents. You can also specify **'Yes'** or **'True'** to set this value

Default value

Sec_Data_Origin = 0

Usage

Not supported with CyberSafe Kerberos Sec_Data_Origin is *not supported* if your security mechanism is CyberSafe Kerberos.

Set Release DBParm to '11' For Sec_Data_Origin to take effect, you *must* also set the Release DBParm parameter to '11' to specify that your application should use Sybase Open Client Client-Library (CT-Lib) 11.x behavior. (The default for Release is to use CT-Lib 10.x behavior, which does not support security services.)

Requirements for use To use Sec_Data_Origin or any other DBParm parameter supporting Open Client 11.1 security services, you must meet certain requirements for using security services in your PowerBuilder application.

FOR INFO For details, see "Requirements for using Open Client security services" on page 263.

Corresponding CT-Lib connection property Specifying a value for Sec_Data_Origin sets the corresponding Sybase CT-Lib 11.1 connection property named CS_SEC_DATAORIGIN.

Examples

To specify that your connection's security mechanism performs data origin stamping:

- **Database profile** Select the Verify Packet Origin checkbox on the Security tab in the Database Profile Setup dialog box.
- **PowerBuilder application script** Type the following in a PowerBuilder script:

```
SQLCA.DBParm = "Sec_Data_Origin = 1"
```

Using the examples in a PowerBuilder application script If you specify DBParm parameters in your database profile, the PowerScript 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.

See also

Release (Sybase System 10 and System 11)
Sec_Confidential
Sec_Data_Integrity
Sec_Replay_Detection
Sec_Seq_Detection

Sec_Delegation

Description

When you access a Sybase Systems 10.x or 11.x database in PowerBuilder through Open Client 11.1 software, Sec_Delegation is one of several DBParm parameters that supports login authentication for network-based security services. (For other login authentication DBParms, see the See Also section.)

For applications that are using network-based login authentication to connect to a Sybase Open Server gateway, Sec_Delegation controls whether the gateway server is allowed to connect to a remote SQL Server using delegated credentials. When Sec_Delegation is set to 1, the gateway can connect to a remote server using the client's delegated credentials. The remote server must also be using network-based authentication and an identical security mechanism.

When Sec_Delegation is set to 0 (the default), the gateway server cannot connect to a remote server using delegated credentials.

You must specify a value for Sec_Delegation *before* connecting to the database in PowerBuilder.

Using third-party security mechanisms

FOR INFO For information about the third-party security mechanisms and operating system platforms that Powersoft has tested with Open Client 11.1 security services, see the Release Notes.

Applies to

SYC and SYD Sybase Systems 10.x and 11.x

Syntax

Sec_Delegation = *value*

Parameter	Description
value	Specifies whether the Sybase Open Server gateway is allowed to connect to a remote SQL Server using the client's delegated credentials. Values are: ♦ **0** (Default) Prohibit the gateway from connecting to a remote server using delegated credentials. You can also specify **'No'** or **'False'** to set this value ♦ **1** Allow the gateway to connect to a remote server using delegated credentials. You can also specify **'Yes'** or **'True'** to set this value

Default value

Sec_Delegation = 0

Usage

Not supported with CyberSafe Kerberos Sec_Delegation is *not supported* if your security mechanism is CyberSafe Kerberos.

Set Release DBParm to '11' For Sec_Delegation to take effect, you *must* also set the Release DBParm parameter to '11' to specify that your application should use Sybase Open Client Client-Library (CT-Lib) 11.x behavior. (The default for Release is to use CT-Lib 10.x behavior, which does not support security services.)

Requirements for use To use Sec_Delegation or any other DBParm parameter supporting Open Client 11.1 security services, you must meet certain requirements for using security services in your PowerBuilder application.

FOR INFO For details, see "Requirements for using Open Client security services" on page 263.

Corresponding CT-Lib connection property Specifying a value for Sec_Delegation sets the corresponding Sybase CT-Lib 11.1 connection property named CS_SEC_DELEGATION.

Examples

To allow the Open Server gateway to connect to a remote server using delegated credentials:

- **Database profile** Select the Use Delegated Credentials checkbox on the Security tab in the Database Profile Setup dialog box.
- **PowerBuilder application script** Type the following in a PowerBuilder script:

```
SQLCA.DBParm = "Sec_Delegation = 1"
```

Using the examples in a PowerBuilder application script If you specify DBParm parameters in your database profile, the PowerScript 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.

See also

Release (Sybase System 10 and System 11)
Sec_Channel_Bind
Sec_Cred_Timeout
Sec_Keytab_File
Sec_Mechanism
Sec_Mutual_Auth
Sec_Network_Auth
Sec_Server_Principal
Sec_Sess_Timeout

Sec_Keytab_File

Description

When you access a Sybase Systems 10.x or 11.x database in PowerBuilder through Open Client 11.1 software, Sec_Keytab_File is one of several DBParm parameters that supports login authentication for network-based security services. (For other login authentication DBParms, see the See Also section.)

Sec_Keytab_File applies only to connections using Distributed Computing Environment (DCE) Kerberos as their security mechanism and requesting network-based login authentication. For those connections, Sec_Keytab_File specifies the name of the keytab file containing the security key for the DCE user.

You *must* set Sec_Keytab_File if the login ID specified in the database profile or PowerBuilder application script is *different* from the username of the DCE user currently running the application.

You must specify a value for Sec_Keytab_File *before* connecting to the database in PowerBuilder.

Using third-party security mechanisms

FOR INFO For information about the third-party security mechanisms and operating system platforms that Powersoft has tested with Open Client 11.1 security services, see the Release Notes.

Applies to

SYC and SYD Sybase Systems 10.x and 11.x

Syntax

Sec_Keytab_File = '*keytab_filename*'

Parameter	Description
keytab_filename	The name of the keytab file containing the security key for the DCE user

Default value

None

PowerBuilder does not set Sec_Keytab_File or the corresponding Sybase Open Client Client-Library (CT-Lib) 11.1 connection parameter CS_SEC_KEYTAB if you do not specify a value.

Usage

Supported only with Distributed Computing Environment Only Distributed Computing Environment (DCE) security servers and clients support the use of keytab files. Therefore, Sec_Keytab_File is supported only when your security mechanism is DCE Kerberos.

When to use If you want your application to be able to connect to a server with a different user name (login ID) than the DCE user currently running the application, set Sec_Keytab_File to specify the name of the keytab file containing the security key for the desired user.

FOR INFO For details, see "Requirements for using Open Client security services" on page 263.

Set Release DBParm to '11' For Sec_Keytab_File to take effect, you *must* also set the Release DBParm parameter to '11' to specify that your application should use Sybase Open Client Client-Library (CT-Lib) 11.x behavior. (The default for Release is to use CT-Lib 10.x behavior, which does not support security services.)

Requirements for use To use Sec_Keytab_File or any other DBParm parameter supporting Open Client 11.1 security services, you must meet certain requirements for using security services in your PowerBuilder application.

FOR INFO For details, see "Requirements for using Open Client 11.1 security services."

Corresponding CT-Lib connection property Specifying a value for Sec_Keytab_File sets the corresponding Sybase CT-Lib 11.1 connection property named CS_SEC_KEYTAB.

Examples

To specify C:\DCE_KEY as the name of the DCE keytab file:

- **Database profile** Type the following in the Keytab File box on the Security tab in the Database Profile Setup dialog box:

```
C:\DCE_KEY
```

- **PowerBuilder application script** Type the following in a PowerBuilder script:

```
SQLCA.DBParm = "Sec_Keytab_File = 'C:\DCE_KEY'"
```

Using the examples in a PowerBuilder application script If you specify DBParm parameters in your database profile, the PowerScript 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.

See also

Release (Sybase System 10 and System 11)
Sec_Channel_Bind
Sec_Cred_Timeout
Sec_Delegation
Sec_Mechanism
Sec_Mutual_Auth
Sec_Network_Auth
Sec_Server_Principal
Sec_Sess_Timeout

Sec_Mechanism

Description

When you access a Sybase Systems 10.x or 11.x database in PowerBuilder through Open Client 11.1 software, Sec_Mechanism is one of several DBParm parameters that supports login authentication for network-based security services. (For other login authentication DBParms, see the See Also section.)

When you use Open Client 11.1 security services, you must specify the name of the security mechanism you want to use in the Open Client/Open Server Configuration utility so the required drivers can be loaded. The default security mechanism is the one currently specified as active in the Configuration utility.

Sec_Mechanism lets you specify a security mechanism name listed in the Open Client/Open Server Configuration utility *other than* the default (activc) mechanism.

You must specify a value for Sec_Mechanism *before* connecting to the database in PowerBuilder.

Using third-party security mechanisms

FOR INFO For information about the third-party security mechanisms and operating system platforms that Powersoft has tested with Open Client 11.1 security services, see the Release Notes.

Applies to

SYC and SYD Sybase Systems 10.x and 11.x

Syntax

Sec_Mechanism = '*mechanism_name*'

Parameter	Description
mechanism_name	The security mechanism name you want to use to establish a connection The security mechanism name is case-sensitive. You must specify it *exactly as it appears* in the Open Client/Open Server Configuration utility

Default value

The default value for Sec_Mechanism is the security mechanism name currently specified as active in the Open Client/Open Server Configuration utility. If there is no security mechanism specified, no security service is used to establish the connection.

Usage

When to use Set Sec_Mechanism to use a security mechanism specified in the Open Client/Open Server Configuration utility *other than* the default (active) security mechanism.

FOR INFO For instructions on using the Open Client/Open Server Configuration utility, see your Sybase Open Client/Server configuration guide.

Set Release DBParm to '11' For Sec_Mechanism to take effect, you *must* also set the Release DBParm parameter to '11' to specify that your application should use Sybase Open Client Client-Library (CT-Lib) 11.x behavior. (The default for Release is to use CT-Lib 10.x behavior, which does not support security services.)

Requirements for use To use Sec_Mechanism or any other DBParm parameter supporting Open Client 11.1 security services, you must meet certain requirements for using security services in your PowerBuilder application.

FOR INFO For details, see "Requirements for using Open Client security services" on page 263.

Corresponding CT-Lib connection property Specifying a value for Sec_Mechanism sets the corresponding Sybase CT-Lib 11.1 connection property named CS_SEC_MECHANISM.

Examples

To specify KERBEROS as your security mechanism name:

- **Database profile** Type the following in the Security Mechanism box on the Security tab in the Database Profile Setup dialog box:

```
KERBEROS
```

- **PowerBuilder application script** Type the following in a PowerBuilder script:

```
SQLCA.DBParm = "Sec_Mechanism = 'KERBEROS'"
```

Using the examples in a PowerBuilder application script If you specify DBParm parameters in your database profile, the PowerScript 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.

See also

Release (Sybase System 10 and System 11)
Sec_Channel_Bind
Sec_Cred_Timeout
Sec_Delegation
Sec_Keytab_File
Sec_Mutual_Auth
Sec_Network_Auth
Sec_Server_Principal
Sec_Sess_Timeout

Sec_Mutual_Auth

Description

When you access a Sybase Systems 10.x or 11.x database in PowerBuilder through Open Client 11.1 software, Sec_Mutual_Auth is one of several DBParm parameters that supports login authentication for network-based security services. (For other login authentication DBParms, see the See Also section.)

Sec_Mutual_Auth controls whether your connection's security mechanism performs mutual authentication. When Sec_Mutual_Auth is set to 1, the server must prove its identity to the client before connecting by sending a credential token containing the server's principal name and proof that this name is authentic.

When Sec_Mutual_Auth is set to 0 (the default), no mutual authentication is performed.

You must specify a value for Sec_Mutual_Auth *before* connecting to the database in PowerBuilder.

Using third-party security mechanisms

FOR INFO For information about the third-party security mechanisms and operating system platforms that Powersoft has tested with Open Client 11.1 security services, see the Release Notes.

Applies to

SYC and SYD Sybase Systems 10.x and 11.x

Syntax

Sec_Mutual_Auth = *value*

Parameter	Description
value	Specifies whether your connection's security mechanism performs mutual authentication. Values are: ♦ **0** (Default) Do *not* perform mutual authentication. You can also specify **'No'** or **'False'** to set this value ♦ **1** Perform mutual authentication. The server must prove its identity to the client before connecting by sending a credential token containing the server's principal name and proof that this name is authentic. You can also specify **'Yes'** or **'True'** to set this value

Default value

Sec_Mutual_Auth = 0

Usage

Set Release DBParm to '11' For Sec_Mutual_Auth to take effect, you *must* also set the Release DBParm parameter to '11' to specify that your application should use Sybase Open Client Client-Library (CT-Lib) 11.x behavior. (The default for Release is to use CT-Lib 10.x behavior, which does not support security services.)

