



Open Tools from Sybase, Inc.

PowerBuilder

Feature Guide

Version 6

Power Builder®

AA0526

October 1997

Copyright © 1991-1997 Sybase, Inc. and its subsidiaries.

All rights reserved.

Printed in the United States of America.

Information in this manual may change without notice and does not represent a commitment on the part of Sybase, Inc. and its subsidiaries.

The software described in this manual is provided by Powersoft Corporation under a Powersoft License agreement. The software may be used only in accordance with the terms of the agreement.

No part of this publication may be reproduced, transmitted, or translated in any form or by any means, electronic, mechanical, manual, optical, or otherwise, without the prior written permission of Sybase, Inc. and its subsidiaries.

Sybase, Inc. and its subsidiaries claim copyright in this program and documentation as an unpublished work, revisions of which were first licensed on the date indicated in the foregoing notice. Claim of copyright does not imply waiver of other rights of Sybase, Inc. and its subsidiaries.

ClearConnect, Column Design, ComponentPack, InfoMaker, ObjectCycle, PowerBuilder, PowerDesigner, Powersoft, S-Designer, SQL SMART, and Sybase are registered trademarks of Sybase, Inc. and its subsidiaries. Adaptive Component Architecture, Adaptive Server Anywhere, Adaptive Server Enterprise, Adaptive Warehouse, AppModeler, DataArchitect, DataExpress, Data Pipeline, DataWindow, dbQueue, ImpactNow, InstaHelp, Jaguar CTS, jConnect for JDBC, MetaWorks, NetImpact, Optima++, Power++, PowerAMC, PowerBuilder Foundation Class Library, Power J, PowerScript, PowerSite, Powersoft Portfolio, Powersoft Professional, PowerTips, ProcessAnalyst, Runtime Kit for Unicode, SQL Anywhere, The Model For Client/Server Solutions, The Future Is Wide Open, Translation Toolkit, UNIBOM, Unilib, Uninull, Unisep, Unistring, Viewer, WarehouseArchitect, Watcom, Watcom SQL Server, Web.PB, and Web.SQL are trademarks of Sybase, Inc. or its subsidiaries. Certified PowerBuilder Developer and CPD are service marks of Sybase, Inc. or its subsidiaries. DataWindow is a patented proprietary technology of Sybase, Inc. or its subsidiaries.

AccuFonts is a trademark of AccuWare Business Solutions Ltd.

All other trademarks are the property of their respective owners.

Contents

About This Book	vii
1	Release 6.0 Features..... 1
	Distributed computing 2
	Shared objects 2
	Server push 3
	Asynchronous processing 4
	DataWindow synchronization 5
	Name Server 5
	Wired for the Web 7
	Internet Tools 7
	PowerBuilder window ActiveX 8
	Enhancements to DataWindow HTML creation 10
	Secure mode for plug-ins and the PowerBuilder window ActiveX 13
	Context feature 15
	Synchronizer for automatic file updates 17
	Web.PB wizard on the PowerBar 17
	Web jumps from within PowerBuilder 17
	Open technology 18
	Component generators 18
	Cross-platform support 19
	Internationalization 21
	Database connectivity 24
	OLE enhancements 26
	Standard source control API 29
	Developer productivity 30
	Tracing and profiling 30
	Debugger interface 32
	DataWindow features 33
	PFC enhancements 38
	Ease-of-use and language features 40
	Deployment and execution time enhancements 47
	Web.PB wizard and OLE GenReg on the PowerBar 51

	Component Gallery	52
	HOW Learning Edition.....	52
	PSChart.....	53
2	What You Can Build	55
	Choosing an architecture	56
	Client/server	57
	Single-tier	57
	Two-tier (traditional client/server)	58
	Multitier.....	60
	Internet.....	62
	Web.PB	62
	PowerBuilder window ActiveX.....	63
	PowerBuilder window plug-in	65
	DataWindow plug-in	66
	PowerBuilder automation server component	68
3	The Building Blocks	71
	Choosing technologies.....	72
	User interface.....	74
	Windows.....	74
	Controls	78
	Menus.....	81
	User interface style.....	83
	Application programming interface.....	86
	Language	88
	Scripting	88
	Objects	94
	Object-oriented programming.....	98
	Visual and nonvisual classes	100
	Class libraries	101
	Data access	105
	Database connections.....	105
	Embedded SQL.....	109
	DataWindow objects.....	110
	DataWindow controls	117
	DataStores	118
	Text and binary files	119
	Clipboard.....	119
	Program access	120
	Executable programs	120
	Help systems.....	120
	E-mail systems	121

Component object model	122
DDE	126
DLL and shared-library functions	127
C++ classes.....	128
Database stored procedures	128
Distributed PowerBuilder objects	129
Output	131
Printed reports.....	131
File generation.....	132
Printing	134
Data pipelines.....	135
Environment	137
Application logistics	137
Application context	138
Platform support	139
International support.....	143
Initialization files and registries.....	143
Code generation	144

4

How You Build It	147
Using tools	148
Using objects.....	151
Testing and deployment.....	158

About This Book

Subject

This book describes new PowerBuilder features as well as PowerBuilder implementation architectures and features.

Audience

This book is for both new and experienced PowerBuilder developers:

- ◆ **New PowerBuilder developers** Use this book to learn about PowerBuilder features and the types of applications you can build with PowerBuilder
- ◆ **Experienced PowerBuilder developers** Use this book to learn about new PowerBuilder features and implementation architectures

Release 6.0 Features

About this chapter

This chapter presents an overview of the new features in PowerBuilder 6.0.

Contents

Topic	Page
Distributed computing	2
Wired for the Web	7
Open technology	18
Developer productivity	30

Distributed computing

PowerBuilder 6.0 includes the following distributed computing enhancements:

- ◆ Shared objects
- ◆ Server push
- ◆ Asynchronous processing
- ◆ DataWindow synchronization
- ◆ Name Server

Shared objects

Description

To allow you to work with persistent, shared data in a distributed application, PowerBuilder provides support for shared objects. Shared objects are user objects that can be shared by multiple client connections.

Not deployable on Windows 3.x and Macintosh

Shared objects are supported only on those platforms where server-side processing and multithreading are supported. Support for shared objects is not available on Windows 3.x and the Macintosh (because you cannot deploy a server application on either of these platforms).

Purpose

Shared objects provide significant benefits to both distributed PowerBuilder applications and Internet applications that use Web.PB. Shared objects allow you to:

- ◆ Provide convenient access to common data that would otherwise need to be retrieved separately by each client connection
- ◆ Reduce the number of database accesses, freeing the database server for other processing
- ◆ Maintain state information in Web.PB applications

Usage

To allow multiple client applications to share a single object, the server application performs these operations:

- 1 Invokes the SharedObjectRegister function to register a named instance of the object
- 2 Invokes the SharedObjectGet function to get an object instance that is a reference to the shared object

The server can perform these operations in its main thread or inside a client session. The server application does not need to issue a CREATE statement for the shared object. When the server calls the SharedObjectRegister function, PowerBuilder automatically creates the shared object instance.

Shared objects are accessible only from the server's main session and from the client sessions created for each client connection on the server. Client applications cannot access a shared object directly. To access a shared object, a client needs to communicate with another object that has an instance variable that provides a reference to the shared object. This object can be thought of as the **shared object wrapper**.

Once the server has registered the shared object instance and retrieved a reference to the object, client applications can call the methods defined for the shared object by referencing the shared object wrapper. The shared object wrapper has methods that provide indirect access to the methods of the shared object. Typically, these methods have the same names as the methods defined for the shared object.

At execution time, PowerBuilder creates a separate thread (session) for each shared object instance and any objects that the shared object creates. The thread for a shared object is created using the Application object definition for the server application.

Documentation See *Application Techniques*.

Server push

Description Using a technique called **server push**, the server can send messages back to the client.

Purpose Server push is particularly useful when the client needs to be notified of the completion of an asynchronous request, or to push updated data to clients at regular intervals.

Usage The server can make both synchronous and asynchronous calls against client-side objects. The client handles server-side requests the same way the server handles client-side requests. Asynchronous requests against a particular client-side object are queued and processed after all synchronous requests.

To send a message to the client, the server needs to know which client-side object to send the message to. So the client must pass an object reference to the server. When the server receives the object reference, it creates a remote reference to the client-side object and calls one or more functions associated with this object. Function calls made against the remote reference are passed back over the wire to the client that contains the object.

To ensure that messages are actually sent to the client-side object, the client must not pass an autoinstantiated object to the server. Instead, the client must pass a reference to an object that is created with the CREATE statement.

Documentation See *Application Techniques*.

Asynchronous processing

Description Unlike synchronous calls, which force the client to wait until processing has completed, **asynchronous** calls free the client to do other work while the server handles requests.

Purpose When the timing of function execution is not critical, you can use asynchronous function calls to improve system throughput.

Usage To make an asynchronous function call, you need to call the remote object function with the POST keyword. If the POST keyword is not used, the command executes synchronously. All asynchronous requests are queued on the server and processed in the order received.

For example, the following script instantiates a remote object on the server and makes an asynchronous call to a function of the remote object:

```
// Global variable: connection myconnect
// Instance variable: uo_custdata mycustdata

myconnect.CreateInstance(mycustdata)
mycustdata.post retrieve_data()
```

Documentation See *Application Techniques*.

DataWindow synchronization

Description	<p>In a conventional client/server application, where database updates are initiated by a single application running on a client machine, PowerBuilder can manage DataWindow state information automatically. But in a distributed application, the situation is somewhat different. Because application components are partitioned between the client and the server, you need to write logic to ensure that the data buffers and status flags for the DataWindow control on the client are synchronized with those for the DataStore on the server.</p>
Usage	<p>To synchronize a DataWindow control on the client with a DataStore on the server, you need to move the DataWindow data buffers and status flags back and forth between the client and the server whenever changes occur. The procedures for doing this are essentially the same whether the source of the changes resides on the client or the server.</p>
Functions	<p>PowerBuilder provides five functions for synchronizing DataWindows and DataStores in a distributed application:</p> <ul style="list-style-type: none">GetFullStateGetChangesGetStateStatusSetFullStateSetChanges <p>Although these functions are most useful in distributed applications, they can also be used in nondistributed applications where multiple DataWindows (or DataStores) must be synchronized.</p>
Documentation	<p>For information about DataWindow synchronization, see <i>Application Techniques</i>. For information about functions, see the <i>PowerScript Reference</i>.</p>

Name Server

Description	<p>The Name Server is a PowerBuilder application that acts as a logical server in a distributed computing environment. By managing requests between clients and physical servers, the Name Server insulates clients from the physical locations of the servers. When a client makes a request to connect to a particular class of server, the Name Server redirects the client to the best physical server choice.</p>
-------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Purpose	The Name Server provides a load balancing mechanism for distributed applications. To optimize the use of available computing resources, the Name Server examines the state of each physical server to determine where to direct client requests.
Usage	To take advantage of the capabilities of the Name Server, you need to assign each physical server to a logical server class. You typically have multiple physical servers for each class you define. For example, the logical class named <i>Accounting</i> could handle requests against the physical servers Acct_1, Acct_2, and Acct_3. When a client makes a request against the Accounting server class, the Name Server returns the physical connection information for the physical server that is best equipped to handle the request.
Documentation	See the online Help.

Wired for the Web

PowerBuilder 6.0 includes features that allow you to deliver more powerful Internet-based applications:

- ◆ Internet Tools
- ◆ PowerBuilder window ActiveX
- ◆ Enhancements to DataWindow HTML creation
- ◆ Secure mode for plug-ins and the PowerBuilder window ActiveX
- ◆ Context feature
- ◆ Web jumps from within PowerBuilder

Internet Tools

Description

PowerBuilder Internet Tools include Web.PB and plug-ins for windows and DataWindows. The following components in the Internet Tools have been enhanced:

Component	Enhancement
Window plug-in	Secure mode
	The ability to use the application open event
	The ability to use PowerBuilder global variables
	New MIME types for the PBD extension: <ul style="list-style-type: none"> ◆ PowerBuilder window plug-in: application/vnd.powerbuilder6 ◆ Secure mode PowerBuilder window plug-in: application/vnd.powerbuilder6-s
Web.PB	Distributed computing enhancements
Class library	Minor enhancements including cookie support

Additionally, the Internet Tools now contain the PowerBuilder window ActiveX control, which you use to provide the window plug-in functionality within Internet Explorer.

Purpose

You use the Internet Tools to create Web-based applications and to Internet-enable current applications.

Documentation For information about distributed computing enhancements, see "Distributed computing" on page 2. For information about using Internet Tools, see *Building Internet Applications with PowerBuilder*.

PowerBuilder window ActiveX

Description The PowerBuilder window ActiveX control is a component object that can hold PowerBuilder child windows. In addition to providing all the capabilities of the PowerBuilder Netscape plug-ins, the window ActiveX control features access via JavaScript or VBScript to a subset of the window's events and functions.

The PowerBuilder window ActiveX control includes functions you can call to invoke methods in the window contained in the control. Additionally, certain events in the control trigger events in the ActiveX container. When an event fires, it first executes any PowerBuilder script for the event and then executes the code (JavaScript or VB Script) implemented for the container.

A PowerBuilder window displayed in the window ActiveX can include all the familiar controls, including DataWindows, OLE objects, other ActiveX controls, and TreeView controls. You can also open other popup and response windows from the child window.

As the user interacts with controls in the windows, scripts for the control's events are executed just as they are in standalone PowerBuilder applications. Database access by the window occurs locally using the client's defined database connections.

The objects in the application can be contained in one or more PowerBuilder dynamic libraries (PBDs).

Purpose Use the PowerBuilder window ActiveX to provide a graphical interface inside HTML pages. You can use it with any Web browser that supports ActiveX. This includes Internet Explorer and Netscape Navigator running the ActiveX plug-in.

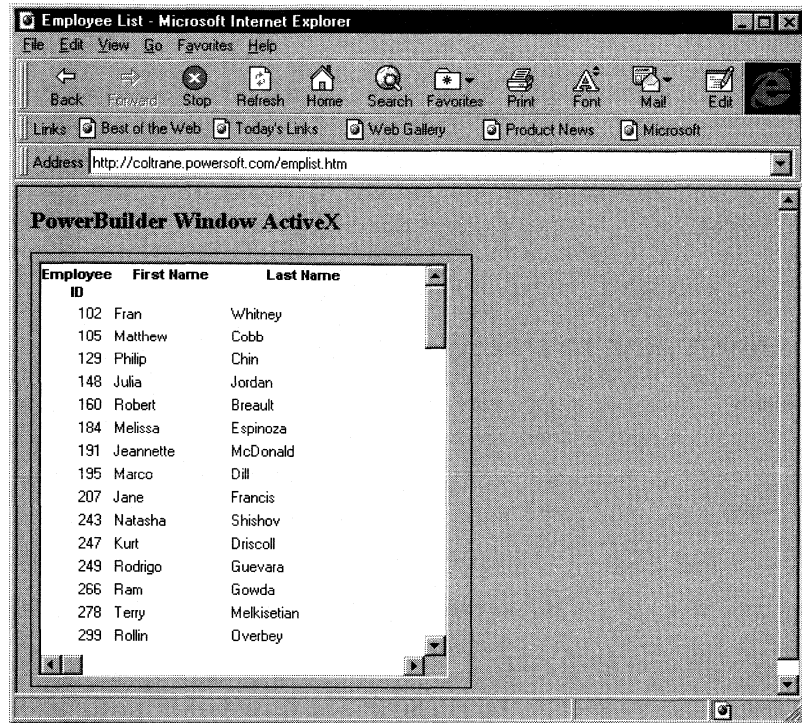
Although this feature is primarily intended for use with Web browsers that support ActiveX, you can display a PowerBuilder window or DataWindow in any application that supports ActiveX.

The PowerBuilder window ActiveX requires that the client workstation contain the PowerBuilder deployment DLLs as well as a registered version of the PowerBuilder Window ActiveX module (either PBRX60.OCX or PBRXS60.OCX).

Secure mode

PBRXS60.OCX is the secure-mode version of the PowerBuilder window ActiveX.

Illustration



Usage

❖ **To use the PowerBuilder window ActiveX:**

- 1 Create, test, and build the PowerBuilder application.
- 2 Create a PBD.
- 3 Register the PowerBuilder window ActiveX on the HTML development workstation.
- 4 Create an HTML page that includes your PowerBuilder application window. This page can include JavaScript or VBScript that interacts with the window.

- 5 Configure the Web server by copying the HTML pages, the PowerBuilder window ActiveX, and PBD files for the application to the appropriate directories.
- 6 On all client workstations, install the PowerBuilder window ActiveX and the PowerBuilder deployment DLLs.

Functions

GetArgElement
InvokePBFunction
TriggerPBEvent
GetLastReturn
ResetArgElements
SetArgElement

Documentation

See *Building Internet Applications with PowerBuilder*.

Enhancements to DataWindow HTML creation

Description

These are the enhancements to PowerBuilder's DataWindow HTML generation capabilities:

- ◆ Additional properties to help you control the display of DataWindow data when displayed in HTML tables
These properties allow you to control cell formatting and borders.
- ◆ Style sheets that contain DataWindow formatting
PowerBuilder converts certain DataWindow presentation information into an HTML cascading style sheet. It saves this style sheet in the HTML.StyleSheet property, and syntax within the HTMLTable property includes references to the generated styles. If you call the SaveAs function with the HTMLTable enumeration, PowerBuilder embeds the style sheet in the generated syntax.
- ◆ Form generation capability
You can call the GenerateHTMLForm function to create an HTML form from data contained in a DataWindow control or DataStore whose DataWindow object uses the Freeform or Tabular presentation style. You can generate HTML forms for a specified set of rows and columns; PowerBuilder preserves formatting such as radio buttons, checkboxes, and listboxes. It also includes any command buttons defined for the DataWindow.

The HTMLTable.GenerateCSS property controls the downward compatibility of the HTML found in the HTMLTable property. If HTMLTable.GenerateCSS is FALSE, formatting (style sheet references) is not referenced in the HTMLTable property; if it is TRUE, the HTMLTable property includes elements that reference the cascading style sheet saved in HTML.StyleSheet.

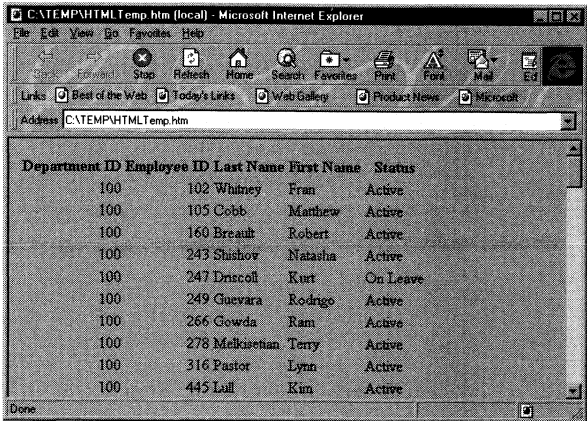
Purpose

The value added in these features is the easy transfer of data from the database to HTML pages. Two typical uses are:

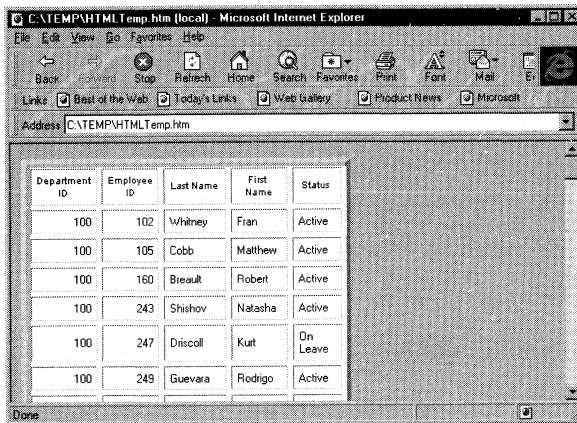
- ◆ An application that creates HTML pages on a regular basis
- ◆ A Web.PB application that creates HTML pages on demand

Illustration

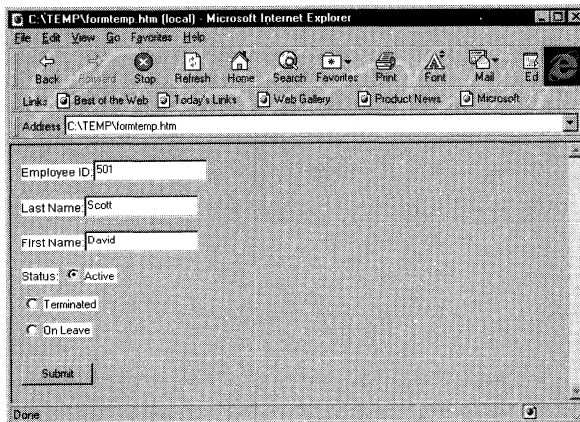
This HTML page shows an HTML table using PowerBuilder 5.0 functionality:



This HTML page shows an HTML table using PowerBuilder 6.0 enhancements:



This HTML page shows a freeform DataWindow converted into a form using syntax generated by the GenerateHTMLForm function:



Usage

HTML tables When working with DataWindows to be displayed as HTML tables, you can:

- ◆ Create HTML using the HTMLTable property, optionally merging HTMLTable with HTMLTable.StyleSheet
- ◆ Create HTML by calling the SaveAs function with the HTMLTable! enumeration
- ◆ Create HTML by saving rows displayed in DataWindow painter Preview

HTML forms When working with DataWindows to be displayed as HTML forms, you can:

- ◆ Create freeform and tabular DataWindow objects for use as HTML forms, including command buttons
- ◆ Create HTML form syntax by calling the GenerateHTMLForm function
- ◆ Build an HTML page by merging the GenerateHTMLForm result with the style sheet and adding other supporting HTML syntax

You can use both the enhanced HTMLTable capabilities and the form creation capability to minimize coding in a Web.PB application.

Functions

GenerateHTMLForm

Documentation

For information about new DataWindow object properties, see *Objects and Controls*. For information about the GenerateHTMLForm function, see the *PowerScript Reference*. For information about using HTML generation, see *Application Techniques*.

Secure mode for plug-ins and the PowerBuilder window ActiveX

Description

Secure mode helps to ensure that PowerBuilder applications downloaded over the Internet will not damage a client system or access information on the client system. The types of activity restricted by secure mode include:

- ◆ **External functions** Calling an external function causes an execution time error
- ◆ **Certain PowerScript functions** Calling a restricted PowerScript function causes an execution time error
- ◆ **Database connection** Calling PowerScript functions that result in database access causes an execution time error
- ◆ **Internet access** Applications running in secure mode can only establish an internet connections to the current Web server
- ◆ **E-mail** Calling PowerScript Mail functions causes an execution time error
- ◆ **OLE (restricted in Netscape plug-in only)** Calling PowerScript OLE functions causes an execution time error
- ◆ **Distributed computing** You cannot connect to PowerBuilder application servers

- ◆ **Dynamic Data Exchange (DDE)** Calling PowerScript DDE functions causes an execution time error

Purpose

You use secure mode to protect client workstations from rogue applications written in PowerBuilder.

Severely restricted functionality

Running in secure mode severely restricts the PowerBuilder application running on the client workstation, denying access to the client system except for printing and (PowerBuilder window ActiveX only) using OLE commands. Because of this, secure mode may not be appropriate in all situations.

Usage

You implement secure mode by deploying special versions of the PowerBuilder window plug-in or PowerBuilder window ActiveX, which you deploy on client workstations as appropriate:

Product	Default version	Secure version
PowerBuilder window plug-in	NPPBA60.DLL	NPPBS60.DLL
PowerBuilder window ActiveX	PBRX60.OCX	PBRXS60.OCX

DataWindow plug-in

The DataWindow plug-in displays Powersoft report (PSR files) only. PSR files are read-only, so there is not a secured version of the DataWindow plug-in (NPDWE60.DLL).

- ❖ **To use the secured version of the PowerBuilder window plug-in:**
 - 1 Copy NPPBS60.DLL to the PLUGINS directory on each client's workstation.
 - 2 Perform all other client setup steps, as described in *Building Internet Applications with PowerBuilder*.
- ❖ **To use the secured version of the PowerBuilder window ActiveX:**
 - 1 Install and register PBRXS60.OCX on the workstation used for HTML page creation.
 - 2 Use the Class ID for PBRXS60.OCX when specifying the OBJECT element for a secured application.

- 3 (Optional) Install and register PBRXS60.OCX on all client workstations, ensuring that you use the same Class ID specified in the HTML Object elements. (You could perform this step when installing the PowerBuilder deployment DLLs on the client workstation.)
- 4 Perform other client setup steps, as described in *Building Internet Applications with PowerBuilder*.

Documentation

See *Building Internet Applications with PowerBuilder*.

Context feature

Description

The PowerBuilder context feature allows applications to access certain host (non-PowerBuilder) services:

- ◆ Context information service (available on all platforms)
- ◆ Keyword service (available on all platforms)
- ◆ Internet service (available on Windows 95 and Windows NT only)

These services provide environment-specific functionality for the following environments (also called contexts):

- ◆ PowerBuilder execution time (default context)
- ◆ PowerBuilder window plug-in
- ◆ PowerBuilder window ActiveX

All aspects of each service may not be available in all contexts. Powersoft may provide additional services after the release of PowerBuilder 6.0. Additionally, you can write your own services by creating descendants of the service object.

Similar to the COM QueryInterface

This feature provides functionality similar to the COM QueryInterface.

- Purpose** Use this feature to enhance the capabilities of your applications. For example, by using the functions provided by the services, you can:
- ◆ Open the default browser, displaying a URL from within a PowerBuilder application
 - ◆ Access application arguments and environment variables
 - ◆ Control the browser from within a PowerBuilder application
 - ◆ Determine the execution context, modifying the application's look, feel, and processing depending on the environment

Usage You begin by instantiating one of the services through the `GetContextService` PowerScript function. This function takes an argument specifying the requested service and returns a reference to the instantiated service. In this example, PowerBuilder instantiates the context information service for use within a window (`icxinfo_base` is a window-level instance variable of type `ContextInformation`):

```

this.GetContextService("ContextInformation", &
    icxinfo_base)

```

You can then use the reference variable in dot notation to call functions for the `ContextInformation` object.

At execution time, PowerBuilder determines the current execution environment and enables the requested service as appropriate for the environment.

This table describes basic usage for each of the services:

Service	Usage
Context information	Access basic information on the current execution context. Sample contexts include native PowerBuilder execution environment, PowerBuilder window plug-in, and window ActiveX. When running the window ActiveX within Internet Explorer, this service can also access the <code>IWebBrowserApp</code> ActiveX automation server object, which provides access to browser methods and properties
Keyword	Access environment information for the current context. When running within the PowerBuilder window plug-in, this service allows you to access parameters specified in the plug-in's <code>Embed</code> element
Internet	Display a URL in the default browser. This service also allows you to get and post URLs

Functions

Service	Functions
To enable a service	GetContextService
ContextInformation	GetName GetHostObject GetShortName GetFixesVersion GetCompanyName GetMajorVersion GetVersionName GetMinorVersion
Keyword	GetContextKeywords
Internet	GetURL HyperlinkToURL PostURL InternetData

Documentation

See *Application Techniques*.

Synchronizer for automatic file updates

The Synchronizer includes an ActiveX that you can insert in an HTML page to update files in a deployed application from the World Wide Web.

FOR INFO For information, see "The Synchronizer" on page 48.

Web.PB wizard on the PowerBar

You can invoke the Web.PB wizard from the PowerBar.