Requirements for use To use Sec_Mutual_Auth or any other DBParm parameter supporting Open Client 11.1 security services, you must meet certain requirements for using security services in your PowerBuilder application.

FOR INFO For details, see "Requirements for using Open Client security services" on page 263.

Corresponding CT-Lib connection property Specifying a value for Sec_Mutual_Auth sets the corresponding Sybase CT-Lib 11.1 connection property named CS_SEC_MUTUALAUTH.

Examples

To specify that your connection's security mechanism performs mutual authentication:

- **Database profile** Select the Mutual Authentication checkbox on the Security tab in the Database Profile dialog.
- **PowerBuilder application script** Type the following in a PowerBuilder script:

```
SQLCA.DBParm = "Sec_Mutual_Auth = 1"
```

Using the examples in a PowerBuilder application script If you specify DBParm parameters in your database profile, the PowerScript 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.

See also

Release (Sybase System 10 and System 11)
Sec_Channel_Bind
Sec_Cred_Timeout
Sec_Delegation
Sec_Keytab_File
Sec_Mechanism
Sec_Network_Auth
Sec_Server_Principal
Sec_Sess_Timeout

Sec_Network_Auth

Description

When you access a Sybase Systems 10.x or 11.x database in PowerBuilder through Open Client 11.1 software, Sec_Network_Auth is one of several DBParm parameters that supports login authentication for network-based security services. (For other login authentication DBParms, see the See Also section.)

Sec_Network_Auth controls whether your connection uses network-based login authentication. When Sec_Network_Auth is set to 1, your connection uses network-based authentication when connecting to a secure SQL Server. **Network-based authentication** means that the security mechanism—not the application—confirms that the specified user name represents the authenticated user running the application.

Since the security mechanism rather than the application authenticates your user name (login ID), you need *not* supply a login password for authentication purposes in the database profile or PowerBuilder application script if Sec_Network_Auth is set to 1.

When Sec_Network_Auth is set to 0 (the default), your connection does not use network-based login authentication to connect to the server.

You must specify a value for Sec_Network_Auth *before* connecting to the database in PowerBuilder.

Using third-party security mechanisms

FOR INFO For information about the third-party security mechanisms and operating system platforms that Powersoft has tested with Open Client 11.1 security services, see the Release Notes.

Applies to

SYC and SYD Sybase Systems 10.x and 11.x

Syntax

Sec_Network_Auth = *value*

Parameter	Description
value	Specifies whether your connection uses network-based login authentication when connecting to a secure SQL Server. Values are: ♦ **0** (Default) Do *not* use network-based login authentication when connecting to the server. You can also specify **'No'** or **'False'** to set this value ♦ **1** Use network-based login authentication when connecting to the server. Since the security mechanism rather than the application authenticates your user name (login ID), you need *not* supply a login password for authentication purposes in the database profile or PowerBuilder application script. You can also specify **'Yes'** or **'True'** to set this value

Default value

Sec_Network_Auth = 0

Usage

When to use Setting Sec_Network_Auth to 1 to enable network-based login authentication provides three important benefits for PowerBuilder users. All of these benefits result from not having to specify a login password in the database profile or PowerBuilder application script to authenticate the login ID when Sec_Network_Auth is set to 1.

- **Password not stored in initialization file** Since you do not specify a login password, it is not stored in the PowerBuilder initialization file. (Typically, PowerBuilder stores the unencrypted password in the initialization file. This can be a security risk if the initialization file is accessible from the network.)
- **Password not transmitted across network** Since you do not specify a login password, it is not transmitted across the network to SQL Server.
- **Same user ID and password for different servers** You can use the same network user ID and password to connect to many different SQL Server database servers. You can change your password for the network security mechanism and have this change apply on all servers to which your application connects.

Set Release DBParm to '11' For Sec_Network_Auth to take effect, you *must* also set the Release DBParm parameter to '11' to specify that your application should use Sybase Open Client Client-Library (CT-Lib) 11.x behavior. (The default for Release is to use CT-Lib 10.x behavior, which does not support security services.)

Requirements for use To use Sec_Network_Auth or any other DBParm parameter supporting Open Client 11.1 security services, you must meet certain requirements for using security services in your PowerBuilder application.

FOR INFO For details, see "Requirements for using Open Client security services" on page 263.

Corresponding CT-Lib connection property Specifying a value for Sec_Network_Auth sets the corresponding Sybase CT-Lib 11.1 connection property named CS_SEC_NETWORKAUTH.

Examples

To specify that your connection uses network-based login authentication when connecting to the server:

- **Database profile** Select the Network Based Authentication checkbox on the Security tab in the Database Profile Setup dialog box.

- **PowerBuilder application script** Type the following in a PowerBuilder script:

```
SQLCA.DBParm = "Sec_Network_Auth = 1"
```

Using the examples in a PowerBuilder application script If you specify DBParm parameters in your database profile, the PowerScript 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.

See also

Release (Sybase System 10 and System 11)
Sec_Channel_Bind
Sec_Cred_Timeout
Sec_Delegation
Sec_Keytab_File
Sec_Mechanism
Sec_Mutual_Auth
Sec_Server_Principal
Sec_Sess_Timeout

Sec_Replay_Detection

Description

When you access a Sybase Systems 10.x or 11.x database in PowerBuilder through Open Client 11.1 software, Sec_Replay_Detection is one of several DBParm parameters that supports per-packet security for network-based security services. (For other per-packet security DBParms, see the See Also section.)

Sec_Replay_Detection controls whether your connection's security mechanism can detect and reject unauthorized attempts to capture and replay transmitted data. When Sec_Replay_Detection is set to 1, the security mechanism detects and subsequently rejects any unauthorized attempts by third parties to capture packets sent to the server and repeat (replay) the commands in the packets at a later time.

When Sec_Replay_Detection is set to 0 (the default), the security mechanism cannot detect unauthorized attempts to capture and replay data.

You must specify a value for Sec_Replay_Detection *before* connecting to the database in PowerBuilder.

Using third-party security mechanisms

FOR INFO For information about the third-party security mechanisms and operating system platforms that Powersoft has tested with Open Client 11.1 security services, see the Release Notes.

Applies to

SYC and SYD Sybase Systems 10.x and 11.x

Syntax

Sec_Replay_Detection = *value*

Parameter	Description
value	Specifies whether your connection's security mechanism can detect and reject unauthorized attempts to capture and replay transmitted data. Values are: ♦ **0** (Default) Prohibit your security mechanism from detecting unauthorized attempts to capture and replay transmitted data. You can also specify **'No'** or **'False'** to set this value ♦ **1** Allow your security mechanism to detect and reject unauthorized attempts to capture and replay transmitted data. You can also specify **'Yes'** or **'True'** to set this value

Default value

Sec_Replay_Detection = 0

Usage

When to use In a nonsecure network, unauthorized third parties may capture the commands sent to a server in order to repeat (replay) these commands at a later date. For example, if packets are sent from the client to the server in the order P1, P2, P3 and the server receives the packets in the order P1, P2, P2, this is considered an attempt to replay the data.

Setting Sec_Replay_Detection to 1 ensures that your security mechanism can detect and subsequently reject all such unauthorized attempts to capture and replay data transmitted over the network.

Set Release DBParm to '11' For Sec_Replay_Detection to take effect, you *must* also set the Release DBParm parameter to '11' to specify that your application should use Sybase Open Client Client-Library (CT-Lib) 11.x behavior. (The default for Release is to use CT-Lib 10.x behavior, which does not support security services.)

Requirements for use To use Sec_Replay_Detection or any other DBParm parameter supporting Open Client 11.1 security services, you must meet certain requirements for using security services in your PowerBuilder application.

FOR INFO For details, see "Requirements for using Open Client security services" on page 263.

Corresponding CT-Lib connection property Specifying a value for Sec_Replay_Detection sets the corresponding Sybase CT-Lib 11.1 connection property named CS_SEC_DETECTREPLAY.

Examples

To allow your security mechanism to detect and reject unauthorized attempts to capture and replay transmitted data:

- **Database profile** Select the Detect Replayed Commands checkbox on the Security tab in the Database Profile Setup dialog box.
- **PowerBuilder application script** Type the following in a PowerBuilder script:

```
SQLCA.DBParm = "Sec_Replay_Detection = 1"
```

Using the examples in a PowerBuilder application script If you specify DBParm parameters in your database profile, the PowerScript 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.

See also

Release (Sybase System 10 and System 11)
Sec_Confidential
Sec_Data_Integrity
Sec_Data_Origin
Sec_Seq_Detection

Sec_Seq_Detection

Description

When you access a Sybase Systems 10.x or 11.x database in PowerBuilder through Open Client 11.1 software, Sec_Seq_Detection is one of several DBParm parameters that supports per-packet security for network-based security services. (For other per-packet security DBParms, see the See Also section.)

Sec_Seq_Detection controls whether your connection's security mechanism can detect and reject transmitted packets that arrive at the server in a different order than they were sent from the client. When Sec_Seq_Detection is set to 1, the security mechanism detects and rejects packets that arrive at the server out-of-sequence.

When Sec_Seq_Detection is set to 0 (the default), the security mechanism cannot detect packets that arrive at the server out-of-sequence.

You must specify a value for Sec_Seq_Detection *before* connecting to the database in PowerBuilder.

Using third-party security mechanisms

FOR INFO For information about the third-party security mechanisms and operating system platforms that Powersoft has tested with Open Client 11.1 security services, see the Release Notes.

Applies to

SYC and SYD Sybase Systems 10.x and 11.x

Syntax

Sec_Seq_Detection = *value*

Parameter	Description
value	Specifies whether your connection's security mechanism can detect and reject packets that arrive at the server in a different order than they were sent from the client. Values are: ♦ **0** (Default) Prohibit your security mechanism from detecting packets that arrive at the server out-of-sequence. You can also specify **'No'** or **'False'** to set this value ♦ **1** Allow your security mechanism to detect and reject packets that arrive at the server out-of-sequence. You can also specify **'Yes'** or **'True'** to set this value

Default value

Sec_Seq_Detection = 0

Usage

When to use When transmitting data over a network, commands sent to a server may arrive out-of-sequence. For example, if packets are sent from the client to the server in the order P1, P2, P3 and the server receives the packets in the order P1, P3, P2, this is considered an out-of-sequence error.

Setting Sec_Seq_Detection to 1 ensures that your security mechanism can detect and subsequently reject packets that arrive at the server out-of-sequence.

Set Release DBParm to '11' For Sec_Seq_Detection to take effect, you *must* also set the Release DBParm parameter to '11' to specify that your application should use Sybase Open Client Client-Library (CT-Lib) 11.x behavior. (The default for Release is to use CT-Lib 10.x behavior, which does not support security services.)

Requirements for use To use Sec_Seq_Detection or any other DBParm parameter supporting Open Client 11.1 security services, you must meet certain requirements for using security services in your PowerBuilder application.

FOR INFO For details, see "Requirements for using Open Client security services" on page 263.

Corresponding CT-Lib connection property Specifying a value for Sec_Seq_Detection sets the corresponding Sybase CT-Lib 11.1 connection property named CS_SEC_DETECTSEQ.

Examples

To allow your security mechanism to detect and reject packets that arrive at the server out-of-sequence:

- **Database profile** Select the Detect Sequence Errors checkbox on the Security tab in the Database Profile Setup dialog box.
- **PowerBuilder application script** Type the following in a PowerBuilder script:

```
SQLCA.DBParm = "Sec_Seq_Detection = 1"
```

Using the examples in a PowerBuilder application script If you specify DBParm parameters in your database profile, the PowerScript 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.

See also

Release (Sybase System 10 and System 11)
Sec_Confidential
Sec_Data_Integrity
Sec_Data_Origin
Sec_Replay_Detection

Sec_Server_Principal

Description

When you access a Sybase Systems 10.x or 11.x database in PowerBuilder through Open Client 11.1 software, Sec_Server_Principal is one of several DBParm parameters that supports login authentication for network-based security services. (For other login authentication DBParms, see the See Also section.)

Sec_Server_Principal specifies the principal name of the server that you want to access. The **server principal name** is the name by which your security mechanism identifies each server.

If the server name (specified in the database profile or PowerBuilder application script) is *different* from the server principal name for the server you want to access, you *must* set Sec_Server_Principal to the correct server principal name in order to connect.

You must specify a value for Sec_Server_Principal *before* connecting to the database in PowerBuilder.