FOR INFO For information, see "Web.PB wizard and OLE GenReg on the PowerBar" on page 51.

Web jumps from within PowerBuilder

On Windows, the Help menu provides four links to the World Wide Web from within the PowerBuilder development environment. This feature gives you easy access to Web pages you use frequently.

Open technology

PowerBuilder 6.0 features open technology that encompasses:

- ◆ Component generators
- ◆ Cross-platform support
- ◆ Internationalization
- ◆ Database connectivity
- ◆ OLE enhancements
- ◆ Standard source control API

Component generators

Description	PowerBuilder 6.0 includes a project generator infrastructure that enables future support for open interfaces to PowerBuilder objects. Component generators will be added to support the creation of standard object types from objects developed in PowerBuilder.
Purpose	Generating multiple component types from PowerBuilder objects will enable an easy transition from one component model to another. Components generated in PowerBuilder 6.0 can be deployed into a variety of middle-tier server environments.
Usage	<p>When you create a new project in the Project painter, the New Project dialog box shows the types of project you can build. Among the choices is Application, which lets you build a standard executable and dynamic libraries for either Pcode or machine code executables. The other choices available depend on which generators are installed. Each generator will produce a different set of output file types. When you deploy generated components, you may need to deploy the PowerBuilder virtual machine (PBVM60.DLL) on the same machine.</p> <p>In addition to the Application generator, the dialog box also displays the Proxy library generator, which builds a Proxy object that you can deploy with the client in a distributed PowerBuilder application. When you install other generators, they will appear in this dialog box.</p> <p>When you select a generator, the Select Objects dialog box opens so that you can choose the objects you want to use to generate a component. Depending on the generator, you may also need to set some properties of the component.</p>

Documentation For more information about generating Proxy objects and C++ classes, see *Application Techniques*. For information about other generators, see the documentation provided when the generator is available.

Cross-platform support

PowerBuilder 6.0 is available on two additional UNIX platforms:

- ◆ HP-UX Version 10.20
- ◆ IBM AIX Version 4.1.5

Changes have been made in the following features to make it easier for you to move applications among platforms:

- ◆ Font mapping
- ◆ PowerBuilder units

Font mapping

Description PowerBuilder provides font mapping files you can use to determine which fonts are used for your application. When PowerBuilder or a PowerBuilder application is first started, it reads and saves information about the available fonts on the system, then reads any font mapping specifications from initialization files and maps fonts as specified.

Purpose Using font mapping files gives you more control over how objects display when you deploy them on different machines or different operating systems.

Usage Font mapping files are provided in the following locations:

Platform	Name	Location
Windows	PBFNT60.INI	Powersoft SHARED directory
Macintosh	Powersoft Font Preferences	System Folder:Preferences
UNIX	.pbfnt60.ini	PowerBuilder install directory. The file should be copied to each user's home directory

The font mapping file has a [FontSubstitutes] section containing lines like the following (the default mappings are different on each platform):

```
Times=Times New Roman
```

This line specifies that if the Times font is used in an application and is not available on the deployment machine, the Times New Roman font should be used instead. You can modify or add to the default mappings to meet your own needs.

On Windows, if a font is not available and you don't provide a mapping for it in PBFNT60.INI, Windows tries to find a font that matches the characteristics of the missing font.

On Macintosh and UNIX, the font mapping file supplements or overrides the existing mechanisms (an internal table on the Macintosh and the .WindU file on UNIX).

Documentation *See Application Techniques.*

PowerBuilder units

Description In PowerBuilder 6.0, the size of PowerBuilder units (PBUs) is defined in terms of logical inches instead of as a fraction of the size of the system font. The size of a logical inch is defined by your operating system as a specific number of pixels. The number is dependent on the display device. For example, on a Windows 95 system using Small Fonts on a standard VGA display, the number of pixels per logical inch is 96.

Purpose This change prevents undesirable resizing of the information in a window or DataWindow when deploying applications on different platforms. Previously, when an application was deployed on a system that had a different system font size from that on which the application was developed, information in windows and DataWindows was compressed if the system font on the deployment system was smaller or spread out if it was larger, because the size of a PBU was determined by the system font size.

Usage PBUs are the units used for the design of windows, visual user objects, and optionally DataWindows. This change affects windows and DataWindows designed using PBUs that were developed under the Macintosh and Solaris operating systems, and also those developed under Windows if a system setting of Large Fonts was used by the developer. Controls on these objects may need to be resized after migration. Once migrated and fixed, they display correctly on all platforms.

Documentation *See Application Techniques.*

Internationalization

Separate versions of PowerBuilder will provide support for developers working on these international platforms:

- ◆ Unicode Standard
- ◆ Japanese DBCS
- ◆ Arabic and Hebrew right-to-left operating systems

Localized deployment kits will be available for Windows platforms after the first release of PowerBuilder 6.0 as part of the International Developer Resources. The International Developer Resources will also contain localized versions of the PowerBuilder Foundation Class Library (for PowerBuilder Enterprise and Professional editions) and a new translation tool:

- ◆ Translation Toolkit

Unicode Standard

Description	A new version of PowerBuilder for the Windows NT 4.x platform will support the Unicode Standard format UTF-16, a character encoding system that has the capacity to encode all characters used for written languages throughout the world.
Purpose	Use PowerBuilder for Unicode to develop applications in any of the character sets supported by the Unicode Standard.
Usage	<p>PowerBuilder supports the full Unicode Standard (UTF-16), which uses two full bytes to store each character. In PowerBuilder for Unicode, text in dialog boxes and other built-in objects in the graphical user interface displays in English, but everything you type in is Unicode.</p> <p>You can migrate applications from the ANSI version of PowerBuilder to the Unicode version. When you open a PBL created in the ANSI version of PowerBuilder, PowerBuilder for Unicode opens a dialog box in which you can specify the library list and the directory where the Unicode version of the application will be saved. The application is saved with the extension PUL, for PowerBuilder Unicode Library.</p> <p>A new menu item in the Library painter in PowerBuilder for Unicode lets you migrate from the Unicode version back to the ANSI version. If you add characters that are not supported in the ANSI version, the results are unpredictable.</p>

You must deploy applications built with PowerBuilder for Unicode with the Unicode deployment DLLs as 32-bit applications on Windows NT 4.x. The Unicode deployment DLLs are included with PowerBuilder for Unicode.

Functions

Two new functions enable conversion to and from Unicode characters:

ToAnsi

ToUnicode

The ANSI string data is stored in a binary large object.

Documentation

For more information about functions, see the *PowerScript Reference*. For more information about internationalization, see *Application Techniques*.

Japanese DBCS

Description

The Japanese version of PowerBuilder is compiled under Japanese Windows and is available as a separate 32-bit executable for the Japanese versions of Windows 95 and Windows NT.

Purpose

Use the Japanese version of PowerBuilder to develop applications in Japanese. You will be able to build applications for deployment on both 16-bit and 32-bit versions.

Usage

Double Byte Character Support (DBCS) is currently enabled only in the Japanese versions of PowerBuilder and InfoMaker.

In the Japanese versions, text in the graphical user interface and everything you type in displays in Japanese, and the string-handling functions are double-byte aware.

You must deploy applications built with the Japanese DBCS version of PowerBuilder with PowerBuilder Japanese deployment DLLs on Japanese Windows, Windows 95, or Windows NT.

You can migrate applications from the ANSI version of PowerBuilder to the Japanese version. If you add characters that are not supported in the ANSI version, they will not convert correctly if you migrate from Japanese to ANSI.

Documentation

For more information about functions, see the *PowerScript Reference*. For more information about internationalization, see *Application Techniques*.

Arabic and Hebrew right-to-left operating systems

Description PowerBuilder Enterprise provides support for the Arabic and Hebrew languages when run on an Arabic- or Hebrew-enabled version of Windows 95. Support includes new functions and a new EditMask character to handle Arabic or Hebrew characters and numbers.

Purpose Use the Arabic and Hebrew features in PowerBuilder to develop applications specifically for those languages. You will be able to build applications for deployment on both 16-bit and 32-bit versions of Windows.

Usage Applications built using the Arabic and Hebrew features in PowerBuilder must be deployed with the PowerBuilder Arabic or Hebrew deployment DLLs on workstations running the appropriate Arabic-enabled or Hebrew-enabled version of Windows.

On an Arabic-enabled or Hebrew-enabled version of Windows, you can display text in right-to-left order and test for Arabic or Hebrew characters in strings. You can migrate applications from the ANSI version of PowerBuilder to the Hebrew-enabled or Arabic-enabled versions and vice versa. Right-to-left support is lost when you migrate to the ANSI version.

A new mask character (b) in the EditMask control allows the entry of Arabic characters when you run PowerBuilder on the Arabic-enabled version of Windows and Hebrew characters when running on the Hebrew-enabled version. This mask is useful when a single-letter prefix is required for an ID or when a single character entry is required.

To use the mask, add an EditMask control to a window and open its property sheet. On the Mask property page, select String from the Type dropdown listbox and then select b from the Masks dropdown listbox.

Functions Six new functions have been added:

IsAllArabic	IsAllHebrew
IsArabicAndNumbers	IsHebrewAndNumbers
IsAnyArabic	IsAnyHebrew

For more information For more information about functions, see the *PowerScript Reference*. For more information about internationalization, see *Application Techniques*.

Translation Toolkit

Description The Translation Toolkit is a set of tools that help you translate any PowerBuilder or InfoMaker Version 6 application.

Availability

The Translation Toolkit for PowerBuilder 6.0 will be available three to six months after the release of PowerBuilder 6.0.

The toolkit includes:

- ◆ Translator tool for translating extracted phrases
- ◆ Project Translator tool for substituting the translations for the original phrases in the project libraries
- ◆ Database Administration tool for importing glossaries, defining new languages, and deleting unused phrases and images in the translation database
- ◆ Text Analyzer tool for saving data about controls you resize in PowerBuilder so translated text fits
- ◆ Microsoft International Glossaries for accessing the translations of phrases found in Microsoft products
- ◆ Phrase Extractor tool for creating a translation project and extracting phrases from the project libraries

Usage After the translations are substituted for the original phrases in the project libraries, you build the translated application on the target deployment platform (Windows 3.1, 95, or NT; Macintosh; or UNIX) and then deploy the translated application.

Documentation See the *Translation Toolkit User's Guide*.

Database connectivity

PowerBuilder Version 6.0 has an improved user interface for database profiles. It also provides many other new and enhanced database connectivity features, including:

- ◆ New database interfaces, including INFORMIX 7 and Oracle 7.3
- ◆ New DBParm parameters
- ◆ Support for Sybase Open Client 11.1

FOR INFO For a complete list of new database connectivity features, see online Help.

Improved user interface for database profiles

Description	PowerBuilder provides an improved user interface for creating and managing database profiles. A database profile is a named set of parameters stored in your PowerBuilder or InfoMaker initialization file that defines a connection to a particular database in the development environment.
Purpose	<p>The improved user interface for creating and managing database profiles in PowerBuilder makes it easier for you to:</p> <ul style="list-style-type: none">◆ See a list of your database profiles organized by Powersoft database interface◆ Access the Configure ODBC dialog box to create and manage ODBC data source definitions◆ Supply values for the connection options required by your Powersoft database interface◆ Set DBParm parameters in the development environment without having to manually edit a complex and lengthy DBParm string◆ Generate correct PowerScript connection syntax in the PowerBuilder development environment for use in your PowerBuilder application script

Usage

The new user interface for database profiles changes the steps you follow to define, establish, and manage database connections in PowerBuilder. The components of the new user interface are:

- ◆ Database Profiles dialog box

The main Database Profiles dialog box has been redesigned using a tree control format so you can easily see each installed Powersoft database interface and its database profiles. You can create, edit, and delete database profiles from this dialog box. And when the ODBC interface or one of its profiles is selected, you can access the Configure ODBC dialog box to create, edit, or delete an ODBC data source definition.

- ◆ Database Profile Setup dialog box for each interface

Each Powersoft database interface now has its own Database Profile Setup dialog box where you can set interface-specific connection options and DBParm parameters. For example, if you select the SYC interface and click New in the Database Profiles dialog box, a dialog box displays with settings only for those connection options that apply to the SYC interface.

The Database Profile Setup dialog box groups similar DBParm parameters on the same tabbed page and lets you easily set values for DBParms using checkboxes, dropdown listboxes, and textboxes. 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 into your PowerBuilder script.

Documentation

See *Connecting to Your Database*.

OLE enhancements

PowerBuilder 6.0 includes changes in the following support for OLE:

- ◆ OLE server features
- ◆ OLE control container features
- ◆ OLE error handling

OLE server features

- Description** PowerBuilder 6.0 support for OLE servers includes new functions to provide support for DCOM, event programming for OLEObjects, and the ability to time out calls to a server. OLE automation performance has been improved, and enumerated types for any OLE automation server display in the PowerBuilder Browser.
- Functions** Support for DCOM is provided by the `ConnectToRemoteObject` and `ConnectToNewRemoteObject` functions. The new functions let you pass the name of a remote host where a COM server resides when you connect to an OLE object.
- You can implement events for an OLEObject by creating a user object that is a descendant of OLEObject. `SetAutomationPointer` assigns an OLE automation pointer into the descendant so that it can use OLE automation.
- `SetAutomationTimeout` lets you set the timeout period for OLE procedure calls from a PowerBuilder client to a server. The default timeout period is five minutes.
- Documentation** For more information about new functions, see the *PowerScript Reference*. For information about using PowerBuilder with OLE, see *Application Techniques*.

OLE control container features

- Description** PowerBuilder 6.0 support for OLE control containers includes extended control properties, an implementation of the `IOleContainer` class, and message reflection.
- Usage** An OLE control can determine its location and modify it at execution time using its extended control properties. In PowerBuilder 6.0, the properties you can set are X (alias Left), Y (alias Top), Width, and Height. These properties are real (R4) values, truncated to long integers when set. They are measured in PBU's and are accessible using C++ and the OLE `IOleControlSite` interface.
- PowerBuilder implements the `IOleContainer` class at the window level to enable OLE controls to find out about their siblings. You can use the OLE `EnumObjects()` method to access the OLE enumerator.
- PowerBuilder OLE control containers perform their own message reflection for a specific set of messages. This feature eliminates the execution-time overhead that would be required if the OLE control had to create a reflector window to handle these messages.

Documentation For information about using PowerBuilder with OLE, see *Application Techniques*.

OLE error handling

Description An OLE control can provide its own stock error event that is executed when the OLE control calls the MFC FireError method.

Purpose You can use this event in PowerBuilder to separate error-handling code for errors triggered when FireError is called from error-handling code for other OLE exceptions. If no stock error event has been defined, the FireError method triggers PowerBuilder's ExternalException event.

Usage A stock error event is an entry in the control's IDL or ODL file created by the MFC Class Wizard. Internally, PowerBuilder translates the stock error event into an OCX_Error event to avoid name conflicts with PowerBuilder's Error event.

The OCX_Error event has the following signature:

```
void OCX_Error ( short Number, REF string Description, long Scode,
                string Source, string Helpfile, long HelpContext, REF boolean
                CancelDisplay )
```

You can set the CancelDisplay variable to TRUE to cancel the display of any MFC error message. Alternatively, you can supply a different description. For example:

```
sle_desc.text = description
sle_source.text = source
sle_helpfile.text = helpfile
sle_excep.text = "scode " + string(scode)
sle_result.text = "number " + string(number)
sle_helpid.text = "HelpID " + string(helpcontext)

canceledisplay = FALSE
description = "This is my new error description"
```

Documentation For information about using PowerBuilder with OLE, see *Application Techniques*.

Standard source control API

Description	PowerBuilder provides a standard version control interface (the PowerBuilder SCC API) that is based on the Microsoft Common Source Code Control Interface Specification, Version 0.00.0823.
Purpose	<p>You can use the PowerBuilder SCC API with any version control system that implements features defined in the Microsoft specification. Some of the vendors that provide an SCC API are INTERSOLV (PVCS), Platinum (CCC/Harvest), and Rational (ClearCase, formerly an Atria product).</p> <p>Among the advantages of using this interface with a version control system are that the PowerBuilder SCC API:</p> <ul style="list-style-type: none">◆ Synchronizes the object in the public library with the archive as part of the check-out procedure◆ Will be modified in future PowerBuilder releases to support functions added to version control systems to augment this Microsoft specification <p>Some of the new features this interface offers are:</p> <ul style="list-style-type: none">◆ More comprehensive registration reporting functionality◆ A new trace logging capability◆ The ability to require check-in comments◆ The ability to compare an object in the work library with the object in the archive and display differences
Usage	In the Library painter, select Source>Connect, then select SCC API in the Vendor listbox. If you have more than one version control system installed and configured for the PowerBuilder SCC API, you will be prompted to select one system.
Documentation	See <i>Version Control Interfaces</i> .

Developer productivity

The following features increase developer productivity:

- ◆ Tracing and profiling
- ◆ Debugger interface
- ◆ DataWindow features
- ◆ PFC enhancements
- ◆ Ease-of-use and language features
- ◆ Deployment and execution time enhancements
- ◆ Component Gallery
- ◆ HOW Learning Edition
- ◆ PSChart

Tracing and profiling

Description	The tracing and profiling feature allows you to collect and analyze information about the execution of your PowerBuilder application.
Purpose	The tracing and profiling feature helps you identify areas that you should rewrite to improve performance and find errors in the application's logic.
Usage	<p>There are three ways to collect trace data:</p> <ul style="list-style-type: none">◆ You can use the Profiling page on the System Options dialog box to enable tracing◆ You can insert trace functions in your application scripts at the beginning and end of the code you want to trace◆ You can add a window to your application that lets you turn tracing on and off as you run the application (<code>w_starttrace</code> is provided in the <code>Profile.pbl</code>) <p>To analyze the collected data, you can use the Application Profiler or the profiling and trace system objects to build your own analysis and/or display applications. The Application Profiler is also provided as a sample application.</p>
Objects	A number of new system objects have been added to support tracing and profiling:

Category	New objects
Objects that provide the functions and properties to create a performance analysis model from your trace file and to extract information from that model	ProfileCall Profiling ProfileRoutine ProfileClass ProfileLine
Objects that provide the functions and properties to create a nested tree model from your trace file and to extract information from that model	TraceTree TraceTreeGarbageCollect TraceTreeObject TraceTreeError TraceTreeLine TraceTreeRoutine TraceTreeESQL TraceTreeNode TraceTreeUser
Objects that provide the functions and properties to access the data in your trace file sequentially	TraceActivityNode TraceFile TraceObject TraceBeginEnd TraceGarbageCollect TraceRoutine TraceError TraceLine TraceUser TraceESQL

Functions

A number of new functions have been added:

Category	New functions
Functions that collect data in a trace file	TraceBegin TraceEnableActivity TraceOpen TraceClose TraceEnd TraceUser TraceDisableActivity TraceError

Category	New functions
Functions that build models and/or extract information	BuildModel GetChildrenList OutgoingCallList ClassList IncomingCallList Reset Close LineList RoutineList DestroyModel NextActivity SetTraceFileName EntryList Open SystemRoutine

Documentation For more information about using the Application Profiler, see the online Help provided with the application. For more information about using tracing and profiling, see the *PowerBuilder User's Guide*. For more information about tracing and profiling functions, see the *PowerScript Reference*. For more information about tracing and profiling objects, see *Objects and Controls*.

Debugger interface

- Description** The PowerBuilder debugger supports breakpoint, watchpoint, and stepping capabilities and displays multiple views of your application in panes that you can move, resize, and overlap for a customized layout. You can also use just-in-time debugging to open the Debug window if a system error occurs or you notice problems while running your application from the Run button.
- Usage** When you run an application in debug mode, use conditional and occasional breakpoints and the ability to set a breakpoint when a variable changes to fine-tune where you suspend execution. Then set variables and expressions you want to watch, and step through your code examining variables and memory objects. You can step into, over, and out of functions, run to the location where you set the cursor, and set the next statement you want executed.
- Views** Multiple views of the state of the application make it easy to monitor changes. Use mouse actions (including drag and drop) to set, enable, disable, or clear breakpoints and watchpoints and to change the context of the application:

View	What it shows
Breakpoints	A list of breakpoints with indicators showing whether the breakpoints are currently active or inactive
Call Stack	The sequence of function calls leading up to the function that was executing when a breakpoint was hit
Objects in Memory	An expandable list of objects currently in memory
Source	The full text of the current script or any script or function in the application
Source Browser	An expandable hierarchy of objects in your application
Source History	A list of the scripts that have been displayed in the Source view
Variables	An expandable list of all the variables in scope, in separate views or views combining local, global, instance, parent, and shared variables
Watch	A list of variables or expressions you have selected to watch as the application proceeds

Documentation

See the *PowerBuilder User's Guide*.

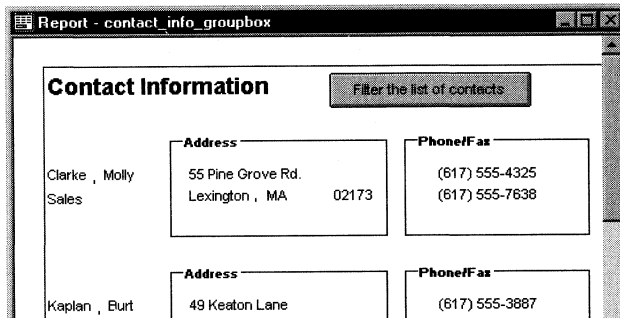
DataWindow features

The following enhancements have been made to DataWindows:

- ◆ Button object
- ◆ GroupBox object
- ◆ Centered checkboxes
- ◆ Scrollbar support in print preview
- ◆ RowFocusChanging event
- ◆ Excel 5 added to SaveAs PowerScript function
- ◆ SaveAsAscii PowerScript function
- ◆ Border painting enhancements
- ◆ Improved handling of display formats
- ◆ Improved n-up row selection

Button object

- Description** The Button object is a command (or picture) style button that can be placed in a DataWindow object. When clicked at execution time, the button activates either a built-in or user-supplied action.
- Purpose** Buttons make it easy to provide command button actions in a DataWindow object (no coding is required). And using Button objects in the DataWindow object (rather than CommandButton controls in a window) ensures that the actions appropriate to the DataWindow object are included in the object.
- Illustration** This example shows a button placed in a report (a nonupdatable DataWindow object). Clicking the button brings up the Filter dialog box, where the user can specify a filter to be applied to the currently retrieved data:



- Usage** In the DataWindow painter workspace, select Objects>Button from the menu bar and click in the workspace.
- To control whether the button displays in print preview and prints on the printed DataWindow object, use the property sheet for the DataWindow object itself.
- What happens at execution time depends on whether Suppress Event Processing is on or off for the button:

If Suppress Event Processing is	When the button is clicked
On	The action assigned to it is executed
Off	The new ButtonClicking event is fired. Code in the ButtonClicking event (if any) is executed. If the return code is 0, the action assigned to the button is then executed. After the action is executed (or if the return code from the ButtonClicking event is 1), the ButtonClicked event is fired

Documentation For information about the object, see *Objects and Controls* or online Help. For usage information, see the *PowerBuilder User's Guide*.

GroupBox object

Description The GroupBox object is a static frame used to group and label a set of objects in a DataWindow object.

Purpose The Group Box is a visual enhancement that improves the layout of information in a DataWindow object.

Illustration The illustration for the Button object on page 34 shows two groupboxes, both with labels, in a report (nonupdatable DataWindow object): the Address groupbox and the Phone/Fax groupbox.

Usage In the DataWindow painter workspace, select Objects>Group Box from the menu bar and click in the workspace.

Documentation For information about the object, see *Objects and Controls* or online Help. For usage information, see the *PowerBuilder User's Guide*.

Centered checkboxes

Description Checkboxes without text can be centered.

Purpose This feature makes it easy to create a neat layout of checkboxes in a DataWindow object.

Usage An easy way to take advantage of checkbox centering is to first work with the column header text object and the column object being displayed as a checkbox. Make these two objects the same size and left aligned. Then center both the column header contents and the column object being displayed as a checkbox using the Stylebar or the objects' property sheets. The checkboxes will be centered under the centered column header text.

For centering to work, the Left Text checkbox in the edit style tab page must not be checked and the checkbox must not have associated text.

Scrollbar support in print preview

Description The scrollbars used for viewing DataWindow objects in print preview mode now scroll through the complete DataWindow object. Support for changing pages and for moving to particular pages easily has also been added.

Purpose This feature improves the behavior of print preview and provides a way to scroll if you are displaying in an Internet browser.

Usage Scrolling behavior is described in the following table:

Action	What happens
Line Down	Display goes down by a line size (1/5 of the screen height). If it goes beyond the print page, the top of the next page displays
Line Up	Display goes up by a line size (1/5 of the screen height). If it goes beyond the print page, the bottom of the previous page displays
Page Down	Display goes down by the screen height (or less) until it reaches the bottom of the page. The following Page Down goes to the top of the next page
Page Up	Display goes up by the screen height (or less) until it reaches the top of the page. The following Page Up goes to the top of the previous page
Top	Display goes to the top of the first page of the DataWindow object
Bottom	Display goes to the top of the last page of the DataWindow object
Dragging the thumb (small box) in the scrollbar	While the thumb is being dragged, the display is unchanged. The page number corresponding to the current location of the thumb displays on the MicroHelp bar and in a small window next to the thumb in the scrollbar. Display changes to new page when mouse is released

RowFocusChanging event

Description The RowFocusChanging event occurs when the DataWindow current row is about to change (the current row of the DataWindow is not necessarily the same as the current row in the database).

The RowFocusChanging event occurs just before the RowFocusChanged event.

Purpose You can place code in the RowFocusChanging event to clean up resources associated with the current row or to avoid an error by providing a default resource to be used if a resource required by the next row is unavailable.

Usage Typically this event will be coded to respond to a mouse click or keyboard action that would change the current row in the DataWindow object.

Documentation See the *PowerScript Reference*.

Excel 5 added to SaveAs PowerScript function

Description Syntax 1 (for DataWindows and DataStores) of the SaveAs PowerScript function has a new value for the *saveastype* argument: Excel5!

Purpose This addition supports the ability to export data into the Excel 5 format, which uses a different storage method from earlier versions of Excel.

Documentation See the *PowerScript Reference*.

SaveAsAscii PowerScript function

Description The SaveAsAscii function saves the contents of a DataWindow or DataStore into a standard ASCII text file.

Usage SaveAsAscii is a cross between the SaveAs (Text!) function and the SaveAs (HTMLTable!) function with additional arguments. You can save computed columns and headers to the ASCII file. Arguments allow you to control how contents are separated and delimited in the ASCII file. PowerBuilder assigns a cell for each object in the DataWindow (which can include computed columns and group totals). If a cell is empty, PowerBuilder puts the quote character between the separator characters in the output file.

Documentation See the *PowerScript Reference*.

Border painting enhancements

DataWindow border painting has been refined to apply zooming and device units conversion to border sizes, resulting in the following improvements:

- ◆ Borders on objects in DataWindows change size appropriately when zooming in preview
- ◆ Borders print more accurately
- ◆ Cell borders stay within cell boundaries in grid DataWindows

Improved handling of display formats

When a column with both a DropDownDataWindow edit style and display format does not have focus, the display format is used for displaying the data value. When the column has focus, the raw data value is displayed.

Improved n-up row selection

The way row selection displays in an n-up DataWindow object has been improved. When a specific row is selected, only that row will be highlighted. Formerly that row and all other rows occurring on the same detail line as the selected row were highlighted.