Using third-party security mechanisms

FOR INFO For information about the third-party security mechanisms and operating system platforms that Powersoft has tested with Open Client 11.1 security services, see the Release Notes.

Applies to

SYC and SYD Sybase Systems 10.x and 11.x

Syntax

Sec_Server_Principal = '*server_principal_name*'

Parameter	Description
server_principal_name	Specifies the principal name of the server you want to access

Default value

None

If you do not specify a value, the security mechanism uses the server's directory entry name, which is the same as the server name specified in the database profile or PowerBuilder application script.

Usage

When to use When using Open Client 11.1 security services with PowerBuilder, the server's directory entry name (which you specify as the server name in the database profile or PowerBuilder application script) may differ from the server principal name. In this case, you *must* set Sec_Server_Principal to the correct server principal name so the security mechanism can identify the server you want to access.

Set Release DBParm to '11' For Sec_Server_Principal to take effect, you *must* also set the Release DBParm parameter to '11' to specify that your application should use Sybase Open Client Client-Library (CT-Lib) 11.x behavior. (The default for Release is to use CT-Lib 10.x behavior, which does not support security services.)

Requirements for use To use Sec_Server_Principal or any other DBParm parameter supporting Open Client 11.1 security services, you must meet certain requirements for using security services in your PowerBuilder application.

FOR INFO For details, see "Requirements for using Open Client security services" on page 263.

Corresponding CT-Lib connection property Specifying a value for Sec_Server_Principal sets the corresponding Sybase CT-Lib 11.1 connection property named CS_SEC_SERVERPRINCIPAL.

Examples

To specify SYS11NT as the principal name of the server you want to access:

- **Database profile** Type the following in the Server Principal Name box on the Security tab in the Database Profile Setup dialog box:

```
SYS11NT
```

- **PowerBuilder application script** Type the following in a PowerBuilder script:

```
SQLCA.DBParm = "Sec_Server_Principal = 'SYS11NT'"
```

Using the examples in a PowerBuilder application script If you specify DBParm parameters in your database profile, the PowerScript 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.

See also

Release (Sybase System 10 and System 11)
Sec_Channel_Bind
Sec_Cred_Timeout
Sec_Delegation
Sec_Keytab_File
Sec_Mechanism

Sec_Mutual_Auth
Sec_Network_Auth
Sec_Sess_Timeout

Sec_Sess_Timeout

Description

When you access a Sybase Systems 10.x or 11.x database in PowerBuilder through Open Client 11.1 software, Sec_Sess_Timeout is one of several DBParm parameters that supports login authentication for network-based security services. (For other login authentication DBParms, see the See Also section.)

Some security mechanisms allow applications to set session timeout values for connections using network-based login authentication. For these connections, Sec_Sess_Timeout specifies the number of seconds remaining before a session expires. The session timeout period begins when the connection is opened.

By default, Sec_Sess_Timeout specifies that there is no session timeout limit; the session will not expire.

You must specify a value for Sec_Sess_Timeout *before* connecting to the database in PowerBuilder.

Using third-party security mechanisms

FOR INFO For information about the third-party security mechanisms and operating system platforms that Powersoft has tested with Open Client 11.1 security services, see the Release Notes.

Applies to

SYC and SYD Sybase Systems 10.x and 11.x

Syntax

Sec_Sess_Timeout = *value*

Parameter	Description
value	Specifies the number of seconds remaining before a session expires. You can also specify **'no_limit'** (the default) to indicate that the session will not expire A session timeout value set by the security system's administrator supersedes any value you specify for Sec_Sess_Timeout

Default value

Sec_Sess_Timeout = 'no_limit'

Usage

CyberSafe Kerberos If your security mechanism is CyberSafe Kerberos, Sec_Sess_Timeout cannot override the installation default value set for session timeout.

Set Release DBParm to '11' For Sec_Sess_Timeout to take effect, you *must* also set the Release DBParm parameter to '11' to specify that your application should use Sybase Open Client Client-Library (CT-Lib) 11.x behavior. (The default for Release is to use CT-Lib 10.x behavior, which does not support security services.)

Requirements for use To use Sec_Sess_Timeout or any other DBParm parameter supporting Open Client 11.1 security services, you must meet certain requirements for using security services in your PowerBuilder application.

FOR INFO For details, see "Requirements for using Open Client security services" on page 263.

Corresponding CT-Lib connection property Specifying a value for Sec_Sess_Timeout sets the corresponding Sybase CT-Lib 11.1 connection property named CS_SEC_SESSTIMEOUT.

Examples

To specify that there are 14,400 seconds (4 hours) remaining before a session expires:

- **Database profile** Type the following in the Session Timeout box on the Security tab in the Database Profile Setup dialog box:

```
14400
```

- **PowerBuilder application script** Type the following in a PowerBuilder script:

```
SQLCA.DBParm = "Sec_Sess_Timeout = 14400"
```

Using the examples in a PowerBuilder application script If you specify DBParm parameters in your database profile, the PowerScript 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.

See also

Release (Sybase System 10 and System 11)
Sec_Channel_Bind
Sec_Cred_Timeout
Sec_Delegation
Sec_Keytab_File
Sec_Mechanism
Sec_Mutual_Auth
Sec_Network_Auth

Sec_Server_Principal

Secure

Description

The Secure DBParm parameter specifies whether you want to use Windows NT integrated login security and a secure (trusted) connection when connecting to a Microsoft SQL Server 6.x database server in PowerBuilder.

By default, PowerBuilder uses SQL Server 6.x standard security and a nontrusted connection when accessing the database. If you set Secure to 1, PowerBuilder uses Windows NT integrated security instead of standard security to establish the connection, regardless of the login security currently in use on the server. With a trusted connection, any login ID or password you supply is ignored.

Applies to

MSS Microsoft SQL Server 6.x

Syntax

Secure = *value*

Parameter	Description
value	Specifies whether you want to use Windows NT integrated login security and a secure (trusted) connection when connecting to Microsoft SQL Server 6.x in PowerBuilder. Values are: ♦ **0** (Default) Use SQL Server 6.x standard login security and nontrusted connections. You can also specify **'No'** or **'False'** to set this value ♦ **1** Use Windows NT integrated login security and trusted connections, assuming this has been set up at your site. You can also specify **'Yes'** or **'True'** to set this value

Default value

Secure = 0

Usage

About Microsoft SQL Server 6.x security To use the Secure DBParm parameter efficiently, it is helpful to understand Microsoft SQL Server 6.x security mechanisms. SQL Server 6.x provides three server login security modes:

- **Integrated security** Uses Windows NT authentication mechanisms to validate SQL Server logins for all connections. This allows a network user with the necessary privileges to log in to SQL Server without supplying a separate login ID or password. Integrated security requires the use of **trusted connections** over network protocols that support authenticated connections between clients and servers.
- **Standard security** Uses the SQL Server login validation mechanism for all connections. Standard security uses nontrusted connections.
- **Mixed security** Uses a combination of integrated security and standard security to validate login requests.

Requirements for using the Secure parameter For the Secure DBParm to take effect in PowerBuilder, you or your database administrator must first grant SQL Server administrator or user privilege to the appropriate Windows NT groups or users, allowing them to use trusted connections.

FOR INFO For information on setting up integrated login security and trusted connections at your site, see the administrator's guide in your Microsoft SQL Server 6.x documentation.

Examples

To specify that you want to use Windows NT integrated login security and trusted connections when connecting to Microsoft SQL Server 6.x in PowerBuilder:

- **Database profile** Select the Integrated Security checkbox on the Network tab in the Database Profile Setup dialog box.
- **PowerBuilder application script** To specify this statement in a PowerBuilder application script, type the following:

```
SQLCA.dbParm = "Secure = 1"
```

Using the examples in a PowerBuilder application script If you specify DBParm parameters in your database profile, the PowerScript 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.

SQLCache

Description

The SQLCache parameter lets you specify the number of SQL statements that PowerBuilder should cache. By default, SQLCache is set to 0, indicating an empty SQL cache. PowerBuilder caches:

- SQL statements generated by a DataWindow object
- Embedded SQL statements

PowerBuilder caches SQL statements for Oracle databases and for ODBC data sources if the backend DBMS supports it.

Applies to

ODBC (if driver and backend DBMS support this feature)
Oracle (all interfaces)

Syntax

The syntax for setting SQLCache depends on whether you are accessing an ODBC data source or an Oracle database.

ODBC syntax Use the following syntax to set the SQLCache parameter for an ODBC data source:

SQLCache = *value*

Parameter	Description
value	The number of cursors you want to open in a script, plus the number of DataWindow-generated SELECT statements with retrieval arguments (Default = 0)

Oracle syntax Use the following syntax to set the SQLCache parameter for an Oracle database:

SQLCache = *value*

where *value* is a number that is less than the maximum number of cursors that can be open on the client machine. To determine this number, use the following formula:

value < = *open_cursors* - *reserved* - *declare_cursor_space*

Parameter	Description
open_cursors	The Oracle server setting (OPEN_CURSORS) that specifies the maximum number of cursors a single process can have open at one time For Oracle 7.x database servers, *open_cursors* is a number between 1 and the operating system limit (Default = 50)

Parameter	Description
reserved	The number of reserved cursors. You should reserve five cursors for use by any of the Oracle database interfaces
declare_cursor_space	The maximum number of cursors you expect to open per Oracle connection in PowerBuilder

Default value

SQLCache = 0

Usage

Maintaining statements in the cache Statements in the SQL cache are maintained on a least-recently-used (**LRU**) basis. In other words, if a statement must be removed from the cache to make room for another statement, PowerBuilder removes the statement that was least recently executed.

SQLCache and bind variables Caching SQL statements that you execute frequently will improve their performance. Statements with bind variables are often the most frequently used. In fact, if your DBMS does not support bind variables, caching statements is of limited value.

Setting DisableBind to use cached statements In order to use cached statements, make sure the DisableBind DBParm parameter is set to 0 (the default). This enables the binding of input variables to SQL statements in PowerBuilder.

FOR INFO For more about using bind variables, see DisableBind on page 444.

What happens The first time you execute a SQL statement containing bind variables, PowerBuilder does the following in this order:

1 Parses the statement.

2 For SQL SELECT statements, calls the appropriate database function to get a description of the result set.

3 Allocates memory buffers for the bind variables.

4 Binds the allocated memory buffers to the parsed statement.

When you cache this SQL statement, PowerBuilder stores the parsed statement, result set description, and memory buffer allocation and binding in the SQL cache. The next time you execute this statement, PowerBuilder finds it in the cache and avoids the overhead of repeating these steps.

If PowerBuilder finds an exact match for this statement in the SQL cache, it simply copies the new values supplied for the bind variables to the pre-allocated memory buffers and executes the statement. This is much faster than having to process the statement from scratch.

Determining the size of your SQL cache To determine an appropriate size for your SQL cache, you can check the value of the SQLReturnData property of the transaction object.

When you disconnect from the database, the number of hits, misses, and entries in the SQL cache is stored in SQLReturnData as follows:

- **Hits** The number of times PowerBuilder found a matching statement in the SQL cache
- **Misses** The number of times PowerBuilder did not find a matching statement in the cache
- **Entries** The total number of statements in the SQL cache, which is determined by your SQLCache setting

Examples

To set the SQL cache size to 25 statements:

- **Database profile** Type the following in the Number of SQL Statements Cached box on the Transaction tab in the Database Profile Setup dialog box:

```
25
```

- **PowerBuilder application script** Type the following in a PowerBuilder application script:

```
SQLCA.dbParm = "SQLCache = 25"
```

Using the examples in a PowerBuilder application script If you specify DBParm parameters in your database profile, the PowerScript 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.

See also

DisableBind

SQLQualifiers

Description

Sets the level of qualification for identifiers (table and column names) in SQL statements when you connect to a DB2 database through the Sybase Net-Gateway for DB2 interface. This affects behavior in DataWindows and reports.

When PowerBuilder **qualifies** a table or column name, it prefixes it with the name of the owner. For example, if a user named Fran owns a tables named Sales, the qualified table name is Fran.Sales.

Applies to

NET Sybase Net-Gateway for DB2 interface

Syntax

SQLQualifiers = *value*

Parameter	Description
value	Sets the level of qualification for identifiers in SQL statements when you connect to a DB2 database through the Sybase Net-Gateway for DB2 interface. Values are: ♦ **0** (Default) Do not qualify identifiers with owner names in SQL statements ♦ **1** Qualify identifiers with owner names in SQL statements

Default value

SQLQualifiers = 0

Usage

Requirements for using SQLQualifiers You can set the SQLQualifiers parameter if both of the following are true:

- DB2 security is off.
- You assume that unqualified tables are owned by the default DB2 table owner.

When PowerBuilder qualifies identifiers If the name of the table owner is the same as the person logged on to the DB2 database, PowerBuilder does not qualify identifiers with owner names in the SQL statements it generates. If you set the SQLQualifiers parameter to 1, PowerBuilder qualifies identifiers with an owner name in SQL statements.

Examples

To specify that you want PowerBuilder to qualify identifiers with owner names in SQL statements:

- **Database profile** Select the Qualify Identifiers with Owner Names checkbox on the Syntax tab in the Database Profile Setup dialog box.
- **PowerBuilder application script** Type the following in a PowerBuilder application script:

```
SQLCA.dbParm = "SQLQualifiers = 1"
```

Using the examples in a PowerBuilder application script If you specify DBParm parameters in your database profile, the PowerScript 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.

StaticBind

Description

When you retrieve data from a database into a DataWindow or report, PowerBuilder does not get a result set description to validate the SELECT statement against the database server before retrieving the data. As a result, the retrieval should be faster, especially when you are accessing the database over a network. (This feature is also called **describeless retrieval**.)

If you want to override the default behavior and have PowerBuilder get a description of the result set before retrieving data, set the StaticBind DBParm parameter to 0 or No.

Applies to

MSS Microsoft SQL Server 6.x
ODBC (if driver and backend DBMS support this feature)
Oracle (all interfaces)
SYB and SYT SQL Server 4.x
MDI Sybase InformationConnect DB2 Gateway interface
NET Sybase Net-Gateway for DB2 interface
SYC and SYD Sybase Systems 10.x and 11.x

Syntax

StaticBind = *value*

Parameter	Description
value	A value specifying whether you want PowerBuilder to get a result set description before retrieving data from a database into a DataWindow or report. Values are: ♦ **0** Get a result set description before retrieving data. You can also specify **'No'** to set this value ♦ **1** (Default) Skip getting a result set description before retrieving data. You can also specify **'Yes'** to set this value

Default value

StaticBind = 1

Usage

Validation When StaticBind is set to 1 (the default), PowerBuilder does not validate the SELECT statement against the database server before retrieving data. It assumes that the result set matches the column format of the DataWindow or report into which it is being retrieved. If a mismatch occurs, PowerBuilder displays an error.

Troubleshooting tips Problems can occur in your application if the result set description obtained by the DataWindow or report is different from the current database description of the result set. This can occur for the following reasons:

- The database definition changes after you build the DataWindow or report
- You build the DataWindow or report while connected to one DBMS and then run it against a different DBMS

In previous releases, PowerBuilder always obtained a dynamic result set description before retrieving data into a DataWindow or reportand adjusted the bind information accordingly. Currently, PowerBuilder's default behavior is to *skip* getting a result set description before retrieving data. PowerBuilder therefore assumes that your bind information is correct and does not make adjustments before each retrieval.

To fix problems caused by conflicting result set descriptions, you can correct your DataWindow or report definition by doing either of the following:

- Export and edit your column definitions
- Force a recompile of the SQL statement in the Database painter's SQL Toolbox (see the *PowerBuilder User's Guide* for instructions)

If your DataWindow or report and DBMS result set descriptions do not match and you want to avoid errors, set StaticBind to 0 or No to specify that PowerBuilder should *always* get a result set description before retrieving data into a DataWindow or report.

Examples

To specify that you want PowerBuilder to get a result set description before retrieving data into a DataWindow or report:

- **Database profile** Clear the Static Bind checkbox on the Transaction tab in the Database Profile Setup dialog box.
- **PowerBuilder application script** To specify this statement in a PowerBuilder application script, type the following:

```
SQLCA.dbParm = "StaticBind = 0"
```

Using the examples in a PowerBuilder application script If you specify DBParm parameters in your database profile, the PowerScript 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.

SystemOwner

Description

Specifies the owner of the IBM DB2 system tables that you want PowerBuilder to use. PowerBuilder accesses the DB2 system tables to get information about the tables and columns in your database.

By default, the owner of the DB2 system tables is SYSIBM. When you use the SystemOwner parameter to specify a nondefault system owner, PowerBuilder uses the set of system tables belonging to this owner instead of the default system tables owned by SYSIBM.

Applies to

IBM
MDI Sybase InformationConnect DB2 Gateway interface
NET Sybase Net-Gateway for DB2 interface

Syntax

SystemOwner = '*owner_name*'

Parameter	Description
owner_name	Specifies the owner of the DB2 system tables that you want PowerBuilder to use (Default = SYSIBM)

Default value

SystemOwner = 'SYSIBM'

Usage

Using shadow catalogs If your site has a large DB2 system catalog, it may be useful to create local copies of the catalog tables and populate them with a subset of the information in the default system catalog. These local copies are sometimes called **shadow catalogs**.

You can then set the value of SystemOwner to the owner of the shadow catalogs. This tells PowerBuilder to access the smaller shadow catalogs instead of the larger default system tables, resulting in faster performance. However, you must make sure to keep the shadow catalogs synchronized with the default system catalog owned by SYSIBM.

FOR INFO For more about creating shadow catalogs, see your DB2 system administrator or check whether there is a FaxLine document that describes how to do it. Updated information about connectivity issues is available from the Powersoft FaxLine system and from Powersoft electronic services on CompuServe, FTP, BBS, and the World Wide Web.

Examples

To specify MYAPP as the owner of the system tables that you want PowerBuilder to use:

- **Database profile** Type the following in the System Owner box on the System tab in the Database Profile Setup dialog box:

```
MYAPP
```

- **PowerBuilder application script** Type the following in a PowerBuilder application script:

```
SQLCA.dbParm = "SystemOwner = 'MYAPP'"
```

Using the examples in a PowerBuilder application script If you specify DBParm parameters in your database profile, the PowerScript 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.

SystemProcs

Description

Specifies whether you want PowerBuilder to display both system stored procedures and user-defined stored procedures in the connected SQL Server database when you request a list of stored procedures.

By default, PowerBuilder displays both system and user-defined stored procedures in the connected database. If you set SystemProcs to 0 or No, only user-defined stored procedures are displayed.

Applies to

MSS Microsoft SQL Server 6.x
SYC and SYD Sybase Systems 10.x and 11.x

Syntax

SystemProcs = *value*

Parameter	Description
value	Specifies whether you want PowerBuilder to display both system stored procedures and user-defined stored procedures in the connected SQL Server database when you request a list of stored procedures. Values are: ♦ **0** Display only user-defined stored procedures. You can also specify **'No'** to set this value ♦ **1** (Default) Display both system stored procedures and user-defined stored procedures. You can also specify **'Yes'** to set this value

Default value

SystemProcs = 1

Usage

Setting SystemProcs to 0 or No speeds response time if you want to work only with user-defined stored procedures.

Examples

To specify that you want PowerBuilder to display only user-defined stored procedures in the connected SQL Server database when you request a list of stored procedures:

- **Database profile** Clear the Display System Stored Procedures checkbox on the System tab in the Database Profile Setup dialog box.
- **PowerBuilder application script** To specify this statement in a PowerBuilder application script, type the following:

```
SQLCA.dbParm = "SystemProcs = 0"
```

Using the examples in a PowerBuilder application script If you specify DBParm parameters in your database profile, the PowerScript 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.

TableCriteria (IBM, InformationConnect Gateway)

Description

Enables you to specify criteria (search conditions) to limit the list of tables and views that displays in the Select Tables list in PowerBuilder. Setting this parameter can be useful if you are working with a very large database in the PowerBuilder development environment.

PowerBuilder appends the search criteria you specify to the WHERE clause of the SQL SELECT statement it generates to create the table list. This SELECT statement has the following format:

```
SELECT column_list FROM SYSTABLES, SYSTABAUTH
    WHERE join_criteria
    AND table_authorizations
    AND tablecriteria_search_conditions
```

When to specify TableCriteria

You must specify the TableCriteria DBParm parameter *before* connecting to the database in PowerBuilder.

The TableCriteria DBParm parameter has no effect in a PowerBuilder application script.

Applies to

IBM
MDI Sybase InformationConnect DB2 Gateway interface

Syntax

search_conditions

Parameter	Description
search_conditions	Specifies the criteria you want to use to limit the tables and views displayed in the Select Tables list. The *search_conditions* string can be any valid DB2/MVS WHERE clause syntax For each literal value in the *search_conditions* string, type two consecutive single quotes, followed by the value, followed by two consecutive single quotes. PowerBuilder interprets these two consecutive single quotes as an escape quote

Default value

None

If you do not specify a value, the TableCriteria parameter is not used.

Examples

About these examples The following examples show different ways to set TableCriteria in a database profile to limit the Select Tables list when you are using the IBM or Sybase InformationConnect interface.

Type the statement exactly as shown in the Table Criteria box on the System tab in the Database Profile Setup dialog box.

Typing single quotes
The quotes in all of the following examples are single quotes. Do not type double quotes.

Example 1 This statement produces a list of tables belonging to either the QR or FS systems.

```
NAME like ''QR%'' OR NAME like ''FS%''
```

Example 2 This statement produces a list of tables created by the specified users.

```
CREATOR in (''FRANS'',''MIKEC'')
```

Example 3 This statement produces a list of accessible tables created in October 1997.

```
MONTH(CTIME)=10 AND YEAR(CTIME)=97
```

TableCriteria (ODBC)

Description

Enables you to specify criteria (search conditions) to limit the list of tables and views that displays in the Select Tables list in PowerBuilder. Setting this parameter can be useful if you are working with a very large database in the PowerBuilder development environment.

When to specify TableCriteria
You must specify the TableCriteria DBParm parameter *before* connecting to the database in PowerBuilder.

The TableCriteria DBParm parameter has no effect in a PowerBuilder application script.

Applies to

ODBC (if driver and backend DBMS support this feature)

Syntax

{*table_name_criteria*},{*table_owner_criteria*},{*table_qualifier_criteria*}, {*table_type_criteria*}

Type the TableCriteria string on a single line. The order of parameters within the TableCriteria string is important. Therefore, when you omit one or more leading parameters, you must include a comma to indicate the omission.

The TableCriteria string can include the following metacharacters:

- Percent (%) matches any sequence of zero or more characters.
- Underscore (_) matches any single character.

To use a metacharacter as a literal, precede it with the escape character for the ODBC data source you are accessing (see Example 3 below).

Parameter	Description
table_name_criteria	Specifies the names of tables to display
table_owner_criteria	Displays only those tables belonging to the specified *table_owner*
table_qualifier_criteria	Displays only those tables belonging to the specified *qualifier*
table_type_criteria	Specifies the type of table objects to display.You can specify one or more of the following types: ♦ **"TABLE"** ♦ **"VIEW"** ♦ **"SYSTEM TABLE"** ♦ **"ALIAS"** ♦ **"SYNONYM"** ♦ **"GLOBAL TEMPORARY"** ♦ **"LOCAL TEMPORARY"** ♦ A data source-specific type identifier (for information, see the documentation for your ODBC data source) For each table type in the *table_type_criteria* string, type two consecutive single quotes, followed by the table type, followed by two consecutive single quotes. Separate the types with commas

Default value

None

If you do not specify a value, the TableCriteria parameter is not used.

Examples

About these examples The following examples show different ways to set TableCriteria in a database profile to limit the Select Tables list when you are using the ODBC interface.

Type the statement exactly as shown in the Table Criteria box on the System tab in the Database Profile Setup dialog box.

Typing single quotes
The quotes in the following examples are single quotes. Do not type double quotes.

Example 1 This statement produces a list of tables and views owned by SDG whose name begins with cust.

```
cust%,SDG
```

Example 2 This statement produces a list of tables owned by SDG. The comma before SDG indicates that *table_name_criteria* was intentionally omitted.

```
,SDG
```

Example 3 This statement produces a list of tables owned by Dept_Marketing. To use the underscore character in Dept_Marketing as a literal, precede it with the escape character for your ODBC data source. In this example, the escape character is a backslash (\ ).

```
,Dept\_Marketing
```

Example 4 This statement produces a list of views owned by SDG.

```
,SDG,''VIEW''
```

TableCriteria (Oracle)

Description

Enables you to specify criteria (search conditions) to limit the list of tables and views that displays in the Select Tables list in PowerBuilder. Setting this parameter can be useful if you are working with a very large database in the PowerBuilder development environment.

When to specify TableCriteria
You must specify the TableCriteria DBParm parameter *before* connecting to the database in PowerBuilder.