PFC enhancements

Description

PFC (PowerBuilder Foundation Class Library) includes several new objects and services, including:

- ◆ DataWindow resize service (where you provide resize capabilities to the objects displayed in a DataWindow)
- ◆ DataWindow properties service (which includes dialog boxes you display to view and modify DataWindow services, contents, and properties)
- ◆ DataStore multitable, print preview, and report services
- ◆ Calendar
- ◆ Calculator
- ◆ Progress bar
- ◆ Splitbar service
- ◆ Application preference service
- ◆ Window most-recently-used service
- ◆ Logical unit of work service
- ◆ MetaClass service

PFC also includes the Library Extender, which you use to create intermediate extension level objects automatically, and the message manager, which you use to access and update the PFC messages table.

Usage enhancements Most objects use constants to signify typical return codes. This practice enables you to write more readable code.

Enhancements have also been made to the following:

- ◆ W_master pfc_Save process
- ◆ Window status bar service
- ◆ Error message service
- ◆ DataWindow linkage service
- ◆ DataWindow caching
- ◆ Row selection service
- ◆ Security service
- ◆ Code examples

Objects

The following table lists some of the new objects and services:

Object	Description
u_st_splitbar	Splitbar object
n_cst_dwsrv_resize	DataWindow resize service
n_cst_dropdown	Dropdown service
u_calculator	Calculator object
u_calendar	Calendar object
n_cst_luw	Logical unit of work service
u_base	Ancestor for all custom visual user objects
n_cst_appreference	Application preference service
u_progressbar	Progress meter
n_cst_mru	Window most-recently-used service
u_lvs	Service-based ListView object
u_tvsv	Service-based TreeView object
n_cst_metaclass	MetaClass service

Documentation

See the *PFC User's Guide* and the *PFC Object Reference*.

Ease-of-use and language features

The following features have been added to make PowerBuilder easier to use:

- ◆ Nonmodal Browser with context-sensitive Help
- ◆ Last compiled timestamp in the Library painter
- ◆ Flat toolbars
- ◆ Support for the IntelliMouse Pointing Device
- ◆ Enhanced source control for Application objects
- ◆ Timing object
- ◆ Updated control property arrays
- ◆ AncestorReturnValue variable

There are also several new objects that provide class definition information for developers of tools and object frameworks:

- ◆ Class definition information

Nonmodal Browser with context-sensitive Help

Description	The PowerBuilder Browser is nonmodal: it remains open when you move between painter workspaces. Actions that were previously available as buttons in the Browser, as well as new actions, have been moved to the popup menus. You can open an object in a painter from its popup menu.
Purpose	You can use the information available from the Browser as you move from one PowerBuilder painter to another and access context-sensitive Help for objects, controls, and functions directly from the Browser.
Usage	<p>In the PowerScript painter PainterBar, click Browse Object. The Browser opens with the current object selected. If the Browser is already open, it changes to display the currently selected object.</p> <p>In the PowerBar, click Browser. The Browser opens with the current application object selected. If the Browser is already open, it does not change.</p> <p>From the popup menu for items in the Browser, you can select some or all of the following items:</p>

Menu item	Select to
Edit	Opens the selected object, or the parent of the selected control, in the appropriate painter

Menu item	Select to
Copy	Copies the selected object or control to the clipboard
Paste	Pastes the object or control from the clipboard into the script
Expand All	Expands the objects or controls within the selected object
Regenerate	Regenerates descendants of the selected object
Show Inherited	Toggles between displaying inherited properties, functions, events, and variables (on) and displaying only items defined in that object (off)
Show Legend	Toggles between displaying (on) and hiding (off) the text describing the selected property, function, event, or variable
Show Hierarchy	Toggles between displaying objects in an inheritance hierarchy (on) and an alphabetical listing (off)
Document	Copies information about the selected object to a preview window with printing, exporting, and copying options
Help	Displays context-sensitive Help

Last compiled timestamp in the Library painter

Description	Developers can now display the date each object was last compiled.
Purpose	With this additional information, you can identify the most recently compiled PowerBuilder objects in the Library painter. If an object has not yet been compiled, no compile date displays.
Usage	In the Library painter, select Design>Options from the menu bar, then check the Compilation Date checkbox.

Flat toolbars

Description	<p>PowerBuilder toolbars look and behave like those in Microsoft Office 97:</p> <ul style="list-style-type: none"> ◆ Active toolbar items appear as 2D graphics until you move the mouse pointer over them ◆ Disabled menu items have a silhouetted, engraved appearance and do not respond to the mouse pointer ◆ A grab bar is available for moving and docking the toolbars
-------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Tooltips and MicroHelp status are available, and menu items can be grouped by vertical lines.

Usage Toolbars defined in the Menu painter have the flat appearance.

Support for the IntelliMouse Pointing Device

Description The IntelliMouse Pointing Device is a new piece of hardware that can be used as a standard 2-button mouse on 16-bit operating systems or as a 3-button mouse with the added capability of a scroll wheel on 32-bit operating systems—clicking on the wheel simulates the third (middle) mouse button. This feature is not currently available on the Macintosh and UNIX.

Usage **In DataWindow objects** Using the IntelliMouse Pointing Device, a user can scroll a DataWindow object (at execution time or in preview) by rotating the wheel. The user can also zoom a DataWindow object larger or smaller by holding down the CTRL key while rotating the wheel.

In standard controls such as TreeViews and ListViews Using the IntelliMouse Pointing Device, a user can scroll a TreeView or a ListView at execution time by rotating the wheel.

Enhanced source control for Application objects

Description Modifying an Application object registered in a version control system is as simple as modifying any other registered object.

When you check out a registered Application object, PowerBuilder performs some background tasks to make the Application object in your work library the current application. When you check in an Application object, PowerBuilder performs similar tasks to make the Application object in the public library your current application.

Throughout the modification process, PowerBuilder remains connected to the version control system.

Purpose Now you can modify the Application object in PowerBuilder more efficiently.

Usage Always check out and check in the Application object by itself.

To check out the Application object, select Source>Check Out from the Library painter menu bar.

To check in the Application object, select Source>Check In from the Library painter menu bar.

Documentation See the *PowerBuilder User's Guide*.

Timing object

Description The Timing object is a nonvisual object type that provides a Timer event that is not associated with a visual window.

Purpose You can use a Timing object in any application where you want to use a Timer event without associating it with a window. For example, you may want to use a Timing object in a distributed PowerBuilder server that uses a shared object to access a database at regularly timed intervals.

Usage Create a new standard class user object that inherits from the Timing object and write application-specific code in the user object's Timer event. Then create an instance of the user object. To activate the timer, call the Start function specifying the number of seconds you want between Timer events. The Timer event is executed as soon as possible after the interval has passed.

Functions Two functions have been added to start and stop a Timing object:

Start
Stop

Events The Timer event is now available for Timing objects.

Documentation For information about functions and events, see the *PowerScript Reference*. For information about using the Timing object, see *Application Techniques*.

Updated control property arrays

Description The control property arrays of windows and Tab controls are updated when you use PowerScript functions to add controls or tab pages.

Purpose You can now use the built-in control property array to reference all the controls in a window or all the pages in a Tab control. (Previously, the control property array only kept track of controls or tab pages added in the Window painter.)

Migration note To avoid conflicts, change code that uses a user-defined array to keep track of controls or tab pages to use the built-in control property array instead.

Usage The control property of a window is an array that keeps track of the controls on the window. The control property of a Tab control is an array that keeps track of the tab pages on the Tab control. You can refer to each control or tab page by its index in the control array.

When you call `OpenUserObject` or `OpenTab`, the control property array grows by one element. The new element is a reference to the newly opened object. For example, the following statement adds a new tab in the second position in the Tab control:

```
tab_1.OpenTab(uo_newtab, 2)
```

The second element in the control array for `tab_1` now refers to `uo_newtab`, and the index into the control array for all subsequent tab pages becomes one greater.

When you call `CloseUserObject` or `CloseTab`, the size of the array is reduced by one and the reference to the user object or page is destroyed. If the closed tab was not the last element in the array, the index for all subsequent tab pages is reduced by one.

The `MoveTab` function changes the order of the pages in a Tab control and also reorders the elements in the control array to match the new tab order.

Functions

The following functions are affected by this feature:

- CloseTab
- OpenTab
- OpenUserObject
- CloseUserObject
- OpenTabWithParm
- OpenUserObjectWithParm
- MoveTab

Documentation

For information about using the Control property array, see *Application Techniques*. For information about functions and events, see the *PowerScript Reference*.

AncestorReturnValue variable

Description

When you extend an event script in a descendent object, the compiler automatically generates a local variable called `AncestorReturnValue` that you can use if you need to know the return value of the ancestor event script. The variable is also generated if you override the ancestor script and use the `CALL` syntax to call the ancestor event script.

Purpose

You can now get the return value of an ancestor event script without explicitly calling the ancestor script, passing it the appropriate arguments and capturing its return value in a local variable.

- Migration note** The compiler generates the `AncestorReturnValue` variable automatically when a script is compiled that extends an event or that calls its ancestor event using the `CALL` syntax. If a script already has a local variable named `AncestorReturnValue`, you will see a compiler error the first time the script is compiled, which may be when you migrate an application to PowerBuilder 6.0. To avoid name conflicts, you should rename the local variable.
- Usage** Sometimes you want to perform some processing in an event in a descendent object, but that processing depends on the return value of the ancestor event script. You can use a local variable called `AncestorReturnValue` that is automatically declared and assigned the value of the ancestor event.
- The first time the compiler encounters a `CALL` statement that calls the ancestor event of a script, the compiler implicitly generates code that declares the `AncestorReturnValue` variable and assigns to it the return value of the ancestor event. If you override the ancestor event script, you have to insert the `CALL` statement explicitly; but if you extend the ancestor event script, the `CALL` statement is generated by the Script painter.
- The data type of the `AncestorReturnValue` variable is always the same as the data type defined for the return value of the event. The arguments passed to the call come from the arguments passed to the event in the descendent object.
- Extending event scripts** The `AncestorReturnValue` variable is always available in extended event scripts. When you extend an event script, the Script painter generates the following syntax and inserts it at the beginning of the event script:
- ```
CALL SUPER::event_name
```
- You only see the statement if you export the syntax of the object.
- Overriding event scripts** The `AncestorReturnValue` variable is only available when you override an event script after you call the ancestor event using the `CALL` syntax:
- ```
CALL SUPER::event_name
```
- or
- ```
CALL ancestor_name::event_name
```
- The compiler does not differentiate between the keyword `SUPER` and the name of the ancestor. The keyword is replaced with the name of the ancestor before the script is compiled.
- The `AncestorReturnValue` variable is only declared and a value assigned when you use the `CALL` event syntax. It is not declared if you use the new event syntax:

```
ancestor_name::EVENT event_name ()
```

**Example** You can put code like the following in an extended event script:

```
IF AncestorReturnValue = 1 THEN
 // execute some code
ELSE
 // execute some other code
END IF
```

You can use the same code in a script that overrides its ancestor event script, but you must insert a **CALL** statement before you use the **AncestorReturnValue** variable:

```
// execute code that does some preliminary processing
CALL SUPER::ue_myevent
IF AncestorReturnValue = 1 THEN
 ...
```

Documentation

For more information about using the **AncestorReturnValue** variable, see *Application Techniques* and the *PowerScript Reference*.

## Class definition information

Description

PowerBuilder 6.0 has several new objects that provide information about an object's class definition and its variables and scripts. A class definition object is a PowerBuilder object that provides information about the class definition of another PowerBuilder object. You can examine a class definition in a PowerBuilder library or the class definition associated with an instantiated object.

Purpose

Class definition information is important for developers of tools and object frameworks. Developers can inspect objects to produce reports or to define objects with similar characteristics. Class information is not usually used in typical business applications.

Objects

These new objects provide class definition information:

| Object          | Description                                                                                                                                                              |
|-----------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| TypeDefinition  | Provides information about a data type and is the ancestor of several more specific definition objects: ClassDefinition, SimpleTypeDefinition, and EnumerationDefinition |
| ClassDefinition | Provides the object's class name, library, ancestor, parent, variables, and scripts and the objects it contains                                                          |

| Object                | Description                                                                                                                                                                                                           |
|-----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| SimpleTypeDefinition  | Provides information about a simple data type, such as integer, string, blob, and Any                                                                                                                                 |
| EnumerationDefinition | Provides information about an enumerated data type                                                                                                                                                                    |
| VariableDefinition    | Provides information about a variable or property associated with a class definition, including its access level, scope, and whether it is an array                                                                   |
| ScriptDefinition      | Provides information about a function or event script associated with a class definition, including its arguments and return type, whether it is external, and whether it is defined locally or in an ancestor object |

Other objects and enumerated data types provide values for properties of these objects.

#### Functions and properties

Several new properties and functions give you access to definition information for your objects:

- ◆ PowerObject has a new property called ClassDefinition, making a ClassDefinition object available for any instantiated object
- ◆ New global functions called FindTypeDefinition and FindClassDefinition get a type or class definition object for an entry in a PowerBuilder library
- ◆ New global function FindFunctionDefinition gets script information for a global function

#### Documentation

For information about the objects, see *Objects and Controls* or online Help. For function definitions, see the *PowerScript Reference*. For usage information, see *Application Techniques*.

## Deployment and execution time enhancements

In the Project painter, you can create both machine code and Pcode executables for deployment on 32-bit and 16-bit workstations.

The following additional deployment and execution time enhancements have been added:

- ◆ Simpler deployment kit
- ◆ The Synchronizer
- ◆ PopulateError and SignalError functions
- ◆ Garbage collection
- ◆ Web.PB wizard and OLE GenReg on the PowerBar

## Simpler deployment kit

|               |                                                                                                                                                                                                                                                                                                           |
|---------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Description   | Several of the PowerBuilder shared libraries (DLLs) in the PowerBuilder deployment kit have been combined to simplify the installation of the deployment kit. DLLs that support a specific feature such as RichText or DataWindows are not required if your application does not make use of the feature. |
| Documentation | See <i>Application Techniques</i> .                                                                                                                                                                                                                                                                       |

## The Synchronizer

|             |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|-------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Description | The Synchronizer (Sync) is software that synchronizes two sets of files. Sync compares source files to destination files and then copies the latest source files to the destination so the files match.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| Usage       | <p><b>For synchronizing deployed applications</b> Sync's primary purpose is to synchronize files for your deployed PowerBuilder applications. Without Sync, it's difficult to maintain and deploy applications. With Sync, you can update your users' application files to match the latest application files—so you don't have to do it.</p> <p>When a user starts a synchronizing application, Sync runs first to synchronize application files, and then Sync runs the application. So every time a PowerBuilder application runs, the application will run with the latest version of all the files needed for the application.</p> <p><b>For synchronizing files from a Web page</b> You can use Sync to add synchronization ability to a Web page so that when users access the page, synchronization occurs.</p> <p><b>What the Synchronizer does</b> When Sync runs, it:</p> |



- 1 Looks for synchronization instructions (usually found in a Sync data file) that specify the location of the source files and the destination files.
- 2 Displays a status window and logs the execution of instructions if such instruction is provided.
- 3 Compares the source files to the destination files by checking date/time stamps, sizes, and version information.
- 4 Copies changed source files (including source files that don't exist on the destination) to the destination.
- 5 Runs an application when synchronization completes if such instruction is provided.

**Sources** The Synchronizer can synchronize files from these sources:

| Source                | Example                     |
|-----------------------|-----------------------------|
| FTP                   | FTP://FTP.POWERSOFT.COM     |
| Local drive           | C:\DIRECTORY\FILENAME       |
| Server on the network | \\SHARENAME\COMMON\FILENAME |

The Sync ActiveX can synchronize files from a local or network server or an HTTP server.

**Components** The Synchronizer has three components:

| Component               | What you do                                                                                                             | Platform                                                                                         | Filename                   |
|-------------------------|-------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------|----------------------------|
| Sync Builder            | Create and test a Sync data file (contains synchronization instructions)                                                | 32-bit only:<br>Windows 95 and<br>Windows NT<br>3.51 and 4.0                                     | SYNC.EXE                   |
| Sync runtime executable | Run the EXE and provide a reference to a Sync data file. Optionally start an application when synchronization completes | 32-bit:<br>Windows 95 and<br>Windows NT 4.0<br><br>16-bit:<br>Windows 3.1 and<br>Windows NT 3.51 | SYNCRT.EXE<br>SYNCRT16.EXE |

| Component    | What you do                                                                                                                                                | Platform                                     | Filename |
|--------------|------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------|----------|
| Sync ActiveX | Insert the Sync ActiveX in an application window or on a Web page, specify properties, and call the Execute method to execute synchronization instructions | 32-bit only:<br>Windows 95<br>Windows NT 4.0 | SYNC.OCX |

Documentation      See the *Synchronizer User's Guide*.

## PopulateError and SignalError functions

|               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|---------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Description   | <p>PowerBuilder 6.0 provides two ways to populate the Error object:</p> <ul style="list-style-type: none"> <li>◆ A new SignalError syntax                     <p>This syntax allows you to populate the Error object with a specified error number and text before the function triggers a SystemError event at the application level.</p> </li> <li>◆ A new PopulateError function                     <p>This function allows you to populate the Error object with an error number and error text without immediately triggering a SystemError event.</p> </li> </ul> |
| Purpose       | <p>The new SignalError syntax makes it easier for you to populate the Error object: you supply the error number and error text values, and the remaining structure values are filled in for you when SystemError is triggered.</p> <p>The new PopulateError function allows you to populate the Error object without triggering a SystemError event. With this function, you can populate the Error object in one section of your script and trigger a SystemError event in another section of your script.</p>                                                          |
| Usage         | <p>Use the new SignalError syntax to set values for error number and text that will be passed to the SystemError event.</p> <p>Use PopulateError to set values in the error object before triggering a SystemError event.</p>                                                                                                                                                                                                                                                                                                                                            |
| Documentation | See the <i>PowerScript Reference</i> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |

## Garbage collection

|               |                                                                                                                                                                                                                                                                                                                    |
|---------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Description   | PowerBuilder checks memory for unreferenced and orphaned objects and removes any it finds. When a reference is removed for an object, PowerBuilder counts the remaining references to that object. If there are none, the class is cleared from memory. PowerBuilder checks all the reference counts periodically. |
| Purpose       | The garbage collection feature clears unused objects from memory, ensuring more efficient memory usage. You can use it in the development stage as well as in the deployed application.                                                                                                                            |
| Usage         | Garbage collection occurs periodically in PowerBuilder. New functions allow you to use the existing period, modify the time period, or force immediate garbage collection.                                                                                                                                         |
| Functions     | Three functions have been added for this feature:<br><br>GarbageCollect<br>GarbageCollectGetTimeLimit<br>GarbageCollectSetTimeLimit                                                                                                                                                                                |
| Documentation | See Application Techniques and the <i>PowerScript Reference</i> .                                                                                                                                                                                                                                                  |

## Web.PB wizard and OLE GenReg on the PowerBar

|             |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|-------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Description | <p>This feature allows you to invoke the Web.PB wizard or the OLE GenReg utility from the PowerBuilder development environment.</p> <p>The Web.PB wizard and the OLE GenReg utility appear as two new system level toolbar buttons. They also appear on the dropdown panel and the PowerPanel. When you click the buttons, the applications display as modeless dialog boxes. If you click a button while the application is open, PowerBuilder activates the application. If you close PowerBuilder while an application is open, the application also terminates.</p> <p>The applications must be PBDs that are synchronized to the current version of PowerBuilder.</p> <p>The Web.PB wizard button appears only if you have installed Web.PB.</p> <p>This feature is not currently enabled on UNIX and is not available on the Macintosh.</p> |
|-------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

|               |                                                                                                                                                                                                                        |
|---------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Purpose       | As more and more PowerBuilder applications are deployed as components, access to supporting applications becomes more important. Use this feature for easy access to two commonly used component deployment utilities. |
| Documentation | For information about the Web.PB Wizard, see <i>Building Internet Applications with PowerBuilder</i> . For information about Ole GenReg, see <i>Application Techniques</i> .                                           |

## Component Gallery

|               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|---------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Description   | <p>PowerBuilder 6.0 includes a number of sample ActiveX component products in the Component Gallery for Powersoft Tools. These products are provided for trial use with the application development environment and can be assembled into a larger application architecture. Some of the component products require additional licensing from the manufacturer. The Component Gallery includes components for database and network access, message handling, telecommunications, multimedia functions, and much more.</p> <p>The Component Gallery for Powersoft Tools is available on Windows only.</p> |
| Documentation | See online Help for the Component Gallery.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |

## HOW Learning Edition

|             |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|-------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Description | <p>PowerBuilder Enterprise for Windows includes the Learning Edition (LE) of Riverton Software Corporation's HOW. HOW LE is a specially designed version of HOW intended to expose developers and analysts to the benefits of component-based development. HOW LE has all the graphical builder tools and all of the interfaces to third-party products that HOW Enterprise Edition has. The primary difference between HOW LE and HOW Enterprise Edition is in the limited number of objects that can be stored in the HOW LE repository (25 analysis objects and 75 design objects).</p> |
| Purpose     | <p>HOW is the first component design and assembly environment built expressly for developers of distributed business applications. HOW for PowerBuilder allows developers to snap together multitier applications with business-object-based components based on PowerBuilder user objects. Using HOW, developers can custom-build or adapt and extend business objects and technology components for applications that are partitioned along presentation, business logic, and data access boundaries.</p>                                                                                |

HOW's integrated set of tools supports a practical, flexible, and object-oriented approach to business application development. Business object domains, visual storyboards, workflows, and an object repository ensure the development of reusable PowerBuilder business objects and technology components.

HOW LE (and all HOW 1.2 products) includes HOW OpenFrame. This framework supports development of partitioned PowerBuilder 5.0.03 and 6.0 applications, including support for PowerBuilder's distributed capabilities. HOW applications let you take advantage of PFC using PFC ancestors during generation. You can also use HOW with custom framework objects that extend or specialize PFC.

**Usage**

The HOW installation process provides a PowerBar button that allows you to invoke HOW directly from the PowerBuilder development environment.

HOW LE is developed by Riverton Software Corporation ([www.riverton.com](http://www.riverton.com)). The HOW 1.2 product line (including HOW LE) is provided on Windows 95 and Windows NT 4.0. This product line supports PowerBuilder Versions 5.0.03 and 6.0, S-Designer Version 5.1, PowerDesigner 6.0, and LogicWork's ERwin/ERX for PowerBuilder Versions 2.6 and 3.0.

For HOW support, call Sybase's technical support group.

**Documentation**

See the HOW online help, which you can access from the PowerBuilder Help contents listing, or other HOW online documentation, which you can access from HOW's program group.

## **PSChart**

**Description**

PSChart is a full-featured charting component based on the Visual Components First Impression Charting engine. PSChart is included as a bonus to PowerBuilder customers on the Windows platform. It can be installed from the bonus directory on the PowerBuilder CD. The PSChart ActiveX includes a variety of 2D and 3D charting styles, including bar, line, area, step, pie and scatter graphs. A robust programming interface offers numerous properties, methods, and events that allow PowerBuilder developers convenient access to the many available features and integration with the DataWindow as an OLE presentation style.

**Documentation**

See the online Help provided with PSChart.



About this chapter

With PowerBuilder, you can develop various kinds of applications, using a choice of architectures. This chapter describes those choices.

Contents

| <b>Topic</b>                             | <b>Page</b> |
|------------------------------------------|-------------|
| Choosing an architecture                 | 56          |
| Client/server                            | 57          |
| Internet                                 | 62          |
| PowerBuilder automation server component | 68          |

## Choosing an architecture

System architecture requirements typically focus on how an application is to fit into its target computing environment.

### Architecture requirements

Sometimes system architecture requirements are simple, such as when the application is intended to run on one particular operating system and access a local database. Increasingly, however, system architecture requirements are more complex, such as when an application must run on multiple platforms, access one or more server databases, execute code on application servers, and maybe even port to the Internet.

There are many reasons for this increased complexity. Corporations want to leverage their existing investments, while at the same time increasing functionality and positioning the enterprise for the future. They want to save time and money during both application development and deployment.

### Architectures that PowerBuilder supports

PowerBuilder supports many architectures:

| Category           | Architecture                             |
|--------------------|------------------------------------------|
| Client/server      | Single-tier                              |
|                    | Two-tier (traditional client/server)     |
|                    | Multitier (distributed applications)     |
| Internet           | Web.PB                                   |
|                    | PowerBuilder window plug-in              |
|                    | PowerBuilder window ActiveX              |
|                    | DataWindow plug-in                       |
| ActiveX automation | PowerBuilder automation server component |

PowerBuilder allows you to develop and deploy applications in the architecture that's right for your current needs. And if those needs change, you can easily redeploy to a different architecture.

This chapter gives you a general overview of the application architectures you can implement through PowerBuilder. Within each of these architectures, you can use a wide variety of technologies to build an application. These technologies are described in Chapter 3, "The Building Blocks".



## Client/server

### Single-tier

#### Description

A single-tier application runs entirely within a single workstation. All required logic and processing is contained within a single application. A single-tier application can include print and e-mail functionality, but it does not access data through a server database or communicate with a server application.

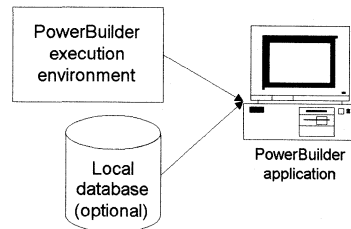
#### Uses

You use a single-tier application when the business problem to be solved does not involve network resources.

PowerBuilder ships with many sample single-tier applications, including:

- ◆ The DWSyntax utility (no database)
- ◆ The Picture Viewer (no database)
- ◆ The PFC Security Administration utility (PFC only)
- ◆ The Application Profiler (no database; PowerBuilder Enterprise only)

#### Illustration



#### Ingredients

A single-tier PowerBuilder application uses windows, window controls, and menus to present the user interface and can use DataWindows to access a local database, such as SQL Anywhere. Its windows can include ActiveX controls for specialized user interface elements.

You can place the application logic within control, window, and menu scripts—or you can use custom class user objects to separate user interface from application interface.

---

#### Planning for the future

If you implement application logic in a custom class user object, a single-tier application will port easily to other architectures.

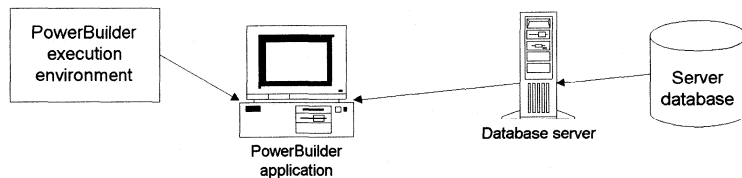
---

|                |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|----------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Implementation | <p>To implement a single-tier application:</p> <ol style="list-style-type: none"><li>1 Use the Window painter to design the user interface.</li><li>2 (Optional) Use the Menu painter to design menus for the application's windows.</li><li>3 (Optional) Use the DataWindow painter to access database data and display it in a suitable presentation style.</li><li>4 Code application processing logic. You can place this code in different locations:<ul style="list-style-type: none"><li>◆ In events for menu items and visual controls</li><li>◆ In window-level events, user events, and window functions</li><li>◆ In custom class user objects</li></ul></li></ol> |
| Documentation  | <p>To learn more about single-tier application development, see <i>Application Techniques</i>.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |

## Two-tier (traditional client/server)

|             |                                                                                                                                                                                                                                                                                                                                                             |
|-------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Description | <p>In a two-tier application, a PowerBuilder program running on a client workstation accesses a database running on the server.</p> <p>One advantage of PowerBuilder is that you can transform a single-tier database application into a two-tier application by changing the application's database connection parameters to access a server database.</p> |
| Uses        | <p>Developers have written thousands of two-tier PowerBuilder applications for both commercial and corporate deployment. These applications include customer service, manufacturing, finance, human resources, and accounting. PowerBuilder is the application development tool of choice for two-tier client/server applications.</p>                      |

### Illustration



**Ingredients**

A two-tier PowerBuilder application uses windows, window controls, and menus to present the user interface and can use DataWindows to access data from a server database, such as Sybase Adaptive Server. Its windows can include ActiveX controls for specialized user interface elements.