The TableCriteria DBParm parameter has no effect in a PowerBuilder application script.

Applies to

Oracle (all interfaces)

Syntax

{*table_name_criteria*},{*table_owner_criteria*},{*table_qualifier_criteria*}

The order of parameters within the TableCriteria string is important. Therefore, when you omit one or more parameters, you must include a comma to indicate the omission.

The TableCriteria string can include the percent (%) metacharacter that matches any sequence of zero or more characters.

Parameter	Description
table_name_criteria	Specifies the names of tables to display. PowerBuilder does not validate the value you supply
table_owner_criteria	Displays only those tables belonging to the specified *table_owner*. PowerBuilder does not validate the value you supply
table_qualifier_criteria	Specifies the type of table objects to display.You can specify one or more of the following types, using only uppercase characters: ◆ **"TABLE"** ◆ **"VIEW"** ◆ **"SYNONYM"** For each table type in the *table_qualifier_criteria* string, type two consecutive single quotes, followed by the table type, followed by two consecutive single quotes. Separate the types with commas

Default value

If you do not specify a value for TableCriteria, all Oracle tables, views, and synonyms that you have permission to access display in the Select Tables list by default.

Examples

About these examples The following examples show different ways to set TableCriteria in a database profile to limit the Select Tables list when you are suing any of the Oracle interfaces.

Type the statement exactly as shown in the Table Criteria box on the System tab in the Database Profile Setup dialog box.

Typing single quotes
The quotes in the following examples are single quotes. Do not type double quotes.

Example 1 This statement produces a list of all tables and views (but no synonyms) owned by FMS whose name begins with emp. The comma after FMS indicates that *table_qualifier_criteria* was intentionally omitted.

```
emp%,FMS,
```

Example 2 This statement produces a list of tables only. The two commas before TABLE indicate that *table_name_criteria* and *table_owner_criteria* were intentionally omitted.

```
,,''TABLE''
```

Example 3 This statement produces a list of synonyms. Only the synonyms display in the list, not the tables and views themselves. If you specify a value for TableCriteria, synonyms display in the table list *only* if you explicitly specify the type SYNONYM in the TableCriteria statement.

```
,,''SYNONYM''
```

Example 4 This statement produces a list of all views and all synonyms for views.

```
,,''VIEW'',''SYNONYM''
```

Example 5 This statement produces a list of all views owned by FMS whose name begins with Prod, such as Product_Quantities_View. It also lists all synonyms for views owned by FMS whose name begins with Prod, such as Product_Quantities_View_Syn.

```
Prod%,FMS,''VIEW'',''SYNONYM''
```

Example 6 This statement (two commas) is an empty TableCriteria string. It produces a list of all tables and views, but no synonyms. It is equivalent to typing TABLECRITERIA=',,"TABLE","VIEW".

```
,,
```

TableCriteria (Sybase Net-Gateway, Sybase System 10 and System 11)

Description

Enables you to specify criteria (search conditions) to limit the list of tables and views that displays in the Select Tables list in PowerBuilder. Setting this parameter can be useful if you are working with a very large database in the PowerBuilder development environment.

These Powersoft database interfaces use stored procedures to create the table list:

- **Sybase Net-Gateway interface** Uses the sp_tables stored procedure.
- **Sybase Systems 10.x and 11.x interfaces** Uses the version of the sp_pb60table stored procedure installed by you or your database administrator.

  FOR INFO For information about which version of sp_pb60table to install when connecting to a System 10.x or 11.x database, see "Installing Powersoft stored procedures in SQL Server databases" on page 274.

The sp_tables and sp_pb60table stored procedures each take the same four optional arguments:

{*table_name*},{*table_owner*},{*table_qualifier*},{*table_type*}

PowerBuilder uses the TableCriteria DBParm parameter to supply the arguments to sp_tables or sp_pb60table and build the table list based on your search criteria.

When to specify TableCriteria

You must specify the TableCriteria DBParm parameter *before* connecting to the database in PowerBuilder.

The TableCriteria DBParm parameter has no effect in a PowerBuilder application script.

Applies to

NET Sybase Net-Gateway for DB2 interface
SYC and SYD Sybase Systems 10.x and 11.x

Syntax

{*table_name*},{*table_owner*},{*table_qualifier*},{"*table_type*"}

Type the TableCriteria string on a single line. The order of parameters in the TableCriteria string is important; when you omit one or more leading parameters, you must include a comma to indicate the omission.

Parameter	Description
table_name	Specifies the names of tables to display in the current database. You can use wildcard characters **Default for Net-Gateway interface** If you omit this value when connected through the Net-Gateway interface, PowerBuilder displays all tables that you have permission to access in the current database **Default for Sybase Systems 10.x and 11.x interfaces** If you omit this value when connected through the Sybase Systems 10.x and 11.x interfaces, PowerBuilder displays all tables in the current database
table_owner	Displays only those tables belonging to the specified *table_owner*. You can use wildcard characters If you omit this value, PowerBuilder displays all tables matching *table_name* that you have permission to access
table_qualifier	Not applicable for use with PowerBuilder
"*table_type*"	Specifies the type of table objects to display. You must enclose the entire *table_type* string in double quotes. You *cannot* use wildcard characters You can specify one or more of the following types. Within the *table_type* string, enclose each type in single quotes and separate the types with commas ♦ **'TABLE'** ♦ **'VIEW'** ♦ **'SYSTEM TABLE'** ♦ **'ALIAS'** ♦ **'SYNONYM'**

Default value

None

If you do not specify a value, the TableCriteria parameter is not used.

Examples

About these examples The following examples show different ways to set TableCriteria in a database profile to limit the Select Tables list when using the Sybase Net-Gateway interface or one of the Sybase Systems 10.x and 11.x interfaces.

Type the statement exactly as shown in the Table Criteria box on the System tab in the Database Profile Setup dialog box.

Example 1 This statement produces a list of tables in the Powersoft repository that have names beginning with pb (such as pbcatcol and pbcatedt). You do not need commas after the percent sign for the missing parameters because they are not leading parameters (*table_name* is the first parameter). All quotes are single quotes.

```
''pb%''
```

Example 2 This statement produces a list of tables owned by dbo. The leading comma indicates that *table_name* was intentionally omitted. All quotes are single quotes.

```
,''dbo''
```

Example 3 This statement produces a list of views only. The three leading commas indicate that *table_name*, *table_owner*, and *table_qualifier* were intentionally omitted. The entire *table_type* string (" 'VIEW' ") is enclosed in double quotes. Within the *table_type* string, the type itself ('VIEW') is enclosed in single quotes.

```
, , , " 'VIEW' "
```

ThreadSafe

Description

When you access an Oracle database server through the Oracle 7.3 database interface in PowerBuilder, ThreadSafe specifies whether your connection should take advantage of the Oracle 7.3 thread-safe client libraries.

By default, ThreadSafe is set to No to specify that your connection does not use the thread-safe client libraries. If you set ThreadSafe to Yes, your connection takes advantage of the thread-safe client libraries.

When to specify ThreadSafe
You must specify a value for ThreadSafe *before* connecting to the database in PowerBuilder.

Applies to

O73 Oracle Version 7.3 (on Windows and Macintosh)

Syntax

ThreadSafe = '*value*'

Parameter	Description
value	Specifies whether a connection through the Oracle 7.3 database interface uses the Oracle 7.3 thread-safe client libraries. Values are: ♦ **Yes** Your connection uses the Oracle 7.3 thread-safe client libraries. Use this setting when building distributed applications that require a multithreaded environment ♦ **No** (Default) Your connection does not use the Oracle 7.3 thread-safe client libraries. Use this setting when building nondistributed applications that require a single-threaded environment

Default value

ThreadSafe = 'No'

Usage

When to use Oracle 7.3 provides support for thread safety in its client libraries. When you are using the Oracle 7.3 database interface to build distributed applications in PowerBuilder that need a multithreaded environment, you should set the ThreadSafe DBParm parameter to Yes to use thread-safe client libraries. This prevents possible side effects among multiple threads of execution making calls to the Oracle 7.3 database server. Your application may incur a performance penalty when you use the thread-safe client libraries.

By default, the Oracle client software (and, consequently, PowerBuilder) assumes that you are building a nondistributed application in a single-threaded environment that does not need the thread-safe client libraries. This default ensures that single-threaded applications do not incur the performance penalty associated with using thread-safe libraries. Therefore, if you are building nondistributed applications, you can leave the ThreadSafe DBParm parameter set to No (the default).

Platform-specific differences Support for thread safety in the Oracle 7.3 client libraries is *not available* on every operating system platform.

FOR INFO For more information about support for thread safety on your platform, see your Oracle system-specific documentation.

On UNIX Due to limitations in the Oracle Call Interface (OCI) on the UNIX platform, ThreadSafe is *not available* when using the Oracle 7.3 database interface in PowerBuilder on UNIX.

Examples

To specify that your connection uses the Oracle 7.3 thread-safe client libraries:

- **Database profile** Select the Thread Safe checkbox on the Connection tab in the Database Profile Setup - Oracle 7.3 dialog box.
- **PowerBuilder application script** Type the following in a PowerBuilder script:

```
SQLCA.DBParm = "ThreadSafe = 'Yes'"
```

Using the examples in a PowerBuilder application script If you specify DBParm parameters in your database profile, the PowerScript 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.

Time

Description

When you update data in the DataWindow painter, PowerBuilder builds a SQL UPDATE statement in the background. If you are updating an ODBC data source or Oracle table, the Time DBParm parameter determines how PowerBuilder specifies a time data type when it builds the SQL UPDATE statement.

Applies to

ODBC (if driver and backend DBMS support this feature)
Oracle (all interfaces)

Syntax

The syntax you use to specify the Time DBParm differs slightly for ODBC and Oracle databases.

In the PowerBuilder development environment, the Database Profile Setup dialog box inserts special characters (quotes and backslashes) where needed, so you can specify just the Powersoft time format.

In a PowerBuilder application script, you must use the following syntax:

ODBC syntax If you are accessing an ODBC data source through the Powersoft ODBC interface, use the following syntax for Time. PowerBuilder parses the backslash followed by two single quotes (\") as a single quote when it builds the SQL UPDATE statement.

Time = '\"*Powersoft_time_format*\"'

Oracle syntax If you are accessing an Oracle database through one of the Powersoft Oracle database interfaces, use the following syntax for Time. PowerBuilder parses each set of four consecutive single quotes ("") as a single quote when it builds the SQL UPDATE statement.

Time = ' ""*Powersoft_time_format*"" '

Parameter	Description
' \"	**ODBC syntax** Type a single quote, followed by one space, followed by a backslash, followed by two single quotes. There is no space between the two single quotes and the beginning of the Powersoft time format
' """"	**Oracle syntax** Type a single quote, followed by one space, followed by four single quotes. There is no space between the four single quotes and the beginning of the Powersoft time format
Powersoft_time_format	The time format you want PowerBuilder to use when it builds a SQL UPDATE statement to update a data source in the DataWindow painter FOR INFO For more on Powersoft display formats, see the *PowerBuilder User's Guide*
\" '	**ODBC syntax** Type a backslash, followed by two single quotes, followed by one space, followed by a single quote. There is no space between the end of the Powersoft time format and the backslash
"""" '	**Oracle syntax** Type four single quotes, followed by one space, followed by a single quote. There is no space between the end of the Powersoft time format and the four single quotes

Default value

The default value for Time depends on the DBMS you are accessing:

DBMS	Date default value
ODBC	If no value is specified for the Time DBParm parameter, PowerBuilder looks for a time format in the section for your ODBC driver in the PBODB60 initialization file. If no time format is found in the PBODB60 initialization file, PowerBuilder uses the ODBC time format escape sequence
Oracle	The default Oracle date format FOR INFO For information, see your Oracle documentation

Examples

About these examples Assume you are updating a table named Workhours by setting the Start column to 08:30. This time is represented by the following Powersoft time format:

```
hh:mm
```

Example 1 (ODBC syntax) To specify that PowerBuilder should use this format for the time data type when it builds the SQL UPDATE statement:

- **Database profile** Type the following in the Time Format box on the Syntax tab in the Database Profile Setup dialog box:

```
hh:mm
```

- **PowerBuilder application script** Type the following in a PowerBuilder application script:

```
SQLCA.dbParm = "Time = ' \''hh:mm\'' '"
```

What happens PowerBuilder builds the following SQL UPDATE statement to update the table:

```
UPDATE WORKHOURS
SET START = '08:30'
```

Example 2 (Oracle syntax) To specify that PowerBuilder should use this format for the time data type when it builds the SQL UPDATE statement:

- **Database profile** Type the following in the Time Format box on the Syntax tab in the Database Profile Setup dialog box:

```
hh:mm
```

- **PowerBuilder application script** Type the following in a PowerBuilder application script:

```
SQLCA.dbParm = "Time = ' ''''hh:mm'''' '"
```

What happens PowerBuilder builds the following SQL UPDATE statement to update the table:

```
UPDATE WORKHOURS
SET START = '08:30'
```

Using the examples in a PowerBuilder application script If you specify DBParm parameters in your database profile, the PowerScript 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.

See also

Date
DateTime

CHAPTER 8

Database Preferences

About this chapter

This chapter describes the syntax and use of each connection-related database preference that you can set in PowerBuilder.

Contents

The database preferences are described in alphabetical order.

Database preferences and supported database interfaces

The following table lists each supported Powersoft database interface and the connection-related database preferences you can use with that interface in PowerBuilder. The preferences listed in the table pertain to the database connection, and not to the behavior of the Database painter itself.

Powersoft database interface	Database preferences
IBM database	Keep Connection Open Read Only Shared Database Profiles SQL Terminator Character Use Powersoft Repository
IN5 INFORMIX (through INFORMIX-NET 5)	AutoCommit Lock Read Only Shared Database Profiles SQL Terminator Character Use Powersoft Repository
IN7 INFORMIX (through INFORMIX ESQL 7.2)	AutoCommit Keep Connection Open Lock Read Only Shared Database Profiles SQL Terminator Character Use Powersoft Repository
MSS Microsoft SQL Server 6.x	AutoCommit Keep Connection Open Lock Read Only Shared Database Profiles SQL Terminator Character Use Powersoft Repository
ODBC **Using AutoCommit and Lock with ODBC** The AutoCommit and Lock database preferences are supported by the Powersoft ODBC interface only if *both* the ODBC driver you are using and the backend DBMS support the feature.	AutoCommit Keep Connection Open Lock Read Only Shared Database Profiles SQL Terminator Character Use Powersoft Repository

Powersoft database interface	Database preferences
OR7 Oracle 7	Keep Connection Open Read Only Shared Database Profiles SQL Terminator Character Use Powersoft Repository
O71 Oracle 7.1	Keep Connection Open Read Only Shared Database Profiles SQL Terminator Character Use Powersoft Repository
O72 Oracle 7.2	Keep Connection Open Read Only Shared Database Profiles SQL Terminator Character Use Powersoft Repository
O73 Oracle 7.3	Read Only Shared Database Profiles SQL Terminator Character Use Powersoft Repository
SYB SQL DB-Lib *and* SYT Sybase SQL Server DB-Lib	AutoCommit Keep Connection Open Read Only Shared Database Profiles SQL Terminator Character Use Powersoft Repository
MDI Sybase InformationConnect DB2 Gateway interface (formerly called MDI Database Gateway Interface for DB2)	AutoCommit Keep Connection Open Read Only Shared Database Profiles SQL Terminator Character Use Powersoft Repository
NET Sybase Net-Gateway for DB2 interface	AutoCommit Keep Connection Open Read Only Shared Database Profiles SQL Terminator Character Use Powersoft Repository
SYC Sybase Systems 10.x and 11.x *and* SYD Sybase Systems 10.x and 11.x distributed application interface on UNIX	AutoCommit Keep Connection Open Lock Read Only Shared Database Profiles SQL Terminator Character Use Powersoft Repository

AutoCommit

Description

For those DBMSs and database interfaces that support it, 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.

When to specify AutoCommit

In the PowerBuilder development environment, you must set AutoCommit before connecting to the database. AutoCommit takes effect only when the database connection occurs. Changes to AutoCommit after the connection occurs have no effect on the current connection.

In a PowerBuilder application script, you can reset the value of AutoCommit at any time. This lets you override the initial setting if necessary.

Applies to

IN5 and IN7 INFORMIX
MSS Microsoft SQL Server 6.x
ODBC (if driver and backend DBMS support this feature)
SYB and SYT SQL Server 4.x
MDI Sybase InformationConnect DB2 Gateway interface
NET Sybase Net-Gateway for DB2 interface
SYC and SYD Sybase Systems 10.x and 11.x

In a PowerBuilder application

For those DBMSs and database interfaces that support it, you can set AutoCommit in a script as a property of the transaction object. The following syntax assumes you are using the default transaction object SQLCA, but you can also define your own transaction object.

SQLCA.AutoCommit = *value*

Parameter	Description
value	Specifies whether PowerBuilder issues SQL statements outside or inside the scope of a transaction. Values are: ♦ **True** PowerBuilder issues SQL statements *outside the scope of a transaction*. That is, the statements are not part of a logical unit of work (LUW). If the SQL statement is successful, the DBMS updates the database immediately as if a COMMIT statement had been issued ♦ **False** (Default) PowerBuilder issues SQL statements *inside the scope of a transaction*. PowerBuilder issues a BEGIN TRANSACTION statement at the start of the connection. In addition, PowerBuilder issues another BEGIN TRANSACTION statement after each COMMIT or ROLLBACK statement is issued

In development environment

Select or clear the AutoCommit Mode checkbox on the Connection tab in the Database Profile Setup dialog box, as follows:

- **Select the checkbox** Sets AutoCommit to True for this connection.
- **Clear the checkbox** (Default) Sets AutoCommit to False for this connection.

FOR INFO For instructions, see "Setting AutoCommit and Lock in the database profile" on page 341.

Default value

AutoCommit = False

Usage

Transactions A **transaction** is one or more SQL statements that forms a **logical unit of work (LUW)**. Within a transaction, all SQL statements must succeed or fail as one logical entity. Changes are made to the database only if all statements in the transaction succeed and a COMMIT is issued. If one or more statements fail, you must issue a ROLLBACK to undo the changes. This ensures the integrity and security of data in your database.

Executing SQL DDL statements Some DBMSs require you to execute certain SQL statements outside the scope of a transaction. For example, when connected to a SQL Server database, you must execute SQL Data Definition Language (DDL) statements such as CREATE TABLE and DROP TABLE outside a transaction. There are two reasons for this:

- It ensures that the structure of your database cannot change during a transaction.
- It improves database performance, because DDL statements are costly operations to recover.

Therefore, to execute DDL statements or stored procedures containing DDL statements in a SQL Server database, you must set AutoCommit to True to issue the DDL statements outside the scope of a transaction. You should, however, set AutoCommit back to False immediately after executing the DDL statements.

When you change the value of AutoCommit from False to True, PowerBuilder issues a COMMIT statement by default.

Caution
When you set AutoCommit to True, you cannot roll back database changes. Therefore, use care when changing the setting of AutoCommit.

Using EXECUTE IMMEDIATE When AutoCommit is set to True, you can use the EXECUTE IMMEDIATE dynamic SQL statement to issue BEGIN TRANSACTION, COMMIT, ROLLBACK, and other SQL statements to control your own transaction processing.

FOR INFO For information about using the EXECUTE IMMEDIATE statement, see the *PowerScript Reference*.

Sybase InformationConnect DB2 Gateway When you connect to a DB2 database in PowerBuilder through the Sybase InformationConnect DB2 Gateway interface, the effect of setting the AutoCommit preference depends on how the gateway is configured at your site:

Gateway configuration	Effect of setting AutoCommit
Long transactions	Has no effect
Short transactions	**False** Causes PowerBuilder to change the gateway configuration to support long transactions **True** Causes PowerBuilder to leave the gateway configured for short transactions

Sybase Net-Gateway for DB2 Version 3.0 or higher required When you connect to a DB2 database in PowerBuilder through the Sybase Net-Gateway interface, AutoCommit is supported only if you are using Version 3.0 or higher of the Sybase Net-Gateway product.

Examples

To set AutoCommit to True and tell PowerBuilder to issue SQL statements outside the scope of a transaction:

- **Development environment** Select the AutoCommit Mode checkbox on the Connection tab in the Database Profile Setup dialog box.

- **PowerBuilder application script** Type the following in a script:

```
SQLCA.AutoCommit = True
```

Using the examples in a PowerBuilder application script If you specify AutoCommit Mode in your database profile, the PowerScript 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.

Keep Connection Open

Description

By default, PowerBuilder opens a database connection the first time you open a painter requiring a connection, and stays connected throughout the session until you exit.

When you connect to a database in the PowerBuilder development environment without using a database profile, you can set the Keep Connection Open database preference to specify when PowerBuilder closes the database connection.

PowerBuilder only
Keep Connection Open applies only when connecting to a database in the PowerBuilder development environment without using a database profile. The setting of Keep Connection Open has no effect in InfoMaker or when you use a database profile to connect in PowerBuilder.

Applies to

All Powersoft database interfaces (only in the PowerBuilder development environment)

In PowerBuilder application

You *cannot* set the Keep Connection Open database preference in a PowerBuilder application script.

In development environment

In the Database painter, select or clear the Keep Connection Open checkbox in the Database Preferences property sheet as follows:

- **Select the checkbox** (Default) Stays connected to the database throughout your PowerBuilder session and closes the connection when you exit
- **Clear the checkbox** Opens the database connection when a painter requires it and closes the connection when you close a painter or finish compiling a script

Default value

The Keep Connection Open checkbox in the Database Preferences property sheet is selected by default.

Usage

Requirements for using Keep Connection Open To use the Keep Connection Open database preference, *both* of the following must be true:

- **Working in PowerBuilder development environment** You must be working in the PowerBuilder development environment. The setting of Keep Connection Open has no effect in InfoMaker because the Connect DB at Startup preference controls when InfoMaker connects to the database.
- **Using default connection information** PowerBuilder must use the default connection information in the [Database] section of the PowerBuilder initialization file to connect to the database. Keep Connection Open has no effect when you select a database profile to connect to the database.

What happens If you meet both of these requirements, clearing the Keep Connection Open checkbox opens a database connection only when you are working in a painter that requires a connection and closes the connection at other times. This can save you money if you are accessing a database that charges for connect time.

Lock

Description

For those DBMSs and database interfaces that support the use of lock values and isolation levels, the Lock preference sets the isolation level to use when connecting to the database.

In multiuser databases, transactions initiated by different users can overlap. If these transactions access common data in the database, they can overwrite each other, or collide.

To prevent concurrent transactions from interfering with each other and compromising the integrity of your database, certain DBMSs allow you to set the isolation level when you connect to the database. **Isolation levels** are defined by your DBMS, and specify the degree to which operations in one transaction are visible to operations in a concurrent transaction. Isolation levels determine how your DBMS isolates or **locks** data from other processes while it is being accessed.

PowerBuilder uses the Lock preference to allow you to set various database lock options. Each lock value corresponds to an isolation level defined by your DBMS.

When to specify the Lock value

You must set the Lock value *before* you connect to the database in the PowerBuilder development environment or in a PowerBuilder application. The Lock value takes effect only when the database connection occurs. Changes to the Lock value after the connection occurs have no effect on the current connection.

Applies to

IN5 and IN7 INFORMIX (OnLine databases)
MSS Microsoft SQL Server 6.x
ODBC (if driver and backend DBMS support this feature)
SYC and SYD Sybase Systems 10.x and 11.x

In a PowerBuilder application

For those DBMSs and database interfaces that support it, you can set the Lock value in a script as a property of the transaction object. The following syntax assumes you are using the default transaction object, SQLCA, but you can also use a user-defined transaction object.

SQLCA.Lock = "*value*"

where *value* is the lock value you want to set.

Lock values

The following table lists the lock values and corresponding isolation levels for each Powersoft database interface that supports locking. You set the lock value in a PowerBuilder application script, and the isolation level in a database profile.

FOR INFO For more about the isolation levels that your DBMS supports, see your DBMS documentation.

Powersoft database interface	Lock values	Isolation levels
IN5 and IN7 INFORMIX (for OnLine databases only)	Dirty Read	Dirty Read
	Committed Read	Committed Read
	Cursor Stability	Cursor Stability
	Repeatable Read	Repeatable Read
Microsoft SQL Server 6.x	RU	Read Uncommitted
	RC	Read Committed
	RR	Repeatable Read
	TS	Serializable

Powersoft database interface	Lock values	Isolation levels
ODBC	RU	Read Uncommitted
	RC	Read Committed
	RR	Repeatable Read
	TS	Serializable Transactions
	TV	Transaction Versioning
Sybase Systems 10.x and 11.x	0	Read Uncommitted
	1	Read Committed (default)
	3	Serializable Transactions

In the development environment

Select the isolation level you want from the Isolation Level dropdown listbox on the Connection tab in the Database Profile Setup dialog box.