You can place the application logic within control, window, and menu scripts—or you can use custom class user objects to separate the user interface from the application interface.

---

**Planning for the future**

If you implement application logic in a custom class user object, a two-tier application will port easily to other architectures.

---

A two-tier application that uses one of Powersoft's native database drivers also requires DBMS-specific client software on the client workstation.

**Implementation**

To implement a two-tier application:

- 1 Use the Window painter to design the user interface.
- 2 (Optional) Use the Menu painter to design menus for the application's windows.
- 3 Use the DataWindow painter to access server database data and display it in a suitable presentation style.
- 4 Code application processing logic. You can place this code in different locations:
  - ◆ In events for menu items and visual controls
  - ◆ In window-level events, user events, and window functions
  - ◆ In custom class user objects

---

**Use PFC**

Consider implementing new applications using PFC (PowerBuilder Foundation Class Library). PFC's service-based architecture and complete selection of precoded objects provide a solid foundation for new application development.

---

**Documentation**

To learn more about building two-tier applications, see *Application Techniques*. To learn more about accessing databases, see *Connecting to Your Database*. To learn more about PFC, see the *PFC User's Guide*.

## Multitier

### Description

In a multitier application, a PowerBuilder program running on a client workstation accesses a PowerBuilder server application for certain application processes, such as validation, business rules, and database access. The server application in turn accesses database servers as necessary.

Multitier PowerBuilder applications use custom class user objects to contain the logic that executes on the application server. Within a custom class user object, you define object functions that the client application calls and that return data to the client application.

### Uses

You use a multitier PowerBuilder application for many purposes:

- ◆ To control access to sensitive information and business rules
- ◆ To simplify client deployment (when the server performs all database access, the client application doesn't need DBMS-specific client software)
- ◆ To insulate the user interface from internal changes to business logic
- ◆ To reduce the workload on client applications

---

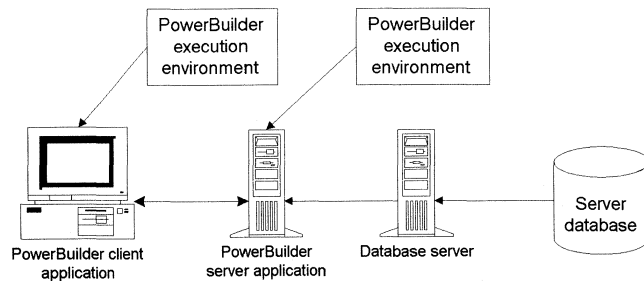
### Other types of multitier PowerBuilder applications

Other types of multitier PowerBuilder applications include:

- ◆ Web.PB
- ◆ OLE automation server (when used with DCOM)

---

### Illustration



### Ingredients

**Client application** Contains windows, menus, DataWindow objects, and proxy user objects. Proxy user objects provide a link to the custom class user objects running on the application server. A PowerBuilder client application also uses a Connection object to access the application server.

**Server application** Contains custom class user objects, which client applications call to perform processing. A PowerBuilder server application also uses a Transport object to listen for client requests, and often has a simple visual component to display server status and statistics.

**Implementation**

**Server application** To implement the server component of a multitier application:

- 1 (Optional) Use the Window and Menu painters to design the application server's user interface.
- 2 Use the DataWindow painter to develop DataWindow objects that access database data for use in client applications.
- 3 Use the User Object painter to create custom class user objects that contain application processing logic.

**Client application** To implement the client component of a multitier application:

- 1 Use the Window painter to design the user interface.
- 2 (Optional) Use the Menu painter to design menus for the application's windows.
- 3 Use the DataWindow painter to create DataWindow objects to contain rows returned by the server application.
- 4 Create a Connection object and establish a connection to the server application.
- 5 Perform application processing by calling functions in custom class user objects on the server application.

**Documentation**

To learn about developing multitier applications, see *Application Techniques*.

# Internet

## Web.PB

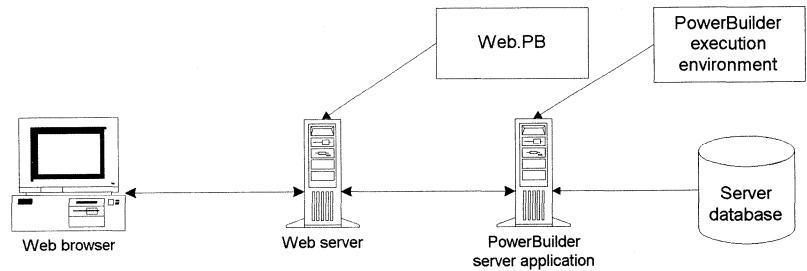
### Description

Web.PB brings the distributed computing capabilities of PowerBuilder to the World Wide Web. By making it possible for Web browsers to call functions in distributed objects, Web.PB enables HTML documents to take advantage of PowerBuilder's advanced capabilities.

### Uses

Use Web.PB to provide a Web-based thin client interface to your application. Using Web.PB, users can access a PowerBuilder application with nothing but a Web browser on the client.

### Illustration



### Ingredients

A Web.PB application contains:

- ◆ Web browser
- ◆ Web server that supports CGI, NSAPI, or ISAPI
- ◆ Custom class user objects
- ◆ Application server
- ◆ Communications driver (WinSock or NamedPipes)

### Implementation

To use Web.PB:

- 1 Create custom class user objects that process data as necessary and return HTML.
- 2 Code a PowerBuilder server application that listens for Web.PB requests. This application must include the custom class user objects coded in step 1.
- 3 Set up the Web server:

- ◆ Install Web.PB's CGI, ISAPI, or NSAPI interface on the Web server
- ◆ Install one of the supported communications drivers (WinSock is included automatically)
- ◆ Code a PBWEB.INI file and make it accessible to the Web server (this file tells the Web.PB interface how to access the server application)
- ◆ Create HTML that calls your user object functions

**Documentation**

FOR INFO To learn more about Web.PB, see *Building Internet Applications with PowerBuilder*.

## PowerBuilder window ActiveX

**Description**

The PowerBuilder window ActiveX lets you display a PowerBuilder child window in an HTML page. You can use JavaScript or VBScript to interact with the child window via the PowerBuilder window ActiveX.

---

**Similar to the PowerBuilder window plug-in**

The PowerBuilder window ActiveX is similar to the PowerBuilder window plug-in. The difference is that the window ActiveX supports programmatic interaction via JavaScript or VBScript.

---

**Uses**

HTML provides a limited user interface. Use the PowerBuilder window ActiveX to provide a rich user interface within an HTML page. This interface can contain all PowerBuilder controls, including the DataWindow.

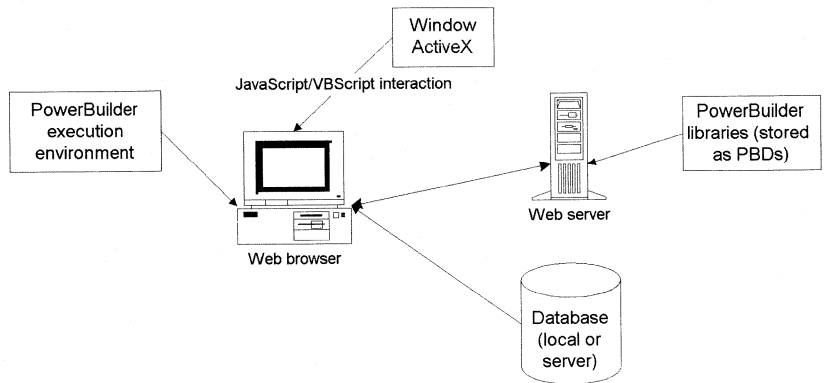
---

**Secure mode**

PowerBuilder also includes a secure version of the PowerBuilder window ActiveX, which disallows all functions that could possibly disrupt the end-user workstation.

---

### Illustration



### Ingredients

A PowerBuilder window ActiveX application contains:

- ◆ A Web browser that supports ActiveX
- ◆ One or more HTML pages to contain the PowerBuilder window ActiveX (these pages can use JavaScript or VBScript to interact with the PowerBuilder window ActiveX)
- ◆ A Web server
- ◆ PowerBuilder windows and other PowerBuilder objects
- ◆ The PowerBuilder window ActiveX
- ◆ The PowerBuilder virtual machine (PBVM60.DLL)
- ◆ (Optional) Database client software

### Implementation

To use the window ActiveX:

- 1 Create, test, and build the PowerBuilder application.
- 2 Save the application in one or more PowerBuilder dynamic libraries (PBDs).
- 3 Create HTML pages with Object elements that point to PowerBuilder child windows. These pages can include JavaScript or VBScript that interacts with the PowerBuilder window ActiveX.
- 4 Configure the Web server by copying the HTML pages, the PowerBuilder window ActiveX, and PowerBuilder dynamic library files to the appropriate directories.
- 5 On all client workstations, install the PowerBuilder window ActiveX, the PowerBuilder deployment DLLs, and (if necessary) database client software.



**Documentation** To learn about the PowerBuilder window ActiveX, see *Building Internet Applications with PowerBuilder*.

## PowerBuilder window plug-in

**Description** The PowerBuilder window plug-in lets you display a PowerBuilder child window in an HTML page.

**Uses** HTML provides a limited user interface. Use the PowerBuilder window plug-in to provide a rich user interface within an HTML page. This interface can contain all PowerBuilder controls, including the DataWindow.

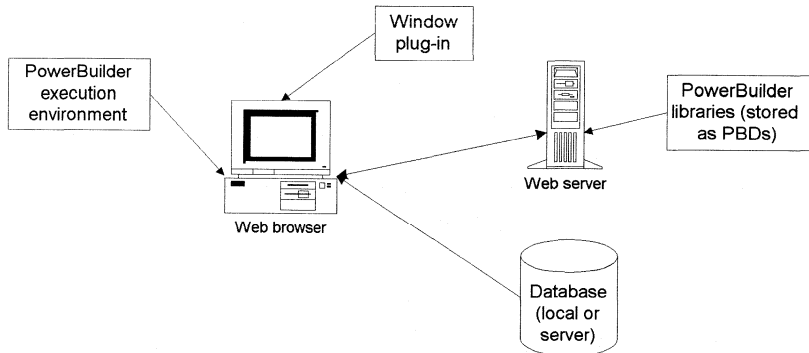
---

### Secure mode

PowerBuilder also includes a secure version of the PowerBuilder window plug-in, which disallows all functions that could possibly disrupt the end-user workstation.

---

### Illustration



|                |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|----------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Ingredients    | <p>A PowerBuilder window plug-in application contains:</p> <ul style="list-style-type: none"><li>◆ A Web browser that supports plug-ins</li><li>◆ One or more HTML pages to contain the window plug-in</li><li>◆ A Web server</li><li>◆ PowerBuilder windows and other PowerBuilder objects</li><li>◆ The PowerBuilder window plug-in module</li><li>◆ The PowerBuilder virtual machine (PBVM60.DLL)</li><li>◆ (Optional) Database client software</li></ul>                                                                                                                                                                                                                                                                                   |
| Implementation | <p>To use the PowerBuilder window plug-in:</p> <ol style="list-style-type: none"><li>1 Create, test, and build the PowerBuilder application.</li><li>2 Save the application in a PowerBuilder dynamic library (PBD).</li><li>3 Create HTML pages with Embed elements that point to PowerBuilder child windows.</li><li>4 Configure the Web server by copying the HTML pages and the PowerBuilder dynamic library files to the appropriate directories, then define the MIME type for PowerBuilder's PBD files.</li><li>5 On all client workstations, install the PowerBuilder window plug-in, the PowerBuilder deployment DLLs, and (if necessary) database client software; then define the MIME type for PowerBuilder's PBD files.</li></ol> |
| Documentation  | <p>To learn about the window plug-in, see <i>Building Internet Applications with PowerBuilder</i>.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |

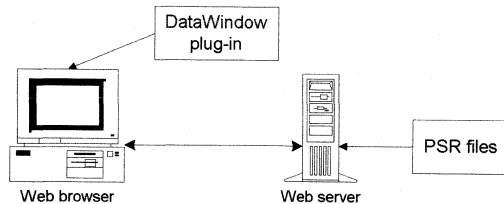
## DataWindow plug-in

|             |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|-------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Description | <p>The DataWindow plug-in lets you display Powersoft report (PSR) files on an HTML page. These reports can use most of the DataWindow presentation styles, including Graph, Crosstab, Freeform, Tabular, Label, and Composite (RichText is not supported).</p> <p>A PSR file contains a report definition (which includes presentation information) as well as the data. Because its data is saved with it, a PSR file does not require a database connection. But the data is static and cannot be refreshed.</p> |
|-------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

**Uses** Many sites already use PSR files to distribute reports. Use the DataWindow plug-in to distribute PSR files through the World Wide Web.

By regularly creating a central source of PSR files from current data (perhaps even nightly), end users can display, print, and save reports using only a Web browser equipped with the DataWindow plug-in.

**Illustration**



**Ingredients**

A DataWindow plug-in application contains:

- ◆ A Web browser that supports plug-ins
- ◆ One or more HTML pages to contain the DataWindow plug-in
- ◆ A Web server
- ◆ PSR files
- ◆ The PowerBuilder DataWindow plug-in

**Implementation**

To use the DataWindow plug-in:

- 1 Create PSR files.
- 2 Create HTML pages with Embed elements that point to PSR files.
- 3 Configure the Web server by copying the HTML pages and the PSR files to the appropriate directories, then define the MIME type for PowerBuilder's PSR files.
- 4 On all client workstations, install the PowerBuilder DataWindow plug-in and define the MIME type for PowerBuilder's PSR files.

**Documentation**

To learn about the DataWindow plug-in, see *Building Internet Applications with PowerBuilder*.

## PowerBuilder automation server component

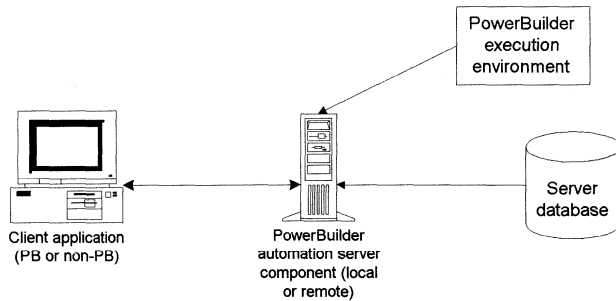
**Description** The PowerBuilder automation server component is a server for programmable (rather than insertable) objects. It allows client components to access functions defined in custom and standard class user objects.

**Uses** Use the PowerBuilder automation server component to allow both PowerBuilder and non-PowerBuilder applications to access functions defined in class user objects. You can also run the server component on a machine other than the client workstation.

Using the PowerBuilder automation server component, you can:

- ◆ Call user object functions from non-PowerBuilder applications, such as Visual Basic
- ◆ Distribute applications without using PowerBuilder's distributed computing capabilities
- ◆ Define user objects as Microsoft Transaction Server components

### Illustration



**Ingredients** Using the PowerBuilder automation server component requires:

- ◆ ActiveX automation
- ◆ Registry entries on the client workstation
- ◆ Standard or custom class user objects
- ◆ The PowerBuilder execution environment on the server

**Implementation** To use the PowerBuilder automation server component:

- 1 Create the standard or custom class user object, including functions to be called by the client.
- 2 Create an execution time library for the user object (either Pcode or machine code).
- 3 Create registry information for the user object and register it on client workstations.
- 4 Write client applications that connect to the user object and call user object functions.

Documentation

To learn more about the PowerBuilder automation server component, see *Application Techniques*.



# The Building Blocks

## About this chapter

PowerBuilder provides a rich selection of technologies that you'll use as the building blocks of your application. This chapter is an overview of them.

## Contents

| <b>Topic</b>                      | <b>Page</b> |
|-----------------------------------|-------------|
| Choosing technologies             | 72          |
| User interface                    | 74          |
| Application programming interface | 86          |
| Language                          | 88          |
| Data access                       | 105         |
| Program access                    | 120         |
| Output                            | 131         |
| Environment                       | 137         |

## Choosing technologies

PowerBuilder provides a rich selection of technologies that you can use in building an application. These include:

- ◆ Native PowerBuilder user interface elements
- ◆ Object-oriented development and programming for both visual and nonvisual objects
- ◆ The PowerBuilder Foundation Class Library (PFC)
- ◆ The advanced database access capabilities provided by the DataWindow
- ◆ Database drivers for ODBC and native access
- ◆ PowerBuilder's distributed computing capabilities
- ◆ Access to external functions
- ◆ OLE, ActiveX, and DDE
- ◆ Access from C++ programs
- ◆ Multiplatform application development
- ◆ Machine code and Pcode deployment
- ◆ Integrated source control
- ◆ International support

### Review this chapter

Review this chapter to learn about the technologies you can use when implementing a PowerBuilder application. The more time you spend planning your application development strategy, the better.

### Plan for the future

When choosing technologies for an application, try to position your application for the future. For example, if you place application logic in custom class user objects, you can use that logic in all supported architectures.



Sample code

If you're the type of person who learns by reading code, you might want to start by looking at the PowerBuilder sample applications and code examples:

◆ *Application Gallery*

This is a collection of working sample applications that have been built in PowerBuilder. You can run these applications to perform useful tasks. You can also examine their contents in PowerBuilder to get ideas for your own projects and to copy objects or code.

◆ *Code Examples*

This is a collection of windows that demonstrate PowerBuilder coding techniques and show you how PowerBuilder can solve common coding problems.

# User interface

## Windows

Windows are the major building blocks of an application's user interface. They provide features you can use to let people view information, manipulate information, and initiate actions.

**The role of a window** You can design the user interface of an application to involve just one window. But most of the time you'll involve several different windows, with each one playing a particular role to help the user get a larger job done. The role of any individual window is usually to present a particular kind of information and let users interact with that information in certain ways.

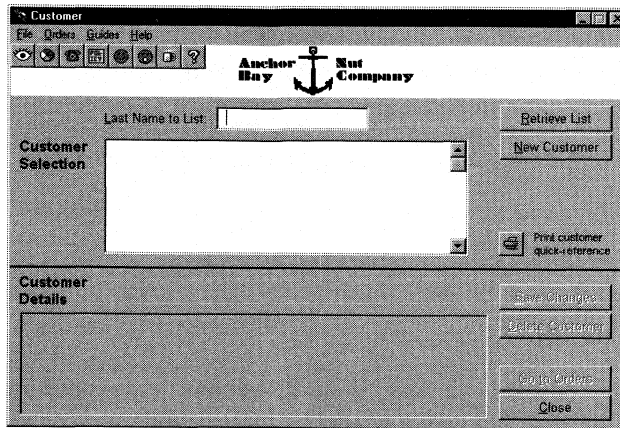
**Types of windows** PowerBuilder provides several different types of windows that you can use in your application. Each type has some unique characteristics that make it good for fulfilling one or more specific presentation or interaction needs:

- ◆ Main windows
- ◆ Response windows and message boxes
- ◆ Popup windows and child windows
- ◆ MDI frames

### Main windows

|             |                                                                                                                                                                    |
|-------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Description | <b>Main windows</b> are where you'll usually have users perform the major activities of the application. You can think of them as home bases for those activities. |
| Uses        | Use main windows for PowerBuilder client/server applications.                                                                                                      |

## Illustration



## Ingredients

Main windows display one or more controls, which users interact with to drive the application. Main windows often include menus, which users use to request application processing.

When used within an MDI frame (explained on page 77), main windows are called **sheets**.

## Implementation

Window painter

## Documentation

To learn about main windows, see the *PowerBuilder User's Guide*.

## Response windows and message boxes

## Description

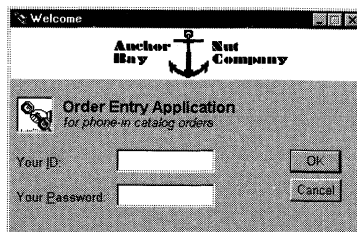
**Response windows** and **message boxes** are good for situations where you want to force users to consider some information and/or choose some action before they can do anything else in the application.

## Uses

Use response windows when an application requires user input before continuing.

Use message boxes to convey information needed before continuing.

## Illustration



|                |                                                                                                                                                                                                      |
|----------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Ingredients    | Response windows and message boxes display text. They also contain buttons the user clicks to direct processing and close the window. Response windows can also contain other PowerBuilder controls. |
| Implementation | Window painter (response windows)<br>PowerScript MessageBox function (message boxes)                                                                                                                 |
| Documentation  | To learn about response windows and message boxes, see the <i>PowerBuilder User's Guide</i> .                                                                                                        |

## Popup and child windows

**Description**                      **Popup windows** and **child windows** are supporting windows that display on top of a parent window. Child windows always display within the parent window; popup windows can display outside the parent window. Both types of windows close when the parent window closes.

---

### **Child windows**

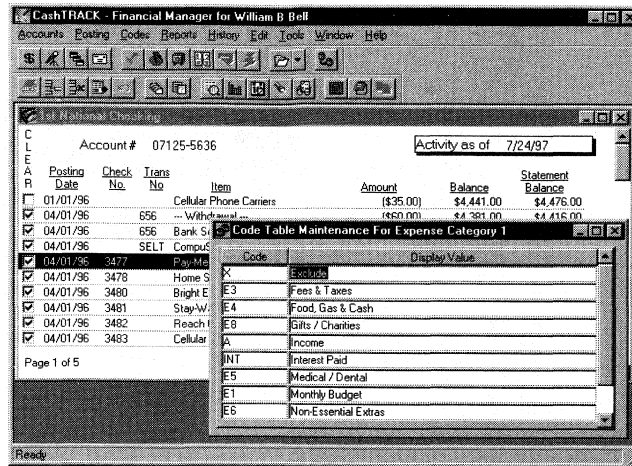
The windows displayed in HTML pages by the PowerBuilder window plug-in and the PowerBuilder window ActiveX must be child windows.

---

**Uses**                                Popup windows and child windows are handy for displaying additional pieces of information (or providing additional services) that support users' activities as they work in a particular main window.

Child windows are the foundation of a PowerBuilder window plug-in or PowerBuilder windows ActiveX application.

## Illustration



## Ingredients

Popup and child windows display one or more controls, with which users interact in conjunction with the parent window.

## Implementation

Window painter

## Documentation

To learn about popup and child windows, see the *PowerBuilder User's Guide*. To learn about the PowerBuilder window plug-in and window ActiveX, see *Building Internet Applications with PowerBuilder*.

## MDI frames

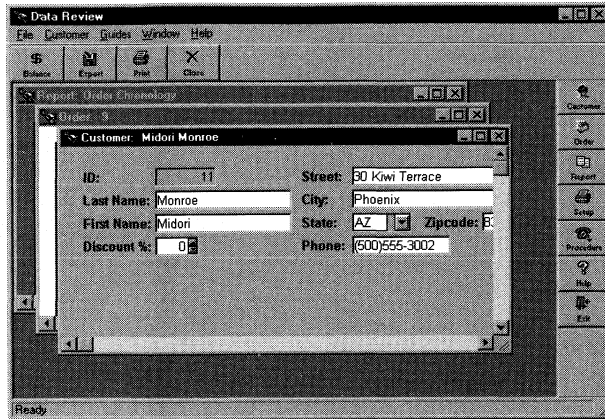
## Description

**MDI frames** are containers for multiple document windows. When placed inside one of these frames, main windows act as sheets that users can easily position.

## Uses

MDI frames are useful in many applications where users need a convenient and organized way to work with multiple main windows. Additionally, MDI frames allow you to open multiple instances of the same sheet, such as multiple customer orders.

Illustration



Ingredients

MDI frames contain zero or more sheet windows. An MDI frame window must use a menu and can optionally display a toolbar.

Implementation

Window painter

Documentation

To learn about MDI frames, see the *PowerBuilder User's Guide*

## Controls

PowerBuilder provides a wide range of controls you can place in a window to hold the information users need, and to implement the interactions users perform.

You can use virtually any combination of controls in a particular window, depending on the activities that the window is intended to support. And you can arrange them in whatever way best suits your needs.

**FOR INFO** For more information on the various kinds of controls, see the *PowerBuilder User's Guide*.

## Standard controls

Description

PowerBuilder supplies a wide range of window controls that you use to create an application's user interface.

Uses

Your application uses window controls to display and edit values and to allow users to request a particular action.

## Kinds

There are controls for:

- ◆ **Displaying and/or manipulating values** These controls include:

- StaticText
- SingleLineEdit
- MultiLineEdit
- RichTextEdit
- EditMask

- ◆ **Making choices** These controls include:

- ListBox
- PictureListBox
- DropDownListBox
- DropDownPictureListBox
- CheckBox
- RadioButton

- ◆ **Initiating actions** These controls include:

- CommandButton
- PictureButton

- ◆ **Presenting your data** PowerBuilder provides a special kind of control called a:

- DataWindow

that you use when you want a window to display formatted data. By the way, when you design a DataWindow, you'll specify that it is to present the data it accesses in a style that's just like one or more of the standard PowerBuilder controls. As a result, the look and use of what's inside your DataWindow control will be consistent with the look and use of the other controls around it in the window.

- ◆ **Presenting sophisticated lists** These controls include:

- ListView
- TreeView

- ◆ **Showing information graphically** These controls include:

- Graph
- HScrollBar
- VScrollBar

- ◆ **Dressing up a window** These controls include:

- Line
- Oval
- Rectangle

RoundRectangle  
Picture

◆ **Organizing other controls** These controls include:

GroupBox  
Tab

Implementation Window painter  
User Object painter

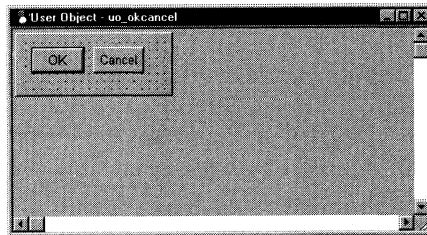
Documentation To learn about standard PowerBuilder controls, see the *PowerBuilder User's Guide*

## Controls you define

Description In addition to standard controls, you can also define your own controls. You base these controls on one or more of the standard PowerBuilder controls and store them as application components called **visual user objects**. Then you can include these custom-made controls in any windows you want.

Uses Use visual user objects to provide reusable window controls. For example, you might define a custom visual user object that contains a set of precoded command buttons, such as OK and Cancel.

Illustration



Ingredients A visual user object that contains precoded object properties, event scripts, and object functions.

Kinds **Standard visual user objects** Contain a single control  
**Custom visual user objects** Contain one or more controls

Implementation User object painter  
Window painter

Documentation To learn about visual user objects, see the *PowerBuilder User's Guide*.



## Controls from external sources

|                |                                                                                                                                                                                                                                                                                                                                                                                                            |
|----------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Description    | <p>You can use controls that were created outside PowerBuilder (in DLLs) by defining them as user objects too. Then you can include these external controls in any windows you want.</p> <p>PowerBuilder also enables you to put ActiveX controls (also called OLE custom controls) in your windows by using its own versatile <b>OLE control</b>. You'll learn more about that later in this chapter.</p> |
| Uses           | Use controls from external sources to provide specific functionality not provided by PowerBuilder controls.                                                                                                                                                                                                                                                                                                |
| Ingredients    | A DLL that contains controls.                                                                                                                                                                                                                                                                                                                                                                              |
| Implementation | User Object painter<br>Window painter                                                                                                                                                                                                                                                                                                                                                                      |
| Documentation  | To learn about external controls, see the <i>PowerBuilder User's Guide</i> .                                                                                                                                                                                                                                                                                                                               |

## Menus

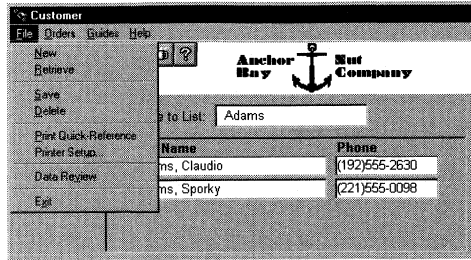
Another way to let the user initiate actions in a window is to use menus. A menu lists commands (**menu items**) that are currently available so that the user can select one. PowerBuilder provides several methods for providing access to menu item commands:

- ◆ On the window's menu bar
- ◆ On the MDI frame's toolbar
- ◆ Inside the window as a popup

## Menu bars

|             |                                                                                                                                 |
|-------------|---------------------------------------------------------------------------------------------------------------------------------|
| Description | In many of the windows you design you'll want to display a menu of one or more items across the top in the menu bar.            |
| Uses        | The menu bar enables users to move through menu items and pull down submenus (dropdown menus) that you've defined for each one. |

Illustration



Implementation

Menu painter

Documentation

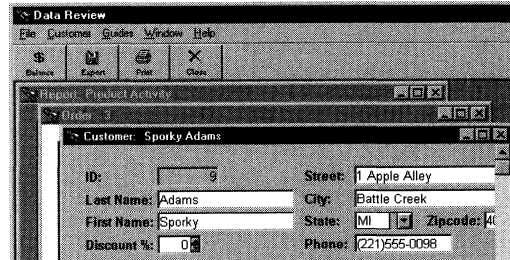
To learn about menu bars, see the *PowerBuilder User's Guide*.

## Toolbars

Description

If the window is an MDI frame, you can optionally define a **toolbar** to accompany the menu. The toolbar displays buttons corresponding to one or more menu items, giving the user an alternative way to select those items.

Illustration



Implementation

Menu painter

Documentation

To learn about toolbars, see the *PowerBuilder User's Guide*.

## Popup menus

Description

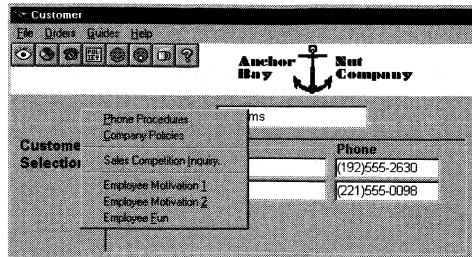
Sometimes you may want to let users initiate certain actions by popping up a menu within the window. A popup menu lists its items vertically, enabling users to move up or down to select the item they want.

Uses

Popup menus can be handy for giving users quick access to a subset of the most common actions they perform in the window or to just those actions that apply to the current control.

Not all types of windows support menus (for example, response windows don't), but in those that do (such as main windows) you can use a menu bar, popup menus, both, or neither. It all depends on your design goals for the window.

#### Illustration



#### Implementation

Menu painter  
PowerScript painter  
Window painter

#### Documentation

To learn about popup menus, see the *PowerBuilder User's Guide*.

## User interface style

After you choose an application architecture, you need to design the user interface.

**Choosing a user-interface style** To design a successful user interface for an application—one that enables people to interactively and smoothly perform all of their required activities—you must do more than just draw up the individual windows that support those activities. You've also got to figure out the connections among your windows, including:

- ◆ How users are to navigate through the application from one window to another
- ◆ Which windows should be available to use at a given moment
- ◆ Whether a particular window will depend in some way on one or more other windows

The best way to start addressing these issues is to decide on the user interface style that your application is to follow.

**Your choices** There are three user interface styles you'll usually consider using. Each one will give your application a particular look and feel:

- ◆ **SDI** Single Document Interface
- ◆ **MDI** Multiple Document Interface
- ◆ **Combination** The application uses an SDI approach to implement some of its activities and an MDI approach to implement others

Determining which style best suits your application depends on a lot of factors, among them: the nature of the application’s data, the complexity of the application’s processing, the preferences of the application’s users, and the conventions (if any) you’re obligated to follow.

## Single Document Interface (SDI)

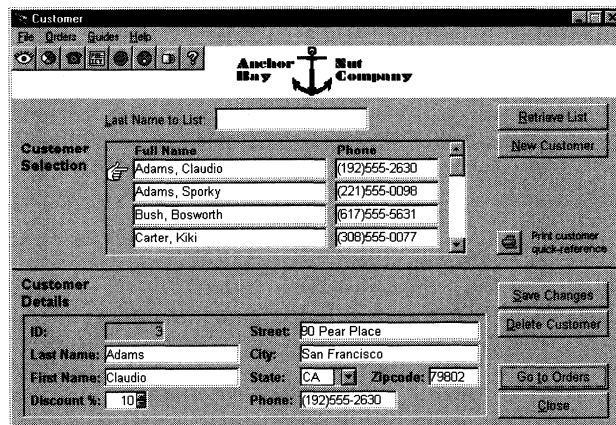
Description

In this style, users typically work with one main window at a time to perform an activity (although that window may display various popup or child windows to do supporting chores as a user works). When users want to perform a different kind of activity, they go to a different main window to do it.

Uses

SDI may be appropriate if your application is very simple, especially if it deals with only one kind of data and the user needs to perform only one operation at a time.

Illustration



Ingredients

Main windows  
Menus (optional)  
Supporting windows (response, child, popup)

Implementation

Window painter  
Menu painter

Documentation To learn about SDI applications, see the *PowerBuilder User's Guide*.

## Multiple Document Interface (MDI)

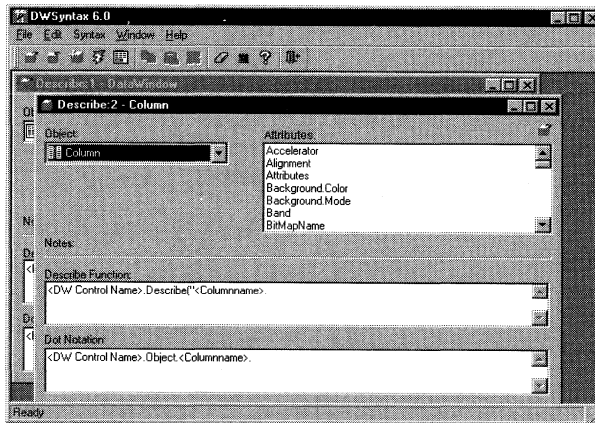
Description In this style, users work within a frame that lets them perform activities on multiple sheets of information. MDI tends to be most useful in applications where users require the ability to do several different things at a time.

Uses MDI is the most popular choice because of the flexibility it gives the user and all of the built-in services it provides. You should probably think about using it by default.

### Consider this

In the world of graphical applications, many of the more prevalent commercial products (including PowerBuilder) use an MDI interface.

### Illustration



Ingredients

- Frame window
- Frame menu and toolbar
- Sheet windows (main windows)
- Sheet menus and toolbars
- Supporting windows (response, child, popup)

Implementation

- Window painter
- Menu painter

Documentation To learn about MDI applications, see the *PowerBuilder User's Guide*.

# Application programming interface

**Description**

PowerBuilder is an extremely flexible development environment. There are many ways to code an application, and you can place your application logic in various places. In general you want to create reusable code, and you can achieve varying degrees of reuse depending on where you place application logic:

| Code placement                                                            | Example                                                                                                                                                    | Reusability       |
|---------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------|
| Within the events that occur when the user requests the associated action | The Clicked event of a Save command button or menu item saves changes in all DataWindows on a window                                                       | None              |
| Within user events for the window in which the user requests actions      | The Clicked event of a Save command button or menu item calls the window's ue_Save event, which saves changes for all DataWindows on the window            | Within the window |
| Within window functions for the window in which the user requests actions | The Clicked event of a Save command button or menu item calls the window's wf_Save function, which saves changes for all DataWindows on the window         | Within the window |
| Within custom class user objects                                          | The Clicked event of a Save command button or menu item might call the user object's of_Save function, which saves changes for all DataWindows on a window | Application-wide  |

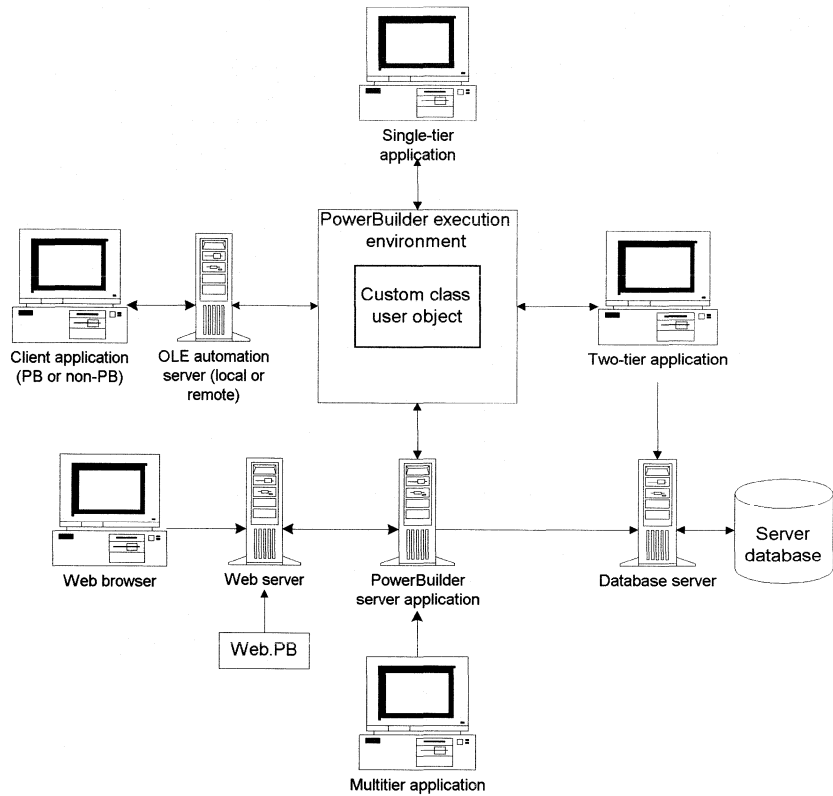
Because custom class user objects provide the most reusability, you should use them to code application-wide functionality. When you create these objects, you define a set of user object functions (called an **application programming interface**, or API) that applications call to perform application processing.

Custom class user objects are the foundation upon which you build PowerBuilder server applications. By using custom class user object to contain your application logic, it can be used in all architectures.

**Uses**

By coding application processing logic in custom class user objects, you separate the user interface from the application programming interface. This separation allows you to create applications using any of the architectures described in Chapter 2, "What You Can Build".

Illustration



Ingredients

Custom class user object  
 PowerBuilder application (single-tier, two-tier, or multitier)  
 PowerBuilder server application  
 Web.PB  
 C++

Implementation

User Object painter  
 PowerScript painter  
 Project painter

Documentation

FOR INFO To learn about using custom class user objects to define an API, see *Application Techniques*. To learn about events and functions, see "Scripting" next.

# Language

## Scripting

You write scripts using PowerScript, the PowerBuilder language. Scripts consist of PowerScript commands, function calls, and statements that perform processing. You code scripts for events and functions.

Events and functions are similar in many ways:

- ◆ They can be triggered or posted
- ◆ They can have arguments and return values
- ◆ You can create user events and user-defined functions
- ◆ They can both call ancestor events and functions

But there are a few differences:

- ◆ Calling a nonexistent event fails silently; calling a nonexistent function creates an execution time error
- ◆ Functions can be overloaded; events cannot
- ◆ Events can be extended; functions cannot

## Events

### Description

PowerBuilder provides an approach to flow-of-control in an application that puts users in charge. PowerBuilder applications are **event driven**—they wait to see what actions a user takes in a window to determine what processing to perform.

**How it works** Whenever the user does something involving one of the application's user interface components (such as a window, control, or menu item), that action triggers a particular **event**. For example, each of the following actions triggers a different event:

| Doing this                                                 | Triggers                                        |
|------------------------------------------------------------|-------------------------------------------------|
| Clicking on a particular CommandButton control in a window | The Clicked event of that CommandButton control |
| Clicking on a particular menu item in a window's menu      | The Clicked event of that menu item             |



| Doing this                                                             | Triggers                                          |
|------------------------------------------------------------------------|---------------------------------------------------|
| Modifying the value in a particular SingleLineEdit control of a window | The Modified event of that SingleLineEdit control |
| Closing a particular window                                            | The Close event of that window                    |

**What an event does** When an event is triggered, the application executes a corresponding **script**, which contains any processing logic you've written for that event.

Each kind of user-interface component has its own set of several events that can happen to it. For instance:

| This component           | Has                                                                               |
|--------------------------|-----------------------------------------------------------------------------------|
| A CommandButton control  | About a dozen different events, including: Clicked, GetFocus, and LoseFocus       |
| A menu item              | Just a couple of events: Clicked and Selected                                     |
| A SingleLineEdit control | About a dozen different events, including: Modified, GetFocus, and LoseFocus      |
| A window                 | More than 25 different events, including: Open, Close, Resize, Timer, and Clicked |

Events can have arguments (that your scripts can use to determine how to process) and return values (that your scripts can set to control how processing continues).

### Defining events in user objects

You can define events within a user object. This enables you to encapsulate the event processing within the object to which the processing applies.

### Uses

Your job as a designer is to figure out all the events of interest that might occur in your application and provide appropriate processing logic for each event (in its script). To do that, you've got to know more about the various kinds of events there are.

Although this quickly adds up to a lot of events, you don't have to worry about handling every one. In many cases, you'll need to write scripts for just one or two of the events of a particular component (and sometimes you won't need any for that component).

**If you need to take control** Letting users drive the flow of processing is appropriate most of the time, but on occasion you'll want the application to temporarily take control. In these situations, you can write code in the script of one event that *programmatically* causes another event to occur. When doing this, you can either:

- ◆ *Trigger the event* so that its script executes right away, *or*
- ◆ *Post the event to a queue* so that its script execution is deferred (until after the scripts of any earlier events have executed)

You can also define your own events for any particular component and then programmatically trigger or post them to execute their scripts. These are called **user events**, and they can be useful for such things as:

- ◆ Extending the processing of other event scripts by serving as subroutines
- ◆ Responding to certain lower-level messages (from your operating environment) that PowerBuilder doesn't provide as standard events

**Kinds**

Hardware events  
Additional window events you can define  
User events that use a predefined PowerBuilder ID  
User events with arguments and/or return values

**Implementation**

PowerScript painter

**Documentation**

To learn about events, see the *PowerBuilder User's Guide*. To learn about triggering and posting events, see the *PowerScript Reference*. For more on defining user events, see the *PowerBuilder User's Guide*.

**Functions**

**Description**

You don't have to put all the code you write in event scripts. In many cases you'll find that it's better to separate out certain chunks of code to be independent of any particular event.

For these cases, PowerBuilder gives you the ability to create your own functions, known as **user-defined functions**. When you create a user-defined function, you'll specify:

- ◆ The arguments it requires when you execute it (if any)
- ◆ The PowerScript code it is to execute
- ◆ The value it is to return (if any)

You can then call that user-defined function from event scripts or from other user-defined functions.

---

**Defining functions in user objects**

You can define functions within a user object. This enables you to encapsulate the event and function processing within the object to which the processing applies.

---

**Uses** Use functions to create independent processing units. This enables you to centralize processing logic that's required in multiple situations, organize your application's code so that it's easier to maintain and extend, or simply break an especially long body of code into more manageable subroutines.

---

**Choosing between user events and user-defined functions**

User events and user-defined functions are similar in many respects. For instance, both can have arguments and return values. Both can be triggered or posted. As a result, you can use either technique just as successfully in many coding situations.

But user events and user-defined functions do have some differences that may cause you to choose one instead of the other in certain cases.

---

**Ingredients** Arguments  
PowerScript code  
Return values

**Kinds** PowerScript functions  
External functions  
Global functions  
User object functions

**Implementation** PowerScript painter  
Function painter

**Documentation** To learn about the difference between user events and user-defined functions, see *Application Techniques*. To learn about PowerScript function syntax, see the *PowerScript Reference*.

## PowerScript

### Description

A script is a body of procedural code that you write in the **PowerScript** language to express the processing logic to perform. Most scripts are relatively short (tens of lines long, not hundreds), since they just need to express the processing for particular events and functions and not for the whole application.

**Ingredients of a script** PowerScript is a high-level language that provides several different syntactic ingredients you can use to write the code you need. These ingredients include:

- ◆ **Variable declarations** PowerBuilder supports many data types, as well as arrays and structures of variables. It also offers several levels of scoping that you can choose from for each variable you declare.
- ◆ **PowerScript statements** These statements mostly provide flow-of-control mechanisms (such as branching and looping) that you can use to steer the processing in a particular script.
- ◆ **Function calls** PowerScript supplies a multitude of built-in functions you can call. You'll use these built-in functions to handle much of the processing in your scripts.

When calling functions you can either trigger or post them, just as you do when calling events. Triggering executes a function right away. Posting defers execution of a function (until earlier functions and events have executed).

PowerBuilder also lets you create your own functions and then call them in your scripts.

- ◆ **Embedded SQL statements** If you need to perform some table processing and you've decided to use SQL (instead of DataWindows) to do it, you can embed the appropriate SQL statements in a script. PowerBuilder lets you embed standard as well as dynamic SQL statements, and it supports DBMS-specific clauses and reserved words.
- ◆ **Comments** PowerBuilder makes it easy to insert comments in your code. You can use comments to document how a script works or to temporarily block out some code that you don't want to execute.

### Uses

You use scripts to perform your application's processing.

## Illustration

```

Script for Clicked event of the
cb_new CommandButton
Script for
RButtonDown event
of the w_customer
window
Script for Clicked
event of the

```

```

////////////////////////////////////
//
// To begin, check whether user did any typing in dw_detail /
// DataWindow contr
// will be lost if
//
// We've coded a wi
// wf_warndataloss)
//
////////////////////////////////////
// If the user clicks the right mouse button while the
// pointer is on the window itself (but not on a control),
// then display a popup menu. The popup menu we want to
// display for the Customer window consists of the m_guide
// portion of the m_custmenu menu that we already display
// in the window's menu bar.
//
////////////////////////////////////
IF wf_warndataloss(
 RETURN
END IF

// First, determine the current position (x and y coordinates)
// of the pointer.

integer li_x, li_y

//
// Next, some init
//
li_x = PointerX ()
li_y = PointerY ()

//
// Now, display the m_guide menu as a popup menu at the current
// pointer location.
//
// Reset the
// disable i
m_custmenu.m_guide.PopupMenu (li_x, li_y)

dw_list.Reset()
dw_list.enabled = false
slc_lname.text = ""

```

```

//
// First, loop through each row of the dw_orddetail
// control and delete each one from the primary buf
//

integer li_counter
long ll_detailrows

// Determine how many item rows are currentl
// dw_orddetail.

ll_detailrows = dw_orddetail.RowCount ()

// Now loop through them.

FOR li_counter = 1 TO ll_detailrows

// Delete the current item row.

```

```
// This script uses embedded SQL to find the highest Cust_Id
// currently stored in the database. It then adds 1 to that
// number to provide the Id for a new customer.

integer li_fetched_id = 0

SELECT Max("customer"."cust_id")
 INTO :li_fetched_id
 FROM "customer"
 USING sqlca;

IF sqlca.sqlcode >= 0 THEN

 IF IsNull(li_fetched_id) THEN
 li_fetched_id = 0
 END IF

 li_fetched_id ++

ELSE

 li_fetched_id = -1

END IF

 .
 .
 .
```

Ingredients

Events  
Functions

Implementation

PowerScript painter

Documentation

To learn about coding event and function scripts, see *Application Techniques*.  
To learn about event and function syntax, see the *PowerScript Reference*.

## Objects

### Objects you build

Description

The components you create in PowerBuilder (such as windows, menus, and user objects) are not merely pieces of a particular application, but self-contained **objects**.

That means each one *encapsulates* the particular characteristics and behaviors (properties, events, and functions) that are appropriate to it. It also means that they support other standard object-oriented capabilities, including *inheritance* and *polymorphism*.

|                |                                                                                                                                                                                                                                                                  |
|----------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Uses           | You use objects for practically everything in an application. By leveraging PowerBuilder's object-oriented capabilities, you can benefit in several important ways—in particular by making your work more modular, reusable, extensible, flexible, and powerful. |
| Implementation | Window painter<br>Menu painter<br>User Object painter                                                                                                                                                                                                            |
| Documentation  | To learn about PowerBuilder objects, see the <i>PowerBuilder User's Guide</i> .                                                                                                                                                                                  |

## System objects

|             |                                                                                                                                                                                                                                                                                                                                                                                                                   |
|-------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Description | <p>In addition to the objects you create in the painters, PowerBuilder provides a number of <b>system objects</b>. These objects aren't maintained in libraries. Instead, they're managed by PowerBuilder itself (although you can work with them in your scripts).</p> <p>Each system object contains properties, events, and functions that your application can use to perform the appropriate processing.</p> |
|-------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

---

### Internally, everything is an object

The objects you create in the painters inherit from PowerBuilder system objects. For example, when you create a new window, PowerBuilder creates an object internally that inherits from the system object *Window*.

---

|      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Uses | <p>You use these objects throughout your application:</p> <ul style="list-style-type: none"> <li>◆ Controls placed on a window are descendants of system objects (for example, <code>CommandButton</code> or <code>ListBox</code>)</li> <li>◆ The application object contains properties of type <code>Transaction</code>, <code>DynamicDescriptionArea</code>, <code>DynamicStagingArea</code>, <code>Error</code>, and <code>Message</code>, each of which is a system object</li> <li>◆ You can define and create variables of a system object type. This provides access to the object's properties, events, and functions</li> </ul> <p>You can even use the User Object painter to create customized descendants of certain system objects:</p> |
|------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

- ◆ **Standard visual user objects** Descend from any of the visual controls, such as CommandButton, DataWindow, ListBox, and TreeView
- ◆ **Standard class user objects** Descend from certain system objects, such as Transaction, DataStore, Error, and Message

You use these user objects to extend the object's basic capabilities by defining additional properties, events, and functions.

|                |                                                                                                                                           |
|----------------|-------------------------------------------------------------------------------------------------------------------------------------------|
| Kinds          | There are many types of system objects. For a complete list of properties, events, and functions for each system object, use the Browser. |
| Implementation | Application painter<br>PowerScript painter<br>The Browser                                                                                 |
| Documentation  | To learn about standard class user objects, see the <i>PowerBuilder User's Guide</i> .                                                    |

## Libraries

**Description** A **library** is the container in which you collect a particular set of objects for use in your applications. It's where an object goes when you save it in a painter and where a painter gets an object you ask to open. PowerBuilder maintains each library you use in its own file in your operating system. PowerBuilder libraries are files with an extension of **PBL**.

To help you organize these libraries and control the objects they contain, PowerBuilder includes a tool called the **Library painter**.

**Uses** A PowerBuilder application contains one or more libraries in its library list. This list names all the libraries that contain objects used in an application. How you organize your libraries is up to you. For instance, you can:

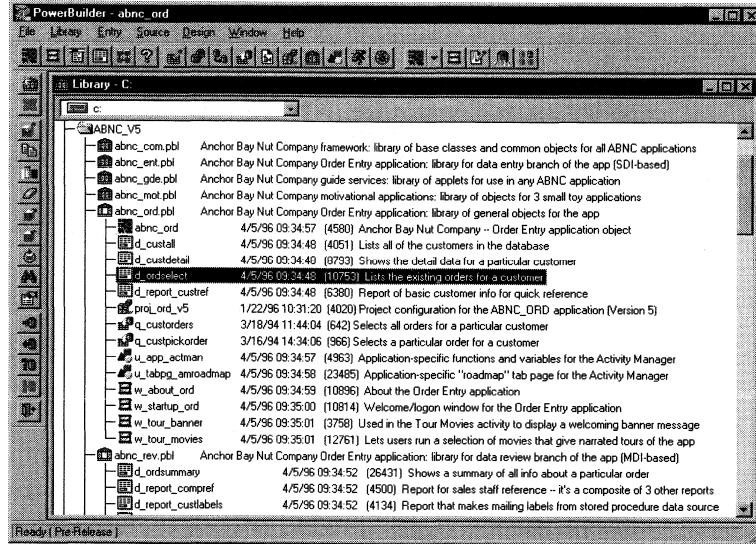
- ◆ Create one or more libraries for a particular application
- ◆ Use the same library in more than one application
- ◆ Store libraries in different locations, including your computer or a server
- ◆ Store objects related to a particular task in the same library



### Cross-platform use of libraries

Libraries and the objects in them are *platform independent*. That means you can move or share your PBL files across platforms (such as Microsoft Windows, Apple Macintosh, or UNIX) during an application development project.

#### Illustration



#### Ingredients

Whenever you save an object in a library, PowerBuilder actually stores *two* forms of that object:

| This form of an object | Is                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Source                 | A syntactic representation of the object, including any script code it contains. You can think of this as the object's definition                                                                                                                                                                                                                                                                                                                                                                       |
| Compiled               | <p>A pseudocode (<b>Pcode</b>) representation of the object that's ready for execution. Every time you save an object in a painter, PowerBuilder automatically compiles that object into Pcode for you</p> <p>Pcode is an interpreted language that makes it quick and easy for you to test-run objects from the PowerBuilder development environment. It is also one of the compiler options available to you (along with machine code) when generating the executable version of your application</p> |

|                |                                                                                              |
|----------------|----------------------------------------------------------------------------------------------|
| Implementation | Library painter                                                                              |
| Documentation  | To learn about libraries and the Library painter, see the <i>PowerBuilder User's Guide</i> . |

## Object-oriented programming

### Classes

**Description** In object-oriented programming, you create reusable **classes** to perform application processing. These classes include **properties** and **methods** that define the classes' behavior. To perform application processing, you create **instances** of these classes. PowerBuilder implements these concepts as follows:

- ◆ **Classes** PowerBuilder objects, such as windows, menus, window controls, and user objects
- ◆ **Properties** Object variables and instance variables
- ◆ **Methods** Events and functions

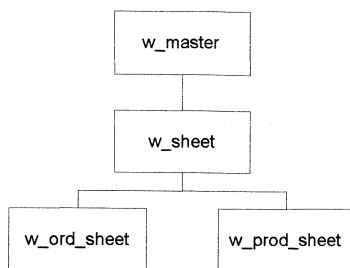
**Documentation** To learn more about object-oriented programming in PowerBuilder, see *Application Techniques*.

### Inheritance

**Description** Using inheritance, objects can be derived from existing objects, with access to their visual component, data, and code. Inheritance saves coding time, maximizes code reuse, reduces maintenance, and enhances consistency. A descendent object is also called a **subclass**.