FOR INFO For instructions, see "Setting AutoCommit and Lock in the database profile" on page 341.

Default value

The default lock value depends on how your database is configured.

FOR INFO For more information, see your DBMS documentation.

Usage

INFORMIX The lock values for INFORMIX apply only to INFORMIX-OnLine databases Versions 5.x, 6.x, and 7.x. INFORMIX-SE (Standard Edition) databases do not support the use of lock values and isolation levels.

Microsoft SQL Server 6.x Microsoft SQL Server 6.x treats the RR (Repeatable Read) and TS (Serializable) settings the same, so choosing either of these settings results in the same behavior.

ODBC The TV (Transaction Versioning) setting does *not* apply to SQL Anywhere databases.

Sybase Systems 10.x and 11.x Sybase Systems 10.x and 11.x support the following lock values, which correspond to SQL Server isolation levels:

- **0—Read Uncommitted (dirty reads)** Isolation level 0 prevents other transactions from changing data that an uncommitted transaction has already modified (through SQL statements such as UPDATE).

  Other transactions cannot modify the data until the transaction commits. However, other transactions can still read the uncommitted data (perform dirty reads). Isolation level 0 prohibits retrieval locks on tables or pages.

  Isolation level 0 is valid only for Sybase System 10.1 or higher databases.

- **1—Read Committed** (Default) Isolation level 1 prevents dirty reads by issuing shared locks on tables or pages.

  A **dirty read** occurs when one transaction modifies a table row and a second transaction reads that row before the first transaction commits the change. If the first transaction rolls back the change, the information read by the second transaction becomes invalid.

- **3—Serializable Transactions (HOLDLOCK behavior)** Isolation level 3 prevents dirty reads, nonrepeatable reads, and phantoms for the duration of a transaction.

  A **nonrepeatable read** occurs when one transaction reads a row and then a second transaction modifies that row. If the second transaction commits the change, subsequent reads by the first transaction produce different results than the original read.

  A **phantom** occurs when one transaction reads a set of rows that satisfy a search condition, and then a second transaction modifies that data through a SQL INSERT, UPDATE, or DELETE statement. Subsequent reads by the first transaction using the same search conditions produce a different set of rows than the original read.

Dynamically controlling the isolation level PowerBuilder makes a second connection to implement either of the following while connected to a Sybase System 10.x or 11.x database:

- The Retrieve.AsNeeded property to specify that a DataWindow or report should retrieve only as many rows as needed from the database
- A SELECTBLOB embedded SQL statement to select a single blob column in a specified table row

The lock value you set before making the first System 10.x or 11.x connection is automatically inherited by the second connection, and *cannot be changed for the second connection.*

However, you can dynamically control the isolation level for the first (original) System 10.x or 11.x connection in a PowerBuilder application by coding the following PowerScript embedded SQL statement, where *n* is 0, 1, or 3 for the isolation level you want to set for the first connection:

```
EXECUTE IMMEDIATE "set transaction isolation level n"
```

For example, the following PowerScript embedded SQL code specifies isolation level 0 (dirty read behavior) for the second System 10.x or 11.x connection, and isolation level 1 (read committed behavior) for the first connection:

```
// Isolation level inherited by second connection
SQLCA.Lock = "0"
CONNECT USING SQLCA;
// Override lock value 0 for first connection only
EXECUTE IMMEDIATE "set transaction isolation level 1";
```

Examples

Example 1 To set the Lock value to RC (Read Committed) for a SQL Anywhere database:

- **Development environment** Select Read Committed from the Isolation Level dropdown listbox in the Database Profile Setup dialog box.
- **PowerBuilder application script** Type the following in a script:

```
SQLCA.Lock = "RC"
```

Example 2 To set the Lock value to 3 (Serializable Transactions) for a Sybase System 10.x or 11.x database:

- **Development environment** Select Serializable Transactions from the Isolation Level dropdown listbox in the Database Profile Setup dialog box.
- **PowerBuilder application script** Type the following in a script:

```
SQLCA.Lock = "3"
```

Using the examples in a PowerBuilder application script If you specify Isolation Level in your database profile, the PowerScript 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.

Read Only

Description

Read Only specifies whether PowerBuilder should update the repository tables and any other tables in your database. The **Powersoft repository** (also known as the Powersoft extended catalog or Powersoft system tables) consists of five tables that contain default extended attribute information for your database.

The Read Only setting determines whether you can modify (update) the tables in your database. By default, the Read Only checkbox is cleared in the Database Preferences property sheet. This means that PowerBuilder updates the repository tables and other tables in your database when you make changes.

If you select the Read Only checkbox, PowerBuilder *does not update* the repository tables or any other tables in your database. You *cannot* modify (update) information in the repository tables or any other database tables from the DataWindow or Report painters when the Read Only checkbox is selected.

Applies to

All Powersoft database interfaces

In PowerBuilder application

You *cannot* set the Read Only database preference in a PowerBuilder application script.

In development environment

In the PowerBuilder Database painter, select or clear the Read Only checkbox in the Database Preferences property sheet as follows:

- **Select the checkbox** PowerBuilder does not update the repository tables or any other tables in your database. You *cannot* modify (update) information in the repository tables or any other database tables from the DataWindow or Report painters when the Read Only checkbox is selected.
- **Clear the checkbox** (Default) PowerBuilder updates the repository tables and any other tables in your database when you modify them.

Default value

The Read Only checkbox in the Database Preferences property sheet is cleared by default.

Usage

If you select the Read Only checkbox in the Database Preferences property sheet, you cannot modify information in *any* tables from the DataWindow or Report painters.

Therefore, you can use only:

- SELECT and Retrieve statements in the DataWindow and Report painters
- SELECT statements in embedded SQL

See also

Use Powersoft Repository

Shared Database Profiles

Description

Specifies the pathname of the PowerBuilder initialization file containing the database profiles you want to share.

FOR INFO For instructions on sharing database profiles in the PowerBuilder development environment, see "Sharing database profiles" on page 321.

Applies to

All Powersoft database interfaces

In PowerBuilder application

You *cannot* set the Shared Database Profiles database preference in a PowerBuilder application script.

In development environment

In the Database painter, supply the pathname of the PowerBuilder initialization file containing shared profiles in the Shared Database Profiles box in the Database Preferences property sheet. You can type the pathname or click the Browse button to display it.

FOR INFO For instructions, see "Setting preferences in the Database Preferences property sheet" on page 343.

Default value

The Shared Database Profiles box in the Database Preferences property sheet is blank (unspecified) by default.

Examples

To share database profiles contained in the file I:\SHARE\PB.INI on the Windows platform, type or browse to the following in the Shared Database Profiles box in the Database Preferences property sheet:

```
I:\SHARE\PB.INI
```

SQL Terminator Character

Description

Specifies the SQL statement terminator character used by the Database Administration painter in PowerBuilder.

The default terminator character for the Database Administration painter is a semicolon (;). If a semicolon conflicts with the terminator character used by your DBMS syntax, you can change the painter's terminator character by specifying a different character in the SQL Terminator Character box in the Database Preferences property sheet. A good choice for a terminator character is the backquote (`) character.

Changing the Database Administration painter's terminator character is recommended when you are using the painter to create or execute stored procedures, triggers, and SQL scripts.

Applies to

All Powersoft database interfaces

In PowerBuilder application

You *cannot* set the SQL Terminator Character database preference in a PowerBuilder application script.

In development environment

In the Database Preferences property sheet in the Database painter, type the Database Administration painter terminator character you want to use in the SQL Terminator Character box.

FOR INFO For instructions, see "Setting preferences in the Database Preferences property sheet" on page 343.

Default value

The default SQL Terminator Character value in the Database Preferences property sheet is a semicolon (;).

Usage

The following are typical situations which may require you to change the default SQL Terminator Character value for the Database Administration painter:

- **Creating stored procedures and triggers** If you are creating stored procedures and triggers in the Database Administration painter, change the painter's terminator character to one that you do not expect to use in the stored procedure or trigger syntax for your DBMS, such as the backquote (`) character.

  After you finish using the stored procedure, you can change the Database Administration painter's terminator character back to semicolon (;) if you want. If you prefer, you can continue to use the new terminator character as long as it does not conflict with any stored procedure or trigger syntax you plan to use.

- **Executing SQL scripts** If you plan to execute any SQL scripts in the Database Administration painter, make sure the terminator character used in the script agrees with the terminator character currently set in the Database Administration painter.

Examples

To change the SQL statement terminator character in the Database Administration painter to a backquote (`), type a backquote in the SQL Terminator Character box in the Database Preferences property sheet.

Use Powersoft Repository

Description

Controls access to the Powersoft repository by specifying whether you want PowerBuilder to create the Powersoft repository tables. The **Powersoft repository** (also known as the Powersoft extended catalog or Powersoft system tables) consists of five tables that contain default extended attribute information for your database.

By default, the Use Powersoft Repository preference is selected in the Database Preferences property sheet. This setting creates the repository tables the first time you connect to a database using PowerBuilder.

Applies to

All Powersoft database interfaces

In PowerBuilder application

You *cannot* set the Use Powersoft Repository database preference in a PowerBuilder application script.

In development environment

In the Database painter, select or clear the Use Powersoft Repository checkbox in the Database Preferences property sheet as follows:

- **Select the checkbox** (Default) Creates the Powersoft repository tables when connecting to the database for the first time.
- **Clear the checkbox** Does *not* create the Powersoft repository tables if they do not exist. Instead, the DataWindow and Report painters use the appropriate default values for extended attributes (such as headers, labels, and text color). If the Powersoft repository tables already exist, PowerBuilder does not use them when you create a new DataWindow object or report.

Default value

The Use Powersoft Repository checkbox in the Database Preferences property sheet is selected by default.

Usage

If you clear the Use Powersoft Repository checkbox in the Database Preferences property sheet, PowerBuilder *does not do* any of the following:

- Create the repository tables
- Insert, update, or delete rows in the repository tables
- Select information (such as header names) from the repository tables
- Execute statements that reference the repository tables

See also

Read Only

APPENDIXES

APPENDIX A

Supported ODBC Drivers and Powersoft Database Interfaces

About this appendix

This appendix lists the ODBC drivers and Powersoft database interfaces supplied in PowerBuilder and InfoMaker.

As additional drivers and database interfaces are supported, updated information will be available electronically from Powersoft services on CompuServe, FTP, BBS, and the World Wide Web.

You should also check the *Release Notes* for the latest information about supported ODBC drivers and Powersoft database interfaces for your version of PowerBuilder or InfoMaker.

Contents

PowerBuilder

This section lists the ODBC drivers and Powersoft database interfaces supplied in PowerBuilder on all supported platforms.

Supported ODBC drivers

PowerBuilder supplies the following ODBC drivers. The DBMS identifier for using all of these drivers is *ODBC*.

Platform	ODBC drivers
Windows NT *and* Windows 95	INTERSOLV 3.0 Btrieve INTERSOLV 3.0 dBASE INTERSOLV 3.0 Excel Workbook INTERSOLV 3.0 Paradox 5 INTERSOLV 3.0 Text Sybase SQL Anywhere
Windows 3.x (deployment only)	INTERSOLV 2.1 Btrieve INTERSOLV 2.1 dBASE INTERSOLV 2.1 Excel 4 INTERSOLV 2.1 Excel Workbook INTERSOLV 2.1 Paradox 4 INTERSOLV 2.1 Paradox 5 INTERSOLV 2.1 Scalable SQL INTERSOLV 2.1 Text Sybase SQL Anywhere
Solaris	INTERSOLV 3.0 INFORMIX
HP-UX	INTERSOLV 3.0 INFORMIX
AIX	INTERSOLV 3.0 INFORMIX
Power Macintosh	Sybase SQL Anywhere
68K Macintosh (deployment only)	Sybase SQL Anywhere (direct connection)

Supported Powersoft database interfaces

PowerBuilder supplies the following native Powersoft database interfaces. The table also lists the DBMS identifier for using each interface.

Not in PowerBuilder Professional and PowerBuilder Desktop
The Powersoft database interfaces are *not supplied* with PowerBuilder Professional and PowerBuilder Desktop.