**Uses** You can use inheritance for windows, menus, and user objects. Implementing inheritance maximizes code reuse and helps to create more reliable and maintainable applications.

## Illustration



## Implementation

Window painter  
Menu painter  
User Object painter

## Documentation

To learn more about inheritance, see the *PowerBuilder User's Guide*.

## Encapsulation

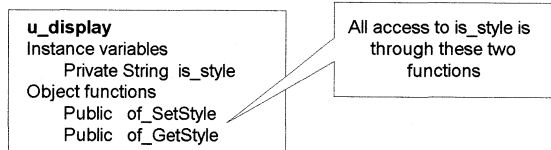
## Description

An encapsulated object contains its own data and code, allowing outside access as appropriate. This principle is also called *information hiding*. PowerBuilder enables and supports encapsulation by giving you features that can enforce it, such as access and scope. But PowerBuilder itself does not require or automatically enforce encapsulation.

## Uses

Use encapsulation to restrict direct access to an object's instance variables. In place of direct access, provide object functions to set and read the instance variables. These object functions can then perform all necessary error checking and validation before modifying or accessing the instance variables.

## Illustration



## Implementation

Window painter  
Menu painter  
User Object painter  
PowerScript painter  
Function painter

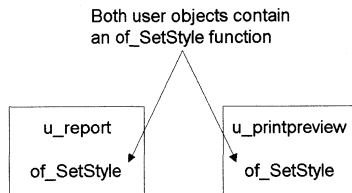
## Documentation

To learn more about encapsulation, see *Application Techniques*.

## Polymorphism

- Description** With polymorphism, functions with the same name behave differently, depending on the referenced object. Polymorphism enables you to provide a consistent interface throughout and within all objects.
- Uses** By implementing polymorphism, your application uses a consistent API. A function can have the same basic name in every object for which it is enabled.

**Illustration**



- Implementation** PowerScript painter  
Function painter
- Documentation** To learn more about polymorphism, see *Application Techniques*.

## Visual and nonvisual classes

### Visual classes

- Description** These objects are the windows, controls, and menus that you build for the user interface of a graphical application.
- Uses** Use visual objects to create your application's user interface.
- Implementation** Window painter  
Menu painter  
User Object painter
- Documentation** To learn about class user objects, see *Application Techniques*.

### Nonvisual classes

- Description** PowerBuilder also lets you develop and use nonvisual objects. Effective use of nonvisual objects is essential in professional-quality applications.

---

|                |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|----------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Uses           | <p>To fully benefit from PowerBuilder's object-oriented capabilities, consider implementing class user objects (also known as <i>nonvisual user objects</i>):</p> <ul style="list-style-type: none"><li>◆ <b>Standard class user objects</b> Inherit their definitions from built-in PowerBuilder system objects, such as Transaction, Message, or Error. Creating customized standard class user objects allows you to provide powerful extensions to built-in PowerBuilder system objects.</li><li>◆ <b>Custom class user objects</b> Inherit their definitions from the PowerBuilder NonVisualObject class. Custom class user objects encapsulate data and code. This type of class user object allows you to define an object class from scratch. To create server applications and make the most of PowerBuilder's object-oriented capabilities, you must use custom class user objects. Typical uses include:<ul style="list-style-type: none"><li>◆ <b>Business rules</b> The custom class user object contains functions and variables that implement business rules. You can either create one object for all business rules or create multiple objects for related groups of business rules.</li><li>◆ <b>Distributed computing</b> The custom class user object contains the functions that run on the server.</li><li>◆ <b>Service object</b> The custom class user object contains functions and variables that are useful either in a specific context (such as a DataWindow) or globally (such as a collection of string-handling functions).</li><li>◆ <b>Global variable container</b> The custom class user object contains variables and functions for use across your application. You encapsulate these variables as appropriate for your application, allowing access directly or through object functions.</li></ul></li></ul> |
| Implementation | User Object painter<br>Function painter                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| Documentation  | To learn about class user objects, see <i>Application Techniques</i> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |

## Class libraries

|             |                                                                                                                                                                                                   |
|-------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Description | To get the most out of PowerBuilder, you've got to take full advantage of its object-oriented features. And the best way to do that is to base each project you begin on a <b>class library</b> . |
|-------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

There's some variation in how people use the term class library (and a related term, **application framework**), but at minimum it refers to a collection of **base classes** (ancestor objects) from which you can inherit most of the application-specific objects you need to develop for your project.

The idea is that you define these base classes with the characteristics and behaviors (properties, events, and functions) you commonly want in the objects you construct. That way, when you create an object inherited from one of them, the new object automatically gets all the characteristics and behaviors of that base class.

#### Uses

Using a class library is much more than just a convenient way to copy features from one object to another. That's because inherited objects don't duplicate the features of their base classes, but *refer* to them. This makes it easy for you to apply global changes, since modifications you make in a base class are automatically picked up by its descendants. And when you execute your application, it can perform more efficiently because those inherited features are loaded just once (via the base class).

In other words, using a class library can help make your applications better in a variety of ways (more consistent and modular, easier to fix and extend, faster and leaner). It can also make them much quicker to build, since you'll be able to reuse so much of your work on each one.

#### Ingredients

If you don't already have an appropriate class library available to use on your projects, you can either:

- ◆ Develop one yourself, *or*
- ◆ Acquire one

**Developing a class library** This involves using the PowerBuilder painters to create one or more libraries in which you then store the ancestor objects you want to serve as base classes.

As you develop your ancestor objects, try to include only those features that will be needed in *most* of the descendent objects you'll inherit from them. This will help ensure efficiency. And avoid creating ancestor objects that don't really serve some purpose (because you don't want to introduce superfluous levels of inheritance into your applications).

If you're part of a team, you may have a particular team member in the role of object manager. This object manager is typically responsible for overseeing the development and maintenance of a class library for your organization.

**Acquiring a class library** There are a variety of class libraries and application frameworks available commercially for use with PowerBuilder. If you don't have the time or resources to develop one yourself, you might consider one of these. This includes the PowerBuilder Foundation Class Library produced by Powersoft (described next).

## PowerBuilder Foundation Class Library (PFC)

|               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|---------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Description   | <p>The PowerBuilder Foundation Class Library (PFC) is an extensive set of base classes that you can extend to create class libraries for:</p> <ul style="list-style-type: none"><li>◆ Corporate use</li><li>◆ Departmental use</li><li>◆ Individual applications</li></ul> <p>PFC includes:</p> <ul style="list-style-type: none"><li>◆ Base classes for all window types</li><li>◆ Bases classes for all standard visual user objects</li><li>◆ Base classes for all standard class visual user objects</li><li>◆ Custom class user objects that implement DataWindow, windows, application, and utility services</li><li>◆ Utilities, including a DataWindow properties dialog</li></ul> <p>PFC is included in the Advanced Developer Toolkit.</p> |
| Uses          | <p>Use PFC as the basis for your own class library.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| Documentation | <p>To learn more about PFC, see the <i>PFC User's Guide</i>.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |

## Special-purpose libraries

|             |                                                                                                                                                                                                                                                                                                               |
|-------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Description | <p>Sometimes you may need to enhance your application by including one or more special-purpose libraries. These are libraries whose objects can be used in an application to enable it to perform some specialized kind of processing, such as interaction with a particular software product or service.</p> |
|-------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

**Uses**

If you need to develop PowerBuilder applications that work with Lotus Notes, you might consider using the PowerBuilder Library for Lotus Notes, which is produced by Powersoft. Another special-purpose class library from Powersoft is the Web.PB class library, which contains functions that help you create HTML. Other vendors also offer special-purpose libraries that you can use in your PowerBuilder applications.



# Data access

## Database connections

### Description

To access the data it needs, your application can connect to one or more databases (or other data sources) of various kinds. It can also control when to connect to a database, when to disconnect from it, and how that connection should operate.

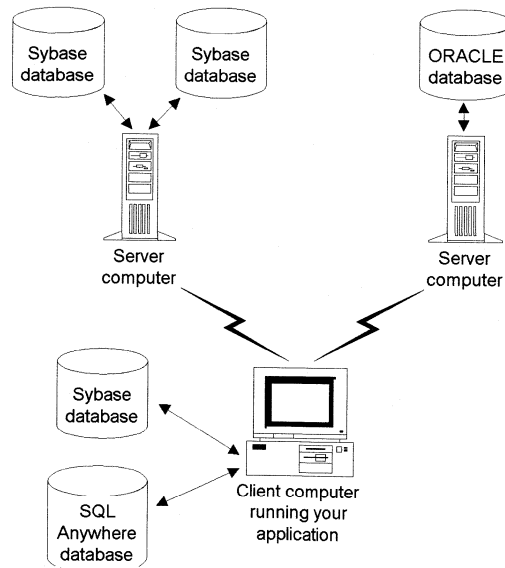
### Uses

The requirements of your application probably call for access to at least one table. In fact, it's usually the case that applications need to access several.

What makes this access tricky in a client/server environment is that these tables are likely to be in different places and different formats. Specifically:

- ◆ The tables may be stored *in one or more databases*
- ◆ Those databases may be located *in a variety of locations*: on the client computer, on one or more server computers, or on a mix
- ◆ Those databases may be implemented *under a variety of DBMSs*

### Illustration



### Documentation

To learn more about database connections, see *Connecting to Your Database*.

## Database interfaces

### Description

PowerBuilder provides an approach to data access that enables you to successfully handle this potential database diversity in the applications you build. It does this by separating the DBMS-specific aspects of data access from an application to make it as independent as possible. This means you can focus on the logical use of a table in your application instead of how that table is implemented in one particular database or another.

**Introducing the database interfaces** PowerBuilder handles DBMS specifics in a separate software layer that you install on the client computer along with your application. This layer consists of various database interfaces, each of which knows how to talk to a particular kind of DBMS and how to take advantage of the unique features of that DBMS. When your application requests any kind of access to a database, it relies on the appropriate database interface (depending on the DBMS for that database) to carry out the operation.

### Uses

**What this approach does for you** The major benefit of this layered approach to data access is that it helps insulate your application from the complicated and potentially dynamic logistics of the typical client/server environment. As a result, the data access you design into your application can be:

- ◆ **Flexible** You can make your application independent of a database's location or DBMS. That way, if the database needs to be moved (from client to server, from server to client, or from server to server) or migrated to a different DBMS, the application itself doesn't have to be disrupted.

Adapting your application to such changes can often be just a matter of pointing it to the database's new location and database interface.

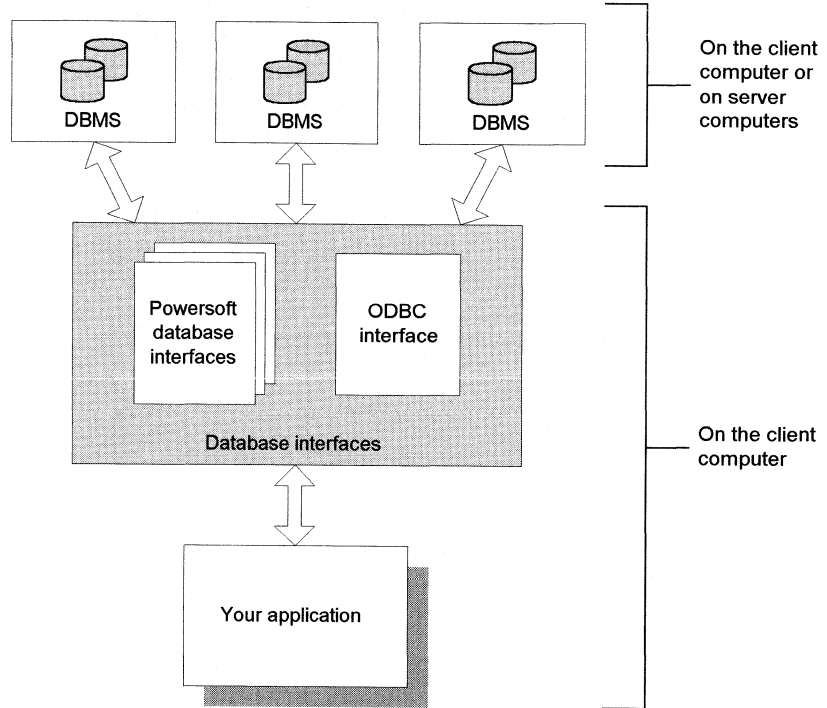
- ◆ **Consistent** You can work with all of the tables in your application in the same way—using the same table-processing features—regardless of the DBMSs that are involved. (You'll learn more about these table-processing features of PowerBuilder in just a moment.) That means you don't have to design DBMS-specific processing routines.

Even in cases where you want to take advantage of capabilities unique to certain DBMSs (such as stored procedures, outer joins, referential integrity checking), you'll still use consistent PowerBuilder techniques to do so. Of course, if you're planning to migrate to a different DBMS later, you should make sure it also supports those capabilities (or you'll need to modify the application).

**Setting up the databases to access** As you'd expect, before your application can access particular databases and their tables, they must exist. In lots of cases the necessary database design, definition, and population work may have already been done (by you or someone else such as a DBA) through DBMS facilities or other design tools.

But if this database setup work is not yet done or if some modifications are needed, you can use certain painters in PowerBuilder to help do the job.

Illustration



Ingredients

DBMS-specific Powersoft database interface  
 ODBC interface

Kinds

Notice that the preceding figure shows two kinds of database interfaces that PowerBuilder provides for your application to use:

- ◆ **The ODBC interface** You'll design your application to use this interface if you want to access one or more ODBC-compliant databases.

ODBC is the Open Database Connectivity API developed by Microsoft to give applications standardized access to diverse data sources (which are usually databases, but can also be other kinds of files such as spreadsheets or text files). The ODBC interface that's included with PowerBuilder was developed by Powersoft to let your applications talk to ODBC, which in turn talks to the actual data sources.

- ◆ **The Powersoft database interfaces** Sometimes you won't want to access a particular database via ODBC, or won't be able to because that database is not ODBC-compliant. For these cases, Powersoft offers a variety of native interfaces, each of which knows how to talk to a specific DBMS (such as Sybase SQL Server or Oracle).

So if you want to access a SQL Server database, for example, you'll design your application to use the Powersoft SQL Server interface.

You can design your application to use any combination of these database interfaces.

Implementation      PowerScript painter  
                             Database painter

Documentation        To learn about database interfaces, see *Connecting to Your Database*.

## Transactions

Description            PowerBuilder lets you access the database either within the scope of a transaction or outside a transaction.

Uses                    In either case, PowerBuilder provides a **Transaction object** that you use to handle the communication between your application and the database. The Transaction object makes it easy for you to control the nature of your database connection (such as whether to work in transaction mode) and to monitor the status of the connection.

When you do use transactions, you can manage their scope by performing commits and rollbacks as needed.

Implementation      PowerScript painter

Documentation        To learn about transactions and the Transaction object, see *Application Techniques*.

## Embedded SQL

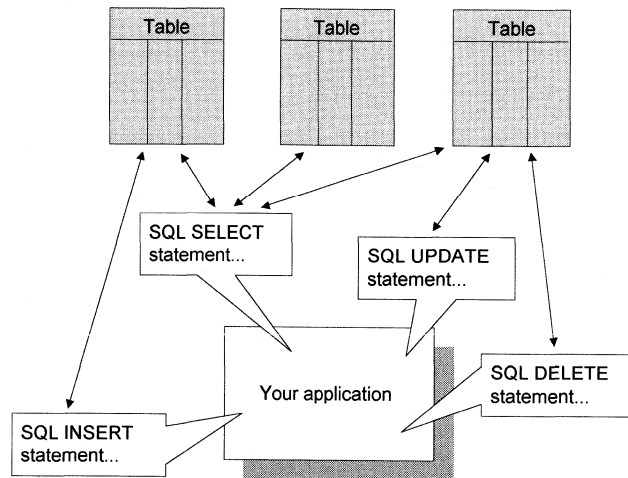
### Description

You can embed **SQL statements** in your application scripts to manipulate the rows in the database. PowerBuilder supports both standard and dynamic SQL.

### Uses

PowerBuilder supports all of the usual features of this industry-standard language, along with DBMS-specific syntax and some powerful extensions of its own. In general, you should think about using embedded SQL in places where your design calls for row manipulation without the need for display.

### Illustration



### Implementation

PowerScript painter  
Database Administration painter  
Select painter  
Query painter

### Documentation

To learn about using embedded SQL in PowerBuilder scripts, see the *PowerBuilder User's Guide*.

## DataWindow objects

### Description

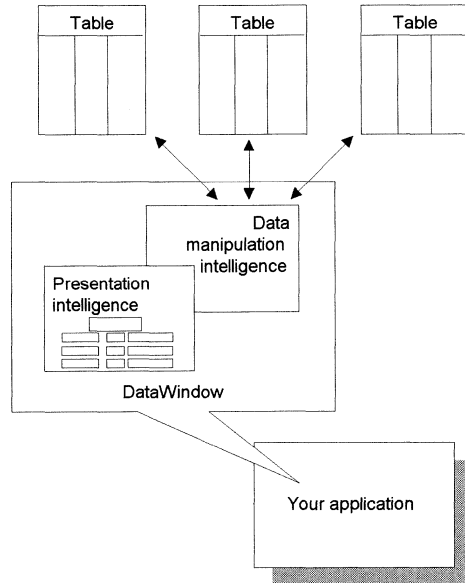
DataWindows are a central feature of PowerBuilder that you'll want to use for most of the table processing your application design requires. That's because DataWindow objects contain both the intelligence to do robust row manipulation (including creation, retrieval, updating, and deletion) and the presentation (user-interface) abilities to let people see and work with those rows:

### Uses

You can use DataWindows for just about all your application display needs. They work well for data display, data input, onscreen and printed reports, and for displaying nondatabase data.

Most of the time you'll want to use DataWindows (instead of embedded SQL) to perform your application's data access tasks.

### Illustration



### Ingredients

You use the PowerBuilder DataWindow painter to build **DataWindow objects**. A DataWindow object consists of a data source (which controls the data that is accessed) and a presentation style (which controls display). Additionally, a DataWindow object provides many advanced features, such as formatting, validation, computed fields, filtering, sorting, and so on.

---

### DataWindow controls and DataStores

To use the DataWindow object in an application, you place it in either a **DataWindow control** (in a window) or a variable of type **DataStore**. The DataWindow control and DataStore system object provide properties, events, and functions that you use to control a DataWindow object.

---

|                |                                                                                                                                                                              |
|----------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Implementation | DataWindow painter                                                                                                                                                           |
| Documentation  | To learn about defining DataWindow objects, see the <i>PowerBuilder User's Guide</i> . To learn about programming in DataWindow objects, see <i>Application Techniques</i> . |

### Data sources

|                |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|----------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Description    | <p>A data source controls the data displayed in a DataWindow object. This can be:</p> <ul style="list-style-type: none"> <li>◆ <b>A SQL SELECT statement</b> The DataWindow object generates the SQL needed to do the appropriate data access. You can fine-tune the SQL yourself if you have special requirements. All standard SQL capabilities are supported, as well as DBMS-specific syntax.</li> <li>◆ <b>A PowerBuilder Query object</b> The DataWindow object uses a PowerBuilder Query object for the SQL SELECT statement.</li> <li>◆ <b>A database stored procedure</b> The DataWindow object uses a DBMS-specific stored procedure to access data.</li> <li>◆ <b>An external source</b> You define the columns to display and populate the DataWindow object programmatically at execution time using an external data source, such as a file.</li> </ul> |
| Uses           | You can define the DataWindow object just for retrieval or for update (insert, update, and delete) too.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| Implementation | DataWindow painter                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| Documentation  | To learn about DataWindow data sources, see the <i>PowerBuilder User's Guide</i> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |

### Presentation styles

|             |                                                                                                                                                                |
|-------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Description | <p>A DataWindow object can present the accessed data in a wide variety of styles:</p> <p style="margin-left: 40px;">Tabular                      Composite</p> |
|-------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------|

|          |          |
|----------|----------|
| Freeform | Graph    |
| Grid     | Crosstab |
| Label    | OLE 2.0  |
| N-up     | RichText |
| Group    |          |

When generating the DataWindow object in your selected presentation style, PowerBuilder uses extended attributes from the Powersoft repository to determine various display characteristics of the columns in the DataWindow object.

Uses

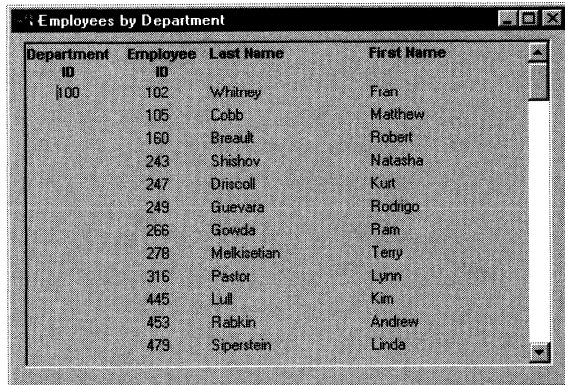
DataWindow object presentation styles have two main uses:

- ◆ **Online data entry and display** The DataWindow's advanced display and validation support allow you to create an intuitive, easy-to-use interface
- ◆ **Online and printed reports** You can also use DataWindows to present many types of online and printed reports

Illustration

Here are a few examples of DataWindow presentation styles:

**Tabular**



The screenshot shows a window titled "Employees by Department" containing a table with the following data:

| Department ID | Employee ID | Last Name   | First Name |
|---------------|-------------|-------------|------------|
| 100           | 102         | Whitney     | Fran       |
|               | 105         | Cobb        | Matthew    |
|               | 160         | Breault     | Robert     |
|               | 243         | Shishov     | Natasha    |
|               | 247         | Driscoll    | Kurt       |
|               | 249         | Guevara     | Rodrigo    |
|               | 266         | Gowda       | Fam        |
|               | 278         | Melkisetian | Terry      |
|               | 316         | Pastor      | Lynn       |
|               | 445         | Lull        | Kim        |
|               | 453         | Rabkin      | Andrew     |
|               | 479         | Siperstein  | Linda      |



Grouped



Product Reference

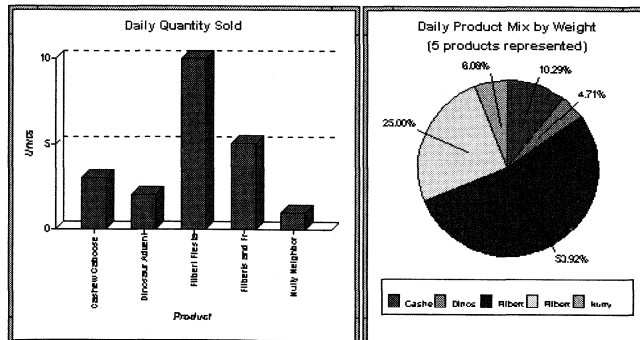
| Category          | Name                           | Price   |
|-------------------|--------------------------------|---------|
| <i>Assortment</i> |                                |         |
|                   | Filberts and Friends           | \$19.95 |
|                   | Nuts of the World              | \$25.95 |
|                   | Nuts-To-You Holiday Assortment | \$29.95 |
|                   | Nutty Neighbor Welcome Basket  | \$23.95 |
| <i>Children's</i> |                                |         |
|                   | Cashew Caboose Kid's Pack      | \$13.95 |
|                   | Dinosaur Adventure Mix         | \$9.95  |
| <i>General</i>    |                                |         |
|                   | Almonds Ahoy!                  | \$15.95 |
|                   | Brazilnut Bonanza              | \$17.95 |
|                   | Carnival of Cashews            | \$21.95 |
|                   | Filbert Fiesta                 | \$18.95 |
|                   | Hooray for Hazelnuts           | \$14.95 |
|                   | Macadamia Mountain             | \$21.95 |
|                   | Peanuts Peanuts Peanuts!       | \$15.95 |
|                   | Pecans on Parade               | \$15.95 |

Graph

Orders By Date



For December 2, 1995





## Crosstab

| Sum of Units Ordered (by State) | Product Category |            |           |             |             |
|---------------------------------|------------------|------------|-----------|-------------|-------------|
| State                           | Assortment       | Children's | General   | Test-Market | Grand Total |
| AZ                              | 0                | 0          | 1         | 0           | 1           |
| CA                              | 7                | 9          | 17        | 10          | 43          |
| GA                              | 23               | 0          | 12        | 1           | 36          |
| HI                              | 5                | 0          | 10        | 0           | 15          |
| IL                              | 0                | 1          | 4         | 0           | 5           |
| LA                              | 3                | 0          | 2         | 0           | 5           |
| MA                              | 3                | 0          | 1         | 0           | 4           |
| MI                              | 0                | 0          | 10        | 0           | 10          |
| MT                              | 6                | 0          | 0         | 0           | 6           |
| OH                              | 9                | 2          | 6         | 0           | 17          |
| TX                              | 0                | 6          | 4         | 0           | 10          |
| UT                              | 1                | 0          | 4         | 0           | 5           |
| VT                              | 5                | 0          | 4         | 2           | 11          |
| <b>Grand Total:</b>             | <b>62</b>        | <b>18</b>  | <b>75</b> | <b>13</b>   | <b>168</b>  |
| <b>Average:</b>                 | <b>5</b>         | <b>1</b>   | <b>6</b>  | <b>1</b>    | <b>13</b>   |

## Label

|                                                                                                                                                 |                                                                                                                                                 |
|-------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------|
|  Bosworth Bush<br>2 Raspberry Row<br>Boston MA 02108           |  Svetlana Jackson<br>501 Lemon Lane<br>Salt Lake City UT 60330 |
|  Homer Washington<br>5 Raisin Road<br>Stowe VT 03384          |  Lana Lincoln<br>2 Grape Way<br>Buzzard Gulch MT 64689        |
|  Suzie Lincoln<br>93 Strawberry Street<br>Sebastian FL 20988 |  May-Lin Jefferson<br>19 Plum Drive<br>Seattle WA 72891      |
|  Kiki Carter<br>43 Orange Street<br>Kansas City MO 28800     |  Claudio Adams<br>90 Pear Place<br>San Francisco CA 79802    |

Implementation

DataWindow painter

Documentation

To learn about presentation styles, see the *PowerBuilder User's Guide*. To learn about online and printed reports, see "Printed reports" on page 131. To learn about extended attributes, see "Detailed features" next.

## Detailed features

**Description**

DataWindows support many other useful features, including:

| <b>Category</b> | <b>Features</b>                                                                                                                                                                                                                                                                        |
|-----------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Cosmetics       | Column formatting (such as: fonts, size, position, edit styles)<br>Display formats<br>Validation rules<br>Newspaper-style column snaking                                                                                                                                               |
| Display         | Display of additional objects (computed fields, text, shapes, and so on)<br>Headers, footers, and summaries<br>Navigation through the data (scrolling, tabbing, paging, and so on)<br>Filtering, sorting, and grouping of rows<br>Prompting for selection criteria<br>Query by example |
| Performance     | Retrieval options (retrieving as needed, retrieving to disk)<br>Save data with object<br>Data sharing with other controls (DataWindow controls, DataStores, or RichTextEdit controls)                                                                                                  |
| Data entry      | Data entry and manipulation (by the user or by the application)<br>Error and status checking                                                                                                                                                                                           |
| Internet        | HTMLTable property contains retrieved data in HTML Table syntax<br>Cascading style sheet defines formatting for HTMLTable<br>GenerateHTMLForm saves rows in HTML Form syntax<br>Saving as HTML allows you to save rows in an HTML table                                                |
| Other           | Dynamic DataWindow object modification at execution time<br>Saving to files (including HTML, Powersoft report, and tab-delimited)<br>Printing                                                                                                                                          |

**Uses**

This abundance of features makes the DataWindow the object of choice when developing database retrieval and display applications. You have design time and execution time control over retrieval, display, and update.

**Implementation**

DataWindow painter  
PowerScript painter

**Documentation**

To learn more about DataWindow object features, see the *PowerBuilder User's Guide*.

## DataWindow controls

|                |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|----------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Description    | A DataWindow control is a visual control (available in the Window painter) that you place on a window. One of the control's properties is DataObject, which specifies the DataWindow object to display in the control. A DataWindow control includes both events and PowerScript functions. For example, the Retrieve function retrieves rows, as specified in the DataWindow object's data source.                                                                                                                                       |
| Uses           | <p>Use a DataWindow control any time you want to display a DataWindow object in a window.</p> <p>Alternatively, you can create a standard visual user object based on the DataWindow control. Code event scripts and define user events and user-defined functions as necessary. Then place the user object in the window instead of a simple DataWindow control. This enables you to encapsulate common processing in one place.</p> <p>At execution time, you can dynamically switch the DataWindow object in a DataWindow control.</p> |
| Illustration   | <p>The diagram shows two main components: 'DataWindow painter' and 'Window painter'. The 'DataWindow painter' is a vertical stack of three boxes, each labeled 'DataWindow object'. The 'Window painter' is a larger box containing a smaller box labeled 'DataWindow control' and the word 'Window' below it. An arrow originates from the top 'DataWindow object' box in the 'DataWindow painter' and points to the 'DataWindow control' box in the 'Window painter'.</p>                                                               |
| Implementation | <p>Window painter</p> <p>User Object painter</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| Documentation  | To learn more about DataWindow controls, see the <i>PowerBuilder User's Guide</i> .                                                                                                                                                                                                                                                                                                                                                                                                                                                       |

## DataStores

### Description

A DataStore is a standard class user object that you can use to contain the data retrieved for a DataWindow object. One of the DataStore's properties is DataObject, which specifies the DataWindow object used to retrieve rows contained in the DataStore. A DataStore includes both events and PowerScript functions. For example, the Retrieve function retrieves rows, as specified in the associated DataWindow object's data source.

### Uses

Use a DataStore any time you need to retrieve data but do not need to display it. Define a variable of type DataStore, create it, assign the DataObject property (the name of the DataWindow object), set the Transaction object, and retrieve rows.

Alternatively, you can create a standard class user object based on DataStore. Code event scripts and define user events and user-defined functions as necessary. Then use the user object as the variable's data type (instead of DataStore). This enables you to encapsulate common processing in one place.

At execution time, you can dynamically switch the DataWindow object in a DataStore.

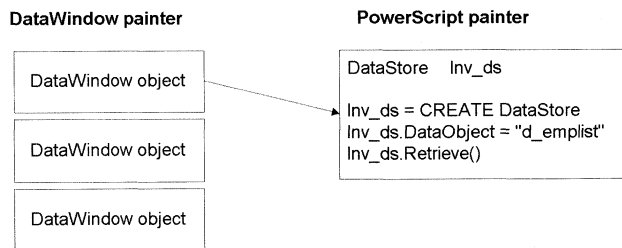
---

### Distributed computing

DataStores are particularly useful in distributed server applications.

---

### Illustration



### Implementation

PowerScript painter

### Documentation

To learn about DataStores, see *Application Techniques*.

## Text and binary files

|                |                                                                                                                                                                                                                                                                                                                                                                                                                                |
|----------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Description    | Although most of the data output from a typical application is directed at databases, you may also need to write data to other kinds of files. For example, users might want the application to save certain data in spreadsheet files that they can use later in another program.                                                                                                                                             |
| Uses           | PowerBuilder provides built-in functions that you can code in scripts to write (and in some cases read) many different kinds of files from your application, including: <ul style="list-style-type: none"><li>◆ Text files</li><li>◆ Microsoft Excel spreadsheets</li><li>◆ Lotus 1-2-3 spreadsheets</li><li>◆ HTML files</li><li>◆ dBASE files</li></ul> You can also access text and Excel database data using ODBC drivers. |
| Implementation | PowerScript painter                                                                                                                                                                                                                                                                                                                                                                                                            |
| Documentation  | To learn about reading and writing different kinds of files, see the <i>PowerBuilder User's Guide</i> .                                                                                                                                                                                                                                                                                                                        |

## Clipboard

|                |                                                                                                                                           |
|----------------|-------------------------------------------------------------------------------------------------------------------------------------------|
| Description    | Your application may need to exchange data with other applications on the user's system. One way to do this is with the system clipboard. |
| Uses           | Use the clipboard to cut, copy, and paste information.                                                                                    |
| Implementation | PowerScript painter                                                                                                                       |
| Documentation  | To learn about clipboard functions, see the <i>PowerScript Reference</i> .                                                                |

## Program access

### Executable programs

Description

In some situations you may want your application to start up another executable application. This might be another PowerBuilder application, a commercial application (such as a word processor or spreadsheet tool), a batch file, a shell, or a shell script.

For these situations, PowerBuilder provides a Run function you can code in any script to run that executable application.

---

#### On Macintosh

If your application is going to be deployed on the Macintosh, you may want it to execute one or more AppleScript scripts to perform some of the processing it requires. For these situations, PowerBuilder provides:

- ◆ The ability to edit and test AppleScript scripts in the PowerBuilder file editor as you develop your application
- ◆ The DoScript function, which you can code in any of the application's scripts to execute the appropriate AppleScript scripts at execution time

Uses

Use the Run function to start other, independently executing programs.

Implementation

PowerScript painter

Documentation

To learn about the Run function, see the *PowerScript Reference*.

### Help systems

Description

Users typically expect to be able to access online Help modules when running an application to get information about what that application does and how to use it. You should determine what these Help modules need to contain as well as how they should be organized and formatted.

Once your end-user Help modules are written, you can display them in an application with the ShowHelp PowerScript function.

PowerBuilder supports the Microsoft Windows Help model.



|                |                                                                                                                                                                                                                                                                                                                                                                                                                      |
|----------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Uses           | Use the ShowHelp PowerScript function to access the Help files for your application.                                                                                                                                                                                                                                                                                                                                 |
| Ingredients    | <p>PowerBuilder supports online Help display for all platforms. However, on Macintosh and UNIX, you must compile the Help files with vendor-specific tools:</p> <ul style="list-style-type: none"><li>◆ <b>On Macintosh</b> QuickHelp from Altura Software, Inc.</li><li>◆ <b>On UNIX</b> HyperHelp from Bristol Technology</li></ul> <p>Contact the vendors for information on acquiring and using these tools.</p> |
| Implementation | PowerScript painter<br>Help authoring tool<br>Help compiler                                                                                                                                                                                                                                                                                                                                                          |
| Documentation  | To learn more about online Help, see <i>Application Techniques</i> .                                                                                                                                                                                                                                                                                                                                                 |

## E-mail systems

|                |                                                                                                                                                                                                                                                                                                                                                                |
|----------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Description    | One of the requirements of your application may be that you make it mail-enabled. Basically, this involves giving your application the ability to send e-mail messages to people and/or receive e-mail messages from them through an electronic mail system.                                                                                                   |
| Uses           | <p>PowerBuilder supports access to any e-mail system that uses the mail standard called MAPI (messaging application program interface). To build this access into your application, you write some appropriate scripts using a few special mail functions that PowerBuilder provides.</p> <p>MAPI is currently supported in the Windows environments only.</p> |
| Implementation | PowerScript painter                                                                                                                                                                                                                                                                                                                                            |
| Documentation  | To learn more about MAPI, see <i>Application Techniques</i> .                                                                                                                                                                                                                                                                                                  |

## Component object model

|               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|---------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Description   | <p>The component object model (COM) provides a framework for communication between software components. It allows programs to share both data and program functionality. PowerBuilder provides containers that call upon COM-based server components to display and manipulate COM objects.</p> <p>Before you choose COM as your solution, make sure it is supported by the program you want to talk to and by your execution platform (not all platforms support COM).</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| Uses          | <p>You can use COM-based technologies to:</p> <ul style="list-style-type: none"><li>◆ Embed or link to data (<b>objects</b>) from other programs inside the windows of your application. That way you can enable a user to point to one of those objects and automatically invoke the capabilities of its corresponding program to edit it.<br/><br/>PowerBuilder provides an <b>OLE control</b> that you can use to do this.</li><li>◆ Connect your DataWindows to other programs in a few different ways, including: using the <b>OLE presentation style</b> to define a DataWindow, inserting an OLE object within a DataWindow, or placing an <b>OLE column</b> in a DataWindow.</li><li>◆ Enable your application to talk to another program under the covers—through a technique called <b>automation</b>—to invoke commands or move data back and forth.<br/><br/>Implementing automation (formerly called OLE automation) involves writing some appropriate scripts that use OLE features of the PowerScript language. These features enable you to develop automation clients as well as servers.</li><li>◆ Include <b>ActiveX controls</b> (formerly called OLE custom controls or OCX controls) in your application.<br/><br/>You can place ActiveX controls in your windows by using the OLE control that PowerBuilder provides. You can also insert them in your DataWindows.</li></ul> |
| Ingredients   | <p>PowerBuilder OLE control<br/>PowerBuilder OLEObject and OLEControl objects<br/>Microsoft OLE 2.0<br/>OLE server application<br/>ActiveX controls<br/>PowerBuilder automation server components and other automation servers</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| Documentation | <p>To learn about OLE in PowerBuilder, see <i>Application Techniques</i>.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |

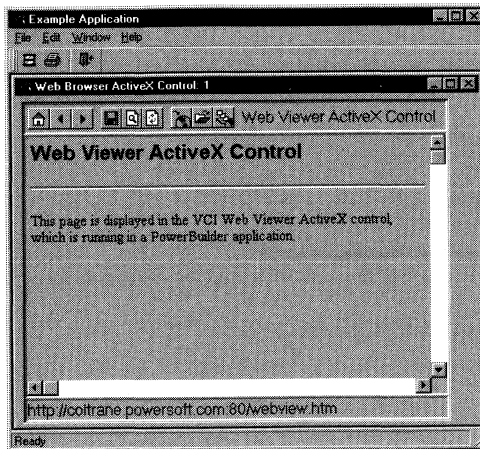
## Compound documents

|                |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|----------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Description    | <p>OLE allows you to create compound documents with components from several server applications (such as Word, Excel, and Visio). The user can activate the control and edit the object using functionality provided by the server application. You can also use automation to programmatically activate the object and send commands to the server. PowerBuilder supports both in-place and offsite activation.</p> <p>PowerBuilder allows you to link or embed OLE objects into a window:</p> <ul style="list-style-type: none"><li>◆ When you <i>embed</i> an object, your application stores its data</li><li>◆ When you <i>link</i> an object, your application contains a reference to the data</li></ul> |
| Uses           | Use compound documents to integrate OLE server functionality into your application.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| Ingredients    | OLE server application<br>PowerBuilder OLE control                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| Implementation | Window painter<br>DataWindow painter<br>User Object painter                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| Documentation  | To learn about object linking and embedding, see <i>Application Techniques</i> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |

## ActiveX controls

|             |                                                                                                                                                                                                                                                                                                                                                        |
|-------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Description | You can add ActiveX controls to a window or user object. An ActiveX control is a server itself that processes user actions according to scripts you program in PowerBuilder. These scripts call functions and set properties that belong to the ActiveX. Typically, an ActiveX control presents its own user interface, although this is not required. |
| Uses        | Use ActiveX controls to enhance your application with precoded functionality. ActiveX controls range from simple visual displays (such as a meter or a gauge) to single activities that are customizable (spellchecking words or phrases) to working environments (image acquisition with annotation and editing).                                     |

Illustration



- Ingredients** ActiveX control  
PowerBuilder window or user object
- Implementation** Window painter  
User Object painter
- Documentation** To learn about ActiveX controls in PowerBuilder, see *Application Techniques*.

## OLE in DataWindows

- Description** A DataWindow can include an object that is a container for an OLE object. The container stores information about the application that created the object, and it can launch the application to display or modify the OLE object.
- The container can fill the whole DataWindow (when you create a new DataWindow using the OLE presentation style) or it can exist alongside other objects in a DataWindow (when you add an OLE object to an existing DataWindow). You can also read OLE data from a blob column in a database and display the objects in the DataWindow. These mechanisms support both OLE 2.0 and OLE 1.0 servers.
- Uses** Use OLE in DataWindows to display data in an OLE server application and to display blob columns.
- Ingredients** OLE server application  
DataWindow object
- Kinds** You can use OLE objects in DataWindows in the following ways:

- ◆ **OLE object in a DataWindow** The OLE object is displayed in its container object with the data and other objects, such as bitmaps or text. You can associate it with data in a particular row, the rows on a page, or with all rows. You choose which columns in the DataWindow are transferred to the OLE object. You can add an OLE container object to a DataWindow that uses any presentation style that supports multiple DataWindows (this does not include Graph and RichTextEdit presentation styles).
- ◆ **OLE presentation style** The OLE presentation style is similar to an OLE object in a DataWindow. The difference is that the OLE container is *the only* object in the DataWindow. The underlying data is *not* presented in column objects, and *there are no other* objects such as bitmaps or text. The OLE object is *always* associated with all rows in the DataWindow.
- ◆ **OLE database blob column** OLE objects that are stored in the database in a blob column are displayed in each row of the DataWindow.

Implementation

DataWindow painter

Documentation

To learn about OLE in DataWindows, see *Application Techniques*.

## Automation

Description

Automation allows applications to use services from other COM objects. PowerBuilder can function as an automation client component or as an out-of-process automation server component.

Uses

Use automation to call functions and manipulate properties in all types of local and remote COM objects. You can use PowerBuilder to create both client components and server components.

You use automation (formerly called OLE automation) to manipulate:

- ◆ Insertable objects in the PowerBuilder OLE control
- ◆ ActiveX controls
- ◆ Programmable objects in server components

Ingredients

Server application (which could be PowerBuilder)  
PowerBuilder OLE control  
Client application (which could be PowerBuilder)

Implementation

Window painter

User Object painter  
PowerScript painter

Documentation To learn about automation in PowerBuilder, see *Application Techniques*.

## Structured storage

**Description** In addition to COM interfaces for controls and automation, PowerBuilder provides an interface to the underpinnings of COM-based data storage.

COM data is stored in objects called **streams**, which live in objects called **storages**. Streams and storages are analogous to the files and directories of a file system. By opening, reading, writing, saving, and deleting streams and storages, you can create, combine, and delete your COM objects. PowerBuilder provides access to storages and streams with the OLEStorage and OLEStream object types.

**Uses** When you define OLE 2.0 controls and OLEObject variables, you have full access to the functionality of server applications and OLE automation, which already provide you with much of OLE's power. You may never need to use PowerBuilder's storage and stream objects unless you want to construct complex combinations of stored data.

**Ingredients** PowerBuilder OLEStorage and OLEStream objects

**Implementation** PowerScript painter

**Documentation** To learn about structured storage in PowerBuilder, see *Application Techniques*.

## DDE

**Description** Dynamic Data Exchange (DDE) enables your application to converse with the external program by exchanging commands and data. Your application can act either as a DDE client (which asks the other program to perform some commands or provide some data) or as a DDE server (which responds to command or data requests from the other program).

**Uses** Although you can use DDE to communicate with external programs, you typically use OLE to perform this functionality.

PowerBuilder supports DDE on the Windows platforms and on UNIX when communicating with another application built with Wind/U from Bristol Technology.

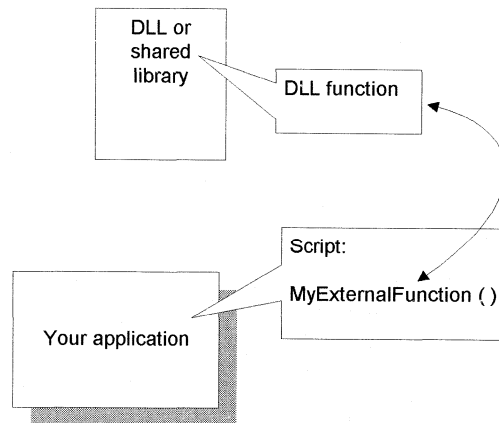
Documentation To learn about DDE, see *Application Techniques*.

## DLL and shared-library functions

Description Your application can call a function in a Windows DLL (dynamic link library) or shared library (on UNIX and Macintosh). To do that, you define and execute an **external function**.

Uses Sometimes you may want to implement a particular service that your application requires by executing a DLL function—a function located outside your application in a dynamic link library and written in a language other than PowerScript. Then you can execute it simply by calling that external function in a script the same way you call any other PowerBuilder function.

Illustration



Implementation PowerScript painter

Documentation To learn about external functions, see *Application Techniques*.

## C++ classes

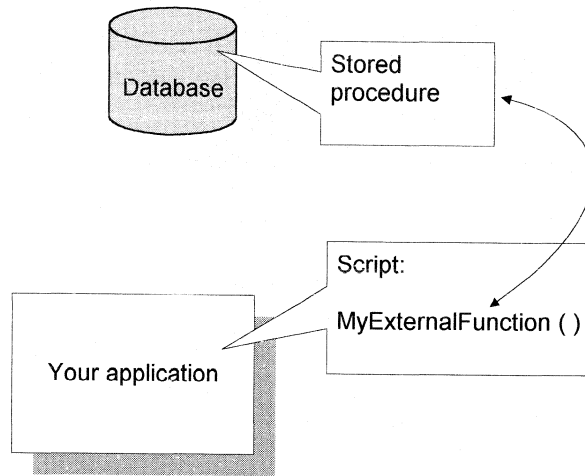
|                |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|----------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Description    | You can define C++ classes and then call their methods from your PowerBuilder application.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| Uses           | <p>To give you extra power for implementing application features in which execution speed is particularly critical or lower-level programming is required, Powersoft provides a facility called the <b>C++ Class Builder</b>. If you have this facility and you know how to code in C++, you can work with the User Object painter in PowerBuilder to define C++ classes, compile them, and store them in DLLs. Then you can execute the methods of those C++ classes from your PowerBuilder applications as user object functions.</p> <p>The C++ Class Builder is available on the Windows 95 and Windows NT platforms only.</p> |
| Implementation | User Object painter<br>C++ Class Builder                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| Documentation  | To learn about the C++ Class Builder, see <i>C++ Class Builder</i> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |

## Database stored procedures

|             |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|-------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Description | <p>If you're using a database that provides stored procedures, you may want to execute one or more of them to implement some of the services that your application requires. For stored procedures that don't return a result set, you can do this by defining external functions in PowerBuilder that refer to them (similar to what you do to access DLL functions, but with a few more housekeeping steps). Then you can execute those stored procedures by calling the corresponding external functions in your application's scripts.</p> <p>The process of executing a stored procedure this way is known as a <b>remote procedure call (RPC)</b>.</p> |
| Uses        | Use remote procedure calls to execute database stored procedures that don't return a result set.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |



## Illustration



## Implementation

PowerScript painter

## Documentation

To learn about remote procedure calls, see *Application Techniques*.

## Distributed PowerBuilder objects

## Description

An important system architecture topic is how an application's components are to be **partitioned** across your computing environment. Partitioning essentially means organizing an application so that the appropriate components reside and execute on the appropriate client or server computers. The different partitioning approaches you can take include:

| Partitioning approach | Description                                                                                                                                                                                                                                                                                                                                                     |
|-----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| One tier              | <p>This is the simplest approach because it doesn't actually involve partitioning. In a single-tier application, the user interface and logic components reside and execute on the client computer, and data is accessed from a local database</p> <p>This is appropriate for standalone applications that don't need to take advantage of server computers</p> |
| Two tier              | <p>This is the most basic client/server approach. In a two-tier application, the user-interface and logic components reside and execute on the client computer, but data is accessed from server databases</p>                                                                                                                                                  |

| Partitioning approach | Description                                                                                                                                                                                                                                                                                                                                                                                                          |
|-----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Multitier             | This is the full client/server approach, also known as the <b>distributed</b> approach. In a multitier application, the user-interface components reside and execute on the client computer along with some of the logic components. But other logic components reside and execute on server computers<br><br>Both the client and server components of a multitier application can access data from server databases |

Uses

PowerBuilder supports all three of these techniques. If you're building a small- or medium-size application, chances are you'll go with the one-tier or two-tier approach. The multitier approach is typically for large-scale applications. It adds complexity to your system but also a lot of power and versatility. To implement the multitier approach, you can use PowerBuilder's **distributed computing capabilities**.

---

**Web.PB**

Web.PB is a specialized usage of PowerBuilder's distributed computing capabilities, with the Web.PB CGI, NSAPI, or ISAPI interface program functioning as a distributed PowerBuilder client.

---

Ingredients

Server application  
Client application  
Communications drivers

Documentation

To learn about PowerBuilder's distributed computing capabilities, see *Application Techniques*.

# Output

## Printed reports

### Description

Users often want an application to generate one or more paper reports that they can distribute, mark up, or simply read later away from the computer. For instance, maybe they want to print an expense report, a parts list, a form letter, or a bill.

If your application needs to generate any reports, you've got to figure out what data to include in each one and how to format that data given the report's purpose.

To design reports in PowerBuilder, you use DataWindow objects. The advantage of using DataWindow objects is that you can create reports that contain both:

- ◆ The *intelligence* to access the appropriate database tables and retrieve the rows you want, *and*
- ◆ The *presentation abilities* to format this row data as you require

---

### Report objects

PowerBuilder also has a Report object, which is actually a specialized DataWindow object. A Report object has no database update capability; other than that, all other DataWindow features apply (data sources, presentation styles, and so on). You use Report objects in DataWindow controls and DataStores just the way you use DataWindow objects.

Within the context of printed reports, DataWindow objects and Report objects are identical.

---

### Uses

Once you've designed a report in PowerBuilder, you're ready to use it in an application. This typically involves:

- 1 Displaying the report in a window through a DataWindow control (just as you do for any other DataWindow).
- 2 Letting users do one or more of the following:
  - ◆ *Examine* the formatted report output in print preview mode
  - ◆ *Send* the formatted report output to a printer

- ◆ Save the formatted report output in one of a variety of file formats, including as a Powersoft report (PSR) file

PowerBuilder provides built-in functions that you can code in scripts to implement these services.

|                |                                                                                                                                                                                                                                            |
|----------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Kinds          | PowerBuilder supports a wide variety of presentation styles and also gives you the ability to customize just about any aspect of a particular report. For examples of selected presentation styles, see "Presentation styles" on page 111. |
| Implementation | DataWindow painter<br>Window painter<br>Report painter                                                                                                                                                                                     |
| Documentation  | To learn about creating reports in PowerBuilder, see <i>Application Techniques</i> .                                                                                                                                                       |

## File generation

|             |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|-------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Description | <p>You can generate files in various formats from your DataWindows.</p> <p>You can do this:</p> <ul style="list-style-type: none"><li>◆ Within the PowerBuilder development environment, <i>or</i></li><li>◆ Within an application (from a DataWindow control, DataStore, or DataWindow plug-in)</li></ul>                                                                                                                                                                                                                                                                                                      |
| Uses        | <p>PowerBuilder provides built-in functions that you can code in scripts to write (and in some cases read) many different kinds of files from your application:</p> <ul style="list-style-type: none"><li>◆ Tab-separated text files</li><li>◆ Microsoft Excel spreadsheets</li><li>◆ Lotus 1-2-3 spreadsheets</li><li>◆ HTML table syntax</li><li>◆ dBASE files</li><li>◆ The clipboard (of your operating environment)</li><li>◆ Powersoft report (PSR) files</li><li>◆ Comma-separated values (CSV)</li><li>◆ Windows metafiles (WMF)</li><li>◆ SQL syntax</li><li>◆ Data interchange format (DIF)</li></ul> |

|                |                                                                                                     |
|----------------|-----------------------------------------------------------------------------------------------------|
| Implementation | DataWindow painter<br>PowerScript painter                                                           |
| Documentation  | To learn about saving files from the DataWindow painter, see the <i>PowerBuilder User's Guide</i> . |

## PSR

|                |                                                                                                                                                                                                                                                                                                                                |
|----------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Description    | You can save the contents of a DataWindow as a Powersoft report (PSR) file. A PSR file contains the DataWindow definition as well as data.                                                                                                                                                                                     |
| Uses           | You can display a PSR file in the following places: <ul style="list-style-type: none"> <li>◆ The InfoMaker Report painter (if you double-click a PSR file and you have installed InfoMaker, InfoMaker opens automatically)</li> <li>◆ A DataWindow control</li> <li>◆ A DataStore</li> <li>◆ The DataWindow plug-in</li> </ul> |
| Implementation | DataWindow painter<br>DataWindow plug-in<br>PowerScript painter                                                                                                                                                                                                                                                                |
| Documentation  | To learn about PSR files, see the <i>PowerBuilder User's Guide</i> .                                                                                                                                                                                                                                                           |

## HTML

|                |                                                                                                                                                                                                                                                                        |
|----------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Description    | You can save the contents of a DataWindow in HTML format, as either a table or a form. The DataWindow object also contains a cascading style sheet to preserve certain formatting when viewed in a browser that supports style sheets.                                 |
| Uses           | You can generate the HTML as a string or as a file, depending on how you plan to use it: <ul style="list-style-type: none"> <li>◆ <b>Use Web.PB to return HTML dynamically</b> Generate a string variable</li> <li>◆ <b>As a static URL</b> Generate a file</li> </ul> |
| Ingredients    | DataWindow painter<br>PowerScript painter<br>Web browser                                                                                                                                                                                                               |
| Implementation | DataWindow painter                                                                                                                                                                                                                                                     |

PowerScript painter

Documentation

To learn about HTML and DataWindows, see *Application Techniques*.

## Printing

### DataWindows

Description

You can print the contents of a DataWindow (or Report object) from a DataWindow control, DataStore, or DataWindow plug-in.

Uses

Virtually all applications need to print reports. In addition to print capabilities, your application can also let the user view the DataWindow in print preview mode and provide previewing services, such as zooming.

Implementation

DataWindow painter  
PowerScript painter  
DataWindow plug-in

Documentation

To learn about printing DataWindows, see the *PowerScript Reference*.

### Rich text

Description

You can print the contents of a RichTextEdit control.

Uses

You use a RichTextEdit control to provide word processing capabilities within a PowerBuilder application. Print capabilities are required for this type of functionality.

Implementation

DataWindow painter  
Window painter

Documentation

To learn about RichTextEdit controls, see *Application Techniques*.

### Printer setup

Description

Your application can display the system's printer setup dialog.

Implementation

PowerScript painter

Documentation

To learn about displaying the printer setup dialog, see the *PowerScript Reference*.