Platform	Powersoft database interfaces	DBMS identifier
Windows NT *and* Windows 95	INFORMIX I-Net 5	IN5
	INFORMIX I-Net 7	IN7
	Microsoft SQL Server 4.x DB-Lib	SYB
	Microsoft SQL Server 6.x	MSS
	ODBC interface	ODBC
	Oracle 7.1	O71
	Oracle 7.2	O72
	Oracle 7.3	O73
	Sybase InformationConnect DB2 Gateway	MDI
	Sybase Net-Gateway for DB2	NET
	Sybase SQL Server DB-Lib	SYT
	Sybase SQL Server 10.x and 11.x CT-Lib	SYC
Windows 3.x (deployment only)	IBM databases	IBM
	INFORMIX I-Net5	IN5
	Microsoft SQL Server 6.x	MSS
	ODBC interface	ODBC
	Oracle 7.1	O71
	Oracle 7.2	O72
	Oracle 7.3	O73
	SQL Server 4.x DB-Lib	SYB
	Sybase InformationConnect DB2 Gateway	MDI
	Sybase Net-Gateway for DB2	NET
	Sybase SQL Server 10.x and 11.x CT-Lib	SYC
Solaris	Oracle 7.2	O72
	Oracle 7.3	O73
	SQL Server 4.x DB-Lib	SYB
	Sybase SQL Server 10.x and 11.x CT-Lib	SYC
	Sybase SQL Server 10.x and 11.x distributed application interface	SYD
HP-UX	Oracle 7.3	O73
	SQL Server 4.x DB-Lib	SYB
	Sybase SQL Server 10.x and 11.x CT-Lib	SYC
	Sybase SQL Server 10.x and 11.x distributed application interface	SYD
AIX	Oracle 7.3	O73
	SQL Server 4.x DB-Lib	SYB
	Sybase SQL Server 10.x and 11.x CT-Lib	SYC
	Sybase SQL Server 10.x and 11.x distributed application interface	SYD

Platform	Powersoft database interfaces	DBMS identifier
Power Macintosh	ODBC interface	ODBC
	Oracle 7	OR7
	Oracle 7.1	O71
	Oracle 7.3	O73
	SQL Server 4.x DB-Lib	SYB
	Sybase SQL Server 10.x and 11.x CT-Lib	SYC
68K Macintosh (deployment only)	Oracle 7	OR7
	Oracle 7.1	O71
	SQL Server 4.x DB-Lib	SYB
	Sybase SQL Server 10.x and 11.x CT-Lib	SYC

InfoMaker

This section lists the ODBC drivers and Powersoft database interfaces supplied in InfoMaker on all supported platforms.

Supported ODBC drivers

InfoMaker supplies the following ODBC drivers. The DBMS identifier for using all of these drivers is *ODBC*.

Platform	ODBC drivers
Windows NT *and* Windows 95	INTERSOLV 3.0 Btrieve INTERSOLV 3.0 dBASE INTERSOLV 3.0 Excel Workbook INTERSOLV 3.0 Paradox 5 INTERSOLV 3.0 Text Sybase SQL Anywhere
Windows 3.x	INTERSOLV 2.1 Btrieve INTERSOLV 2.1 dBASE INTERSOLV 2.1 Excel 4 INTERSOLV 2.1 Excel Workbook INTERSOLV 2.1 Paradox 4 INTERSOLV 2.1 Paradox 5 INTERSOLV 2.1 Scalable SQL INTERSOLV 2.1 Text Sybase SQL Anywhere
Power Macintosh	Sybase SQL Anywhere
68K Macintosh	Sybase SQL Anywhere (direct connection)

Supported Powersoft database interfaces

InfoMaker supplies the following native Powersoft database interfaces. The table also lists the DBMS identifier for using each interface.

Platform	Powersoft database interfaces	DBMS identifier
Windows NT *and* Windows 95	INFORMIX I-Net 5	IN5
	INFORMIX I-Net 7	IN7
	Microsoft SQL Server 4.x DB-Lib	SYB
	Microsoft SQL Server 6.x	MSS
	ODBC interface	ODBC
	Oracle 7.1	O71
	Oracle 7.2	O72
	Oracle 7.3	O73
	Sybase InformationConnect DB2 Gateway	MDI
	Sybase Net-Gateway for DB2	NET
	Sybase SQL Server DB-Lib	SYT
	Sybase SQL Server 10.x and 11.x CT-Lib	SYC
Windows 3.x	IBM databases	IBM
	INFORMIX I-Net 5	IN5
	Microsoft SQL Server 6.x	MSS
	ODBC interface	ODBC
	Oracle 7.1	O71
	Oracle 7.2	O72
	Oracle 7.3	O73
	SQL Server 4.x DB-Lib	SYB
	Sybase InformationConnect DB2 Gateway	MDI
	Sybase Net-Gateway for DB2	NET
	Sybase SQL Server 10.x and 11.x CT-Lib	SYC
Power Macintosh	ODBC interface	ODBC
	Oracle 7	OR7
	Oracle 7.1	O71
	Oracle 7.3	O73
	SQL Server 4.x DB-Lib	SYB
	Sybase SQL Server 10.x and 11.x CT-Lib	SYC
68K Macintosh	Oracle 7	OR7
	Oracle 7.1	O71
	SQL Server 4.x DB-Lib	SYB
	Sybase SQL Server 10.x and 11.x CT-Lib	SYC

Using ODBC drivers from other vendors

The ability to use Level 1 or higher ODBC-compliant drivers supplied by Powersoft or another vendor to access data in PowerBuilder Enterprise, PowerBuilder Professional, or InfoMaker depends on the platform you are using:

Platform	Product includes ODBC driver(s)	Product allows use of other vendors' ODBC drivers
Windows	Yes (INTERSOLV and SQL Anywhere)	Yes
Power Macintosh	Yes (SQL Anywhere)	Yes
68K Macintosh	Yes (SQL Anywhere)	Yes
UNIX	Yes (INTERSOLV)	No

In most cases, Level 1 or higher ODBC-compliant drivers obtained from other vendors will work with PowerBuilder and InfoMaker. However, Powersoft may not have tested the drivers to verify this.

PowerBuilder Desktop

There are certain restrictions when using ODBC drivers with PowerBuilder Desktop.

FOR INFO For information, see "Using ODBC drivers with PowerBuilder Desktop" next.

Using ODBC drivers with PowerBuilder Desktop

Using ODBC drivers that come with Desktop

If you are using PowerBuilder Desktop, you can access data using the ODBC drivers that are shipped with the product. Unlike PowerBuilder Enterprise and PowerBuilder Professional, you *cannot* use an ODBC driver obtained from another vendor with PowerBuilder Desktop.

Using existing Microsoft ODBC drivers

If you already have Version 2.0 or higher 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 INTERSOLV drivers is recommended

PowerBuilder Desktop comes with INTERSOLV ODBC drivers for all of these data sources *except Access*. Since the INTERSOLV drivers have been tested to work with PowerBuilder Desktop, you should use the INTERSOLV drivers whenever possible to access these data sources.

APPENDIX B

Adding Functions to the PBODB60 Initialization File

About this appendix

In general, *you should not need to modify the PBODB60 initialization file*. In certain situations, however, you may need to add functions to the PBODB60 initialization file for connections to your backend DBMS through the Powersoft ODBC interface in PowerBuilder.

This appendix describes how to add functions to the PBODB60 initialization file if necessary.

Contents

About the PBODB60 initialization file

What is the PBODB60 initialization file?

When you access data through the Powersoft ODBC interface, PowerBuilder uses the PBODB60 initialization file to maintain access to extended functionality in the backend 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 PBODB60 initialization file

In most cases, you should *not* need to modify the PBODB60 initialization file. Changes to this file can adversely affect PowerBuilder. You should change the PBODB60 initialization file only if you are asked to do so by a Technical Support representative.

However, if you *do* need to add functions to the PBODB60 initialization file for your backend DBMS, you can edit the file *only* on those PowerBuilder platforms that fully support ODBC connectivity, as follows:

Platform	Can edit PBODB60 initialization file
Windows	Yes
Macintosh	Yes
UNIX	No

If you modify the PBODB60 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 PBODB60 initialization file, tell the representative that you changed the file and describe the changes you made.

Finding the file on different platforms

The name and default location of the PBODB60 initialization file in PowerBuilder depends on the platform you are using.

Platform	Location
Windows NT Windows 95	Program Files\Powersoft\Shared\PBODB60.INI
Macintosh	System Folder:Preferences:Powersoft ODBC 6.0 Preferences
UNIX	$PBHOME/bin/pbodb60.ini

For simplicity, this appendix refers generically to the PBODB60 initialization file throughout.

Adding functions to the PBODB60 initialization file

The PBODB60 initialization file lists the functions for certain DBMSs that have ODBC drivers. If you need to add a function to the PBODB60 initialization file for use with your backend DBMS, you can do either of the following:

- **Existing sections** Add the function to the Functions section for your backend database if this section exists in the PBODB60 initialization file.
- **New sections** Create new sections for your backend DBMS in the PBODB60 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 backend DBMS *already exist* in the PBODB60 initialization file, use the following procedure to add new functions.

❖ **To add functions to an existing section in the PBODB60 initialization file:**

1 Open the PBODB60 initialization file in one of the following ways:

- Use the File Editor in PowerBuilder. (For instructions, see the *PowerBuilder User's Guide*.)
- Use any text editor outside PowerBuilder.

2 Locate the entry for your backend DBMS in the DBMS Driver/DBMS Settings section of the PBODB60 initialization file.

For example, here is the PBODB60 initialization file entry for Sybase SQL Anywhere:

```
;*****************************************
;DBMS Driver/DBMS Settings see comments at end
;of file
;*****************************************
...
[Sybase SQL Anywhere]
PBSyntax='WATCOM50_SYNTAX'
PBDateTime='STANDARD_DATETIME'
PBFunctions='WATCOM_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 PBODB60 initialization file that contains function information for your backend 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 Sybase SQL Anywhere indicates that the name of the Functions section is WATCOM_FUNCTIONS:

```
PBFunctions='WATCOM_FUNCTIONS'
```

4 Find the Functions section for your backend DBMS in the PBODB60 initialization file.

For example, here is the Functions section for Sybase SQL Anywhere:

```
;*****************************************
;Functions
;*****************************************
[WATCOM_FUNCTIONS]
AggrFuncs=avg(),count(),list(),max(),min(),sum()
```

```
Functions=length(),similar(),soundex(),substr(),
   string(),date(),dateformat(),datetime(),
   day(),days(),dow(),hour(),hours(),minute(),
   minutes(),second(),seconds(),month(),
   months(),now(*),today(*),weeks(),year(),
   years(),ymd(),abs(),ifnull(),isnull(),
   number(*),remainder(),mod()
```

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 backend 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 function for Sybase SQL Anywhere added at the end of the Functions list:

```
;*******************************************
;Functions
;*******************************************
[WATCOM_FUNCTIONS]
AggrFuncs=avg(),count(),list(),max(),min(),sum()
Functions=length(),similar(),soundex(),substr(),
   string(),date(),dateformat(),datetime(),
   day(),days(),dow(),hour(),hours(),minute(),
   minutes(),second(),seconds(),month(),
   months(),now(*),today(*),weeks(),year(),
   years(),ymd(),abs(),ifnull(),isnull(),
   number(*),remainder(),mod() ,newfunction()
```

6 Save your changes to the PBODB60 initialization file.

Adding functions to a new section in the file

If entries for your backend DBMS *do not exist* in the PBODB60 initialization file, use the following procedure to create the required sections and add the appropriate functions.

Before you start

FOR INFO For more about the settings you should supply for your backend DBMS in the PBODB60 initialization file, read the comments at the end of the file.

❖ **To add functions to a new section in the PBODB60 initialization file:**

1 Open the PBODB60 initialization file in one of the following ways:

- Use the File Editor in PowerBuilder. (For instructions, see the *PowerBuilder User's Guide*.)
- Use any text editor outside PowerBuilder.

2 Edit the DBMS Driver/DBMS Settings section of the PBODB60 initialization file to add an entry for your backend DBMS.

Finding the name

The name required to identify the entry for your backend DBMS in the DBMS Driver/DBMS Settings section is in the ODBC initialization file.

FOR INFO For information about the name and location of the ODBC initialization file on your platform, see "How Powersoft products access the data source" on page 35.

Make sure that you:

- Follow the instructions in the comments at the end of the PBODB60 initialization file.
- Use the same syntax as existing entries in the DBMS Driver/DBMS Settings section of the PBODB60 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 PBODB60 initialization file to add an entry for your backend DBMS.

Make sure that you:

- Follow the instructions in the comments at the end of the PBODB60 initialization file.
- Use the same syntax as existing entries in the Functions section of the PBODB60 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 backend 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 PBODB60 initialization file.

Index

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

R

S

T

U

V

W

X