## Print jobs

|                |                                                                                                                                                                       |
|----------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Description    | In addition to automatically printing a DataWindow's contents, PowerBuilder provides functions that allow you to manually construct pages and send them to a printer. |
| Implementation | PowerScript painter                                                                                                                                                   |
| Documentation  | To learn about printing, see <i>Application Techniques</i> .                                                                                                          |

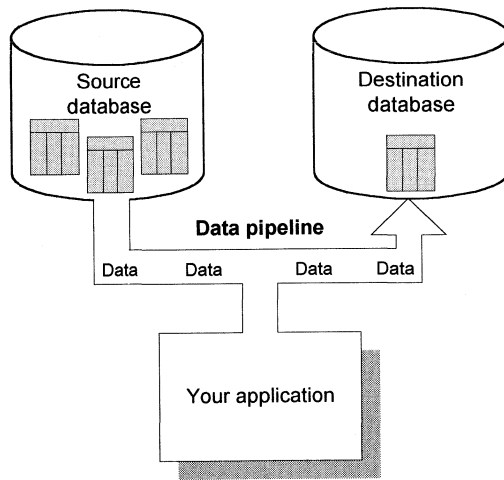
## Data pipelines

|             |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|-------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Description | Sometimes applications require the ability to migrate data from one place to another. To handle requirements such as these, PowerBuilder provides <b>data pipelines</b> . A data pipeline is an application component you can design to pump data from one or more source tables to a new or existing destination table. The source and destination tables you specify can be in either the same database or separate databases (even if they're different kinds of databases that involve different DBMSs). |
|-------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

Data pipelines are a lot like DataWindows when it comes to their built-in data access and manipulation intelligence. And the steps you need to take to make them work in your application are similar in several ways too.

|      |                                                                                                                                                                                                                                                                                                                                                                                                      |
|------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Uses | <p>You might use data pipelines to:</p> <ul style="list-style-type: none"><li>◆ <i>Create a new sales summary table</i> in your database by joining various regional sales tables and extracting particular data from them, <i>or</i></li><li>◆ <i>Copy a payroll table</i> from the server database to a local database so that the application can access it without needing the network</li></ul> |
|------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

Illustration



Implementation

Pipeline painter  
PowerScript painter

Documentation

To learn about data pipelines, see *Application Techniques*.



# Environment

## Application logistics

### Description

The Application object is the entry point into a standard PowerBuilder client/server application. It is a discrete object that is saved in a PowerBuilder library—just like a window, menu, user object, or DataWindow object. However, not all PowerBuilder applications use the Application object:

| Type                                        | Use Application object                                   |
|---------------------------------------------|----------------------------------------------------------|
| PowerBuilder client application             | Yes                                                      |
| Distributed PowerBuilder server application | Yes                                                      |
| PowerBuilder window plug-in application     | Yes (Open event, Close event, and global variables only) |
| PowerBuilder window ActiveX application     | Yes (Open event, Close event, and global variables only) |
| PowerBuilder automation server              | No                                                       |

It's in the application object that you define application-level characteristics and behaviors such as:

- ◆ The *list of libraries* (library search path) for your application
  - PowerBuilder uses this list to find objects when you're developing the application and executing the application; it searches through libraries in the order in which you list them
- ◆ The *icon* that your operating system is to display to represent the executable form of your application
- ◆ The *processing* that you want to perform when:
  - ◆ A user starts (opens) your application
  - ◆ A serious (system) error occurs while a user is in your application
  - ◆ A user ends (closes) your application
- ◆ The *variable types* you want to use by default in your application for certain system objects
- ◆ The *fonts* you want to use by default in your application for various purposes (including text, data, headings, and labels)

### Uses

The tasks that an Application object performs include:

- ◆ **Opening your application** The Application object provides an Open event that's triggered when a user starts your application. You must write a script for this event to specify the initial processing you want the application to perform (which typically includes opening a window).
- ◆ **Trapping system errors** The Application object provides a SystemError event that's triggered when a serious application error occurs during execution. By default, PowerBuilder displays a message box (with the appropriate error number and text) in response to such an error. But if you prefer, you can write a script for this event to perform your own error processing.
- ◆ **Closing your application** The Application object provides a Close event that's triggered when a user ends your application. You should write a script for this event to specify any cleanup processing you want the application to perform (which might include disconnecting from a database or writing to an INI file).

Implementation

Application painter  
User Object painter  
Window painter

Documentation

To learn about Application objects, see the *PowerBuilder User's Guide*.

## Application context

Description

PowerBuilder can run in many contexts, including:

- ◆ PowerBuilder execution time (default)
- ◆ PowerBuilder window plug-in
- ◆ PowerBuilder window ActiveX

The PowerBuilder Context object allows applications to access certain host (non-PowerBuilder) services. The Context object creates service objects appropriate for the current execution context (native PowerBuilder, Powerbuilder window plug-in, PowerBuilder window ActiveX). This allows your application to take full advantage of the execution environment.

The Context object services are:

- ◆ Context information service
- ◆ Keyword service
- ◆ Internet service (available on Windows 95 and Windows NT only)

|                |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|----------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Uses           | <p>Use this feature to enhance the capabilities of your applications. For example, by using the functions provided by the services, you can:</p> <ul style="list-style-type: none"> <li>◆ Determine the execution context, modifying the application's look, feel, and processing depending on the environment</li> <li>◆ Open the default browser, displaying a URL from within a PowerBuilder application</li> <li>◆ Control the browser from within a PowerBuilder application (Internet Explorer only)</li> <li>◆ Access application arguments and environment variables</li> </ul> <p>Powersoft may provide additional services after the release of PowerBuilder Version 6.0. Additionally, you can write your own services by creating descendants of the service object.</p> |
| Ingredients    | <ul style="list-style-type: none"> <li>◆ PowerBuilder execution environment</li> <li>◆ PowerBuilder window plug-in</li> <li>◆ PowerBuilder window ActiveX</li> <li>◆ Internet browser, such as Internet Explorer (Version 3.0 or higher) or Netscape Navigator (Version 3.0 or higher)</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| Implementation | PowerScript painter                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| Documentation  | To learn about the Context object, see <i>Application Techniques</i> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |

## Platform support

Description PowerBuilder provides application development capabilities on a variety of computing platforms:

| Product                  | Description                                                                                                                                                                                                                                                                                |
|--------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| PowerBuilder for Windows | <p>Lets you build graphical client/server applications on Microsoft Windows 3.x, Windows 95, and Windows NT</p> <p>You develop applications using PowerBuilder running on either Windows 95 or Window NT Version 4.0. You can deploy applications on these platforms or on Windows 3.x</p> |

| Product                    | Description                                                                                                                                                                                                                                                                         |
|----------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| PowerBuilder for Macintosh | Lets you build graphical client/server applications on the Apple Macintosh<br><br>You develop applications using PowerBuilder running on a PowerPC Macintosh using Mac OS Version 7.5 or higher. You can deploy applications on this platform or on 68K series Macintosh processors |
| PowerBuilder for UNIX      | Lets you build graphical client/server applications on UNIX<br><br>You develop and deploy applications using PowerBuilder running on any of these systems: <ul style="list-style-type: none"><li>◆ Sun Solaris</li><li>◆ HP/UX</li><li>◆ IBM AIX</li></ul>                          |

**Uses**

If you're developing applications for just one of these operating systems, you can simply use the appropriate PowerBuilder product and focus on that platform. But if your applications need to work on multiple operating systems, you can take advantage of the cross-platform features built into the platform-specific versions of PowerBuilder.

As you would expect, each platform-specific version of PowerBuilder can build applications for its own platform. But no matter which platform you're developing on, the PowerBuilder applications you create can be edited or deployed on any of the other platforms. So, for example, an application developed on PowerBuilder for Windows can run on the Macintosh and UNIX platforms as well.

**Documentation**

To learn how to build an application for multiple platforms, see *Application Techniques*.

**Portability**

**Description**

PowerBuilder provides binary compatibility of libraries across all the platforms. That means you can develop your application on one of these platforms and then immediately port it for further development or testing on any of the others.

When deploying a cross-platform application, you must decide whether to create Pcode or machine code:

- ◆ **Pcode** Portable across every platform supported by PowerBuilder, including Microsoft Windows (16-bit and 32-bit), Apple Macintosh, and UNIX. Pcode applications are generally smaller than machine code applications but may not perform as well.
- ◆ **Machine code** Requires you to generate and maintain complete versions of your executable application on each platform. But you may be willing to do this to ensure the best performance.

Documentation For more on Pcode and machine code, see "Code generation" on page 144.

## Cross-platform applications

Description You can also go a step further and code your application to take advantage of features specific to each platform. For instance, PowerBuilder lets you conditionally test for the current platform, so you can adjust application features accordingly.

Uses This enables you to build a cross-platform application—one that's maintained in a single code base but used on multiple platforms—without sacrificing functionality.

Implementation PowerScript painter

Documentation To learn about executing code for specific platforms, see *Application Techniques*.

## Platform features

Most application features are supported on every platform. But a few are platform specific. This table shows the features that PowerBuilder supports on the various platforms:

| Category | Feature                       | Win 3.1 | Win 95 and NT | Macintosh | UNIX |
|----------|-------------------------------|---------|---------------|-----------|------|
| General  | DDE                           | x       | x             |           |      |
|          | OLE                           | x       | x             |           |      |
|          | Rich text                     | x       | x             | x         | x    |
|          | Powersoft reports (PSR files) | x       | x             | x         | x    |
|          | Mail (MAPI)                   | x       | x             |           |      |

| Category              | Feature                                     | Win 3.1 | Win 95 and NT | Macintosh | UNIX |
|-----------------------|---------------------------------------------|---------|---------------|-----------|------|
|                       | C++ Class Builder                           | x       | x             |           |      |
|                       | External functions                          | x       | x             | x         | x    |
|                       | AppleScript                                 |         |               | x         |      |
|                       | PowerBuilder Foundation Class Library (PFC) | x       | x             | x         | x    |
| Image file formats    | RLE, BMP, and CUR images                    | x       | x             | x         | x    |
|                       | ICO images                                  | x       | x             | x         |      |
|                       | WMF images                                  | x       | x             |           | x    |
|                       | PICT images                                 |         |               | x         |      |
| Distributed computing | Client                                      | x       | x             | x         | x    |
|                       | Server                                      |         | x             |           | x    |
| Internet support      | Save rows as HTML                           | x       | x             | x         | x    |
|                       | Netscape plug-ins                           | x       | x             | x         |      |
|                       | PowerBuilder window ActiveX                 |         | x             |           |      |
|                       | Web.PB                                      |         | x             | x         | x    |

## International support

|               |                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|---------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Description   | <p>PowerBuilder includes features that enable you to internationalize and localize an application. These features include:</p> <ul style="list-style-type: none"> <li>◆ Unicode support</li> <li>◆ Japanese version of PowerBuilder</li> <li>◆ Right-to-left enabled PowerBuilder</li> <li>◆ Localized execution environment for certain European languages</li> <li>◆ Portability</li> <li>◆ Translation Toolkit</li> <li>◆ Localized PFC</li> </ul> |
| Documentation | For complete information, see "Internationalization" on page 21.                                                                                                                                                                                                                                                                                                                                                                                      |

## Initialization files and registries

You may want your application to store user preferences or default settings across sessions. A typical way to do this is to use an initialization (INI) file or (on Microsoft Windows 95 and Windows NT) the registry.

### INI files

|                                                                                                                                                   |                                                                                                                |
|---------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------|
| Description                                                                                                                                       | PowerBuilder provides functions that enable your application to read from and write to INI files.              |
| Uses                                                                                                                                              | You typically use INI files to store database connection settings, application settings, and user information. |
| <hr/>                                                                                                                                             |                                                                                                                |
| <p><b>The PowerBuilder initialization file</b></p> <p>PowerBuilder uses an initialization file to store application and database information.</p> |                                                                                                                |
| <hr/>                                                                                                                                             |                                                                                                                |
| Implementation                                                                                                                                    | PowerScript painter                                                                                            |
| Documentation                                                                                                                                     | To learn about using INI files, see <i>Application Techniques</i> .                                            |

## The Windows registry

|                |                                                                                                                                 |
|----------------|---------------------------------------------------------------------------------------------------------------------------------|
| Description    | PowerBuilder provides functions that enable your Windows 95 and Windows NT applications to read from and write to the registry. |
| Uses           | You typically use the registry to store database connection settings, application settings, and user information.               |
| Implementation | PowerScript painter                                                                                                             |
| Documentation  | To learn about using the registry, see <i>Application Techniques</i> .                                                          |

## Code generation

|             |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|-------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Description | <p>You can generate <b>executable files</b> and/or <b>dynamic libraries</b>. You can include <b>resources</b> in those files or keep them as separate files.</p> <p>When you plan an application, one of the fundamental issues to think about is the compiler format in which you'll want that application generated. PowerBuilder offers two alternatives from which you can pick:</p> <ul style="list-style-type: none"><li>◆ <b>Pcode</b> An interpreted language that's supported on all PowerBuilder platforms</li><li>◆ <b>Machine code</b> Platform-specific executable code</li></ul>                                                                                                                                                                                                                                                             |
| Uses        | <p>Use these guidelines to help you decide whether Pcode or machine code is right for your project:</p> <ul style="list-style-type: none"><li>◆ <b>Speed</b> If your primary goal is to optimize execution speed, then choose machine code. It will perform better than Pcode during many kinds of processing.<br/><i>Exception:</i> Pcode does have one speed advantage—it is faster to <i>generate</i> than machine code. That makes it especially handy for developers when they want to quickly create an executable version of an application for testing.</li><li>◆ <b>Size</b> The files generated for Pcode are smaller than those generated for machine code. If your application is to be deployed on computers where file size is a major issue, then you might decide to give up the speed of machine code and choose Pcode instead.</li></ul> |



- ◆ **Portability** Pcode can be useful for cross-platform applications. In fact, it is portable across every platform supported by PowerBuilder, including Microsoft Windows (16-bit and 32-bit), Apple Macintosh, and UNIX.

Machine code is, of course, platform-specific. That means it requires you to generate and maintain complete versions of your executable application on each platform. But you may be willing to do this to ensure the best performance.

## Ingredients

An executable application that you create in PowerBuilder can consist of one or more of the following pieces:

- ◆ **An executable file** You will *always create exactly one* executable (EXE) file for any PowerBuilder application you want to deploy.

At minimum, this file contains code that enables your application to run as a native application on its target platform. Depending on the packaging model you choose for your application, the executable file may also contain compiled versions of objects, an execution library list, and resources (such as bitmaps) used by the application.

- ◆ **Dynamic libraries** You can optionally reduce the executable file's size by creating PowerBuilder dynamic libraries for the application's PBLs. Pcode dynamic libraries use the PBD file extension; machine code dynamic libraries use the DLL extension.
- ◆ **Resources** You can optionally include resource files (such as DataWindow objects, bitmaps, icons, and cursors) in the executable file or dynamic libraries. This allows the application to find resources that are not part of the application and saves you the trouble of deploying the individual bitmap, icon, and cursor files.

## Implementation

Project painter

## Documentation

To learn about PowerBuilder code generation, see the *PowerBuilder User's Guide*.

## Pcode

### Description

Pcode (pseudocode) is an interpreted language that's supported on all PowerBuilder platforms. It's the same format that PowerBuilder uses in libraries (PBL files) to store individual objects in an executable state.

Advantages of Pcode include its small size and portability.

|                |                                                                                                                         |
|----------------|-------------------------------------------------------------------------------------------------------------------------|
| Kinds          | A Pcode application can use either a single executable (EXE) file or a small executable file and one or more PBD files. |
| Implementation | Project painter                                                                                                         |
| Documentation  | To learn about Pcode executable files, see <i>Application Techniques</i> .                                              |

## **Machine code**

|                |                                                                                                                                                                                                                                                              |
|----------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Description    | Another route you can take is to generate your executable application in machine code. The key advantage of machine code is speed of execution.                                                                                                              |
| Ingredients    | PowerBuilder uses different compiler technology depending on the development platform: <ul style="list-style-type: none"><li>◆ 32-bit Windows</li><li>◆ 16-bit Windows</li><li>◆ Macintosh</li><li>◆ Sun Solaris</li><li>◆ HP-UX</li><li>◆ IBM AIX</li></ul> |
| Kinds          | A machine code application can use either a single executable (EXE) file, or a small executable file plus one or more DLL files.                                                                                                                             |
| Implementation | Project painter                                                                                                                                                                                                                                              |
| Documentation  | To learn about machine code executable files, see <i>Application Techniques</i> .                                                                                                                                                                            |

# How You Build It

## About this chapter

Once you know the kind of application you want, you'll use the tools in PowerBuilder to develop it. This chapter describes how you'll work with those tools (and related products) over the course of your development project.

## Contents

| Topic                  | Page |
|------------------------|------|
| Using tools            | 148  |
| Using objects          | 151  |
| Testing and deployment | 158  |

## Using tools

### Setting up PowerBuilder

| <b>When you want to</b>                                       | <b>Use</b>                                                                                       |
|---------------------------------------------------------------|--------------------------------------------------------------------------------------------------|
| Install PowerBuilder and related tools                        | Setup program (platform-specific)<br><br>FOR INFO See the <i>PowerBuilder Installation Guide</i> |
| Configure PowerBuilder toolbars                               | Toolbars dialog<br><br>FOR INFO See the <i>PowerBuilder User's Guide</i>                         |
| Customize a particular painter (such as the Window painter)   | The painter's design options                                                                     |
| Customize general features of PowerBuilder                    | The PowerBuilder System Options dialog box                                                       |
| Customize the PowerBuilder toolbars (PowerBar and PainterBar) | The Toolbars facility (available in the painters as well as when no painters are open)           |

### Getting Help and documentation

| <b>When you want to</b>                                      | <b>Use</b>                                                                       |
|--------------------------------------------------------------|----------------------------------------------------------------------------------|
| Learn about new features or PowerScript functions and events | Online Help                                                                      |
| Learn how to use PowerBuilder                                | Powersoft Online Books                                                           |
| Get the latest news on all Powersoft products                | Powersoft Web site                                                               |
| Get updates to online books                                  | Powersoft online books Web site                                                  |
| Borrow information or syntax from PowerBuilder documentation | Copy it from the Powersoft Online Books or online Help                           |
| Provide site-specific online Help to your developers         | The User button in online Help<br><br>FOR INFO See <i>Application Techniques</i> |

## Using CASE tools

| When you want to                                                                                     | Use                                                                                                                                                                                                                                                       |
|------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Enhance the development process by using a design or CASE (Computer Aided Software Engineering) tool | PowerDesigner<br>FOR INFO To learn about:<br><ul style="list-style-type: none"> <li>◆ PowerDesigner products and features, see the Powersoft Web site</li> <li>◆ Using PowerDesigner once you have it, see the PowerDesigner documentation set</li> </ul> |

## Connecting to databases within PowerBuilder

| When you want to                             | Use                                                              |
|----------------------------------------------|------------------------------------------------------------------|
| Install appropriate database interfaces      | Setup program<br>FOR INFO See <i>Connecting to Your Database</i> |
| Configure ODBC data sources (if appropriate) | Configure ODBC dialog box                                        |
| Define and use database profiles             | Database Profiles dialog box                                     |

## Managing databases

| In this situation                         | When you want to                                  | Use                                       |
|-------------------------------------------|---------------------------------------------------|-------------------------------------------|
| SQL Anywhere (Windows and Macintosh only) | Create and delete SQL Anywhere databases          | Database painter                          |
|                                           | Perform full management of SQL Anywhere databases | Sybase SQL Central utility (Windows only) |
| Security                                  | Define database security                          | Database Administration painter           |

## Managing tables and views

| When you want to                   | Use              |
|------------------------------------|------------------|
| Create and control tables or views | Database painter |
| Define or edit table details       | Table painter    |

## Managing extended attributes

| When you want to                                            | Use                               |
|-------------------------------------------------------------|-----------------------------------|
| Maintain edit styles, display formats, and validation rules | Database painter (and repository) |

| <b>When you want to</b>                                  | <b>Use</b>                                                                          |
|----------------------------------------------------------|-------------------------------------------------------------------------------------|
| Define extended attributes for table columns             | Table painter (and repository)                                                      |
| Set access options for the repository                    | Database painter<br>(Design>Options)                                                |
| Generate reports about extended attributes in a database | PowerBuilder Extended Attribute Reporter (from the Advanced PowerBuilder Utilities) |

Manipulating table data

| <b>When you want to</b>                        | <b>Use</b>                |
|------------------------------------------------|---------------------------|
| Retrieve, insert, update, delete, or save rows | Data Manipulation painter |

Migrating tables

| <b>When you want to</b>                                                                                                           | <b>Use</b>            |
|-----------------------------------------------------------------------------------------------------------------------------------|-----------------------|
| Migrate data from one or more source tables to a new or existing destination table (either within a database or across databases) | Data Pipeline painter |

Testing and executing SQL

| <b>When you want to</b>           | <b>Use</b>                      |
|-----------------------------------|---------------------------------|
| Test SQL statements interactively | Database Administration painter |

## Using objects

Painting the user interface

| <b>Situation</b>             | <b>When you want to</b>                                                              | <b>Use</b>                                           |
|------------------------------|--------------------------------------------------------------------------------------|------------------------------------------------------|
| Windows, controls, and menus | Paint windows and controls                                                           | Window painter                                       |
|                              | Build your own controls                                                              | User Object painter                                  |
|                              | Paint menus and toolbars                                                             | Menu painter                                         |
| Other components and files   | Use ActiveX controls from the collection provided with PowerBuilder                  | Component Gallery for Powersoft Tools                |
|                              | Use picture files (icons and bitmaps) from the collection provided with PowerBuilder | Art Gallery                                          |
|                              | Edit icons, bitmaps, and cursors                                                     | Watcom Image Editor (installed with the Art Gallery) |

Painting the application interface

| <b>When you want to</b>                                                         | <b>Use</b>          |
|---------------------------------------------------------------------------------|---------------------|
| Extend standard PowerBuilder nonvisual classes                                  | User Object painter |
| Encapsulate application logic or validation rules in reusable nonvisual classes | User Object painter |

Coding PowerScript

| <b>Situation</b> | <b>When you want to</b>                                                                      | <b>Use</b>                                              |
|------------------|----------------------------------------------------------------------------------------------|---------------------------------------------------------|
| Objects          | Display information (properties, events, functions, variables, and structures) about objects | The Browser                                             |
|                  | Display inheritance information about class hierarchies                                      | The Browser (select Show Hierarchy from the popup menu) |
|                  | Display OLE object information (classes, properties, events, functions)                      | The Browser (OLE tab)                                   |
|                  | Find a particular text string in objects                                                     | Library painter (Entry>Search)                          |

| Situation            | When you want to                                             | Use                                                                                                                                                |
|----------------------|--------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------|
|                      | Get reports about objects                                    | The Browser (display the object information you want, and select Document from the popup menu)<br>The Library painter (Entry>Print)                |
|                      | See the ancestor version of a script you're editing          | PowerScript painter (Design>Display Ancestor Script)                                                                                               |
|                      | Track object access in a multiple-developer environment      | Library painter                                                                                                                                    |
|                      | Track versions of objects                                    | Library painter                                                                                                                                    |
|                      | View a complete or selected list of the objects in a library | Library painter                                                                                                                                    |
| Events and functions | Define global functions                                      | Function painter                                                                                                                                   |
|                      | Define user-defined events                                   | Window painter, User Object painter, Menu painter, Application painter (Declare>User Events)                                                       |
|                      | Define object functions                                      | Window painter, User Object painter, Menu painter, Application painter (Declare> <i>objecttype</i> Functions, which displays the Function painter) |
| Variables            | Declare local variables                                      | PowerScript painter                                                                                                                                |
|                      | Declare instance, shared, and global variables               | Window painter, User Object painter, Menu painter, Application painter                                                                             |



| Situation          | When you want to                                                                                                | Use                                                                                      |
|--------------------|-----------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------|
| External functions | Declare global external functions (for DLL functions, shared-library functions, and database stored procedures) | Window painter, User Object painter, Menu painter, Application painter, Function painter |
| Structures         | Define structures for objects                                                                                   | Window painter, User Object painter, Menu painter, Application painter, Function painter |
|                    | Define global structures                                                                                        | Structure painter                                                                        |

### Building database access

| Situation                       | When you want to                                                                                                          | Use                                                                                   |
|---------------------------------|---------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------|
| Embedded SQL                    | Code embedded SQL in event and function scripts                                                                           | PowerScript painter (invoked whenever you edit scripts in one of the object painters) |
| Data-access objects             | Paint DataWindow objects                                                                                                  | DataWindow painter                                                                    |
|                                 | Paint Report objects                                                                                                      | Report painter or InfoMaker                                                           |
|                                 | Paint data pipeline objects                                                                                               | Data Pipeline painter                                                                 |
|                                 | Develop canned queries for use in other painters                                                                          | Query painter                                                                         |
| DataWindow development services | Generate syntax to report on and manipulate properties of DataWindow objects                                              | DWSyntax utility (dwsyn60.exe)                                                        |
|                                 | Generate PowerScript statements to override default DataWindow behavior and update the database through stored procedures | Stored Procedure Update for DataWindows (Advanced PowerBuilder Utilities)             |
|                                 | Match properties of a DataWindow object with extended attributes from the repository                                      | DataWindow Extended Attribute Synchronizer (Advanced PowerBuilder Utilities)          |
|                                 | Check the validity of SQL statements used by DataWindow objects                                                           | DataWindow SQL Verifier (Advanced PowerBuilder Utilities)                             |

| Situation    | When you want to                                                                                     | Use                                                                                                               |
|--------------|------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------|
| Output files | Generate output files (PSR, HTML, or other formats) from DataWindows, reports, queries, or databases | DataWindow painter, Report painter, Query painter, or Data Manipulation painter (File>SaveRowsAs in preview mode) |

Developing nonvisual classes

| When you want to                                                                                                                           | Use                 |
|--------------------------------------------------------------------------------------------------------------------------------------------|---------------------|
| Extend PowerBuilder standard class user objects                                                                                            | User Object painter |
| Create custom class user objects that encapsulate application logic or validation rules (for use locally or with distributed applications) | User Object painter |
| Create an extensible attribute-only object to use instead of a global structure                                                            | User Object painter |

Using other languages

| When you want to          | Use                                         |
|---------------------------|---------------------------------------------|
| Develop C++ classes       | C++ Class Builder (via User Object painter) |
| Write AppleScript scripts | File editor                                 |

Specifying application logistics

| When you want to                                                        | Use                                                                   |
|-------------------------------------------------------------------------|-----------------------------------------------------------------------|
| List the PBLs that contain objects used in the application              | Application painter (does not apply to all application architectures) |
| Specify the icon associated with an application                         | Application painter                                                   |
| Override the variable types for an application's default system objects | Application painter                                                   |
| Specify default fonts for an application                                | Application painter                                                   |

Generating a quick application

| When you want to                                                        | Use                                                                                 |
|-------------------------------------------------------------------------|-------------------------------------------------------------------------------------|
| Quickly create the basic components of a skeletal MDI-style application | The quick application feature (and application template) in the Application painter |

Translating application text

| <b>When you want to</b>                                                                      | <b>Use</b>          |
|----------------------------------------------------------------------------------------------|---------------------|
| Translate application text to another language to create localized PowerBuilder applications | Translation Toolkit |

Editing text files

| <b>When you want to</b> | <b>Use</b>  |
|-------------------------|-------------|
| Edit text files         | File editor |

Opening objects

| <b>When you want to</b>                     | <b>Use</b>                                                                                                                                            |
|---------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------|
| Open up an object in a PowerBuilder painter | PowerBar, Library painter, Browser, Application painter, File menu (most recently edited objects), or the File menu of the appropriate object painter |

Getting object information

| <b>Situation</b> | <b>When you want to</b>                                                            | <b>Use</b>                                          |
|------------------|------------------------------------------------------------------------------------|-----------------------------------------------------|
| Browsing         | Display or copy/paste object properties, events, functions, and variables          | The Browser                                         |
|                  | Display OLE object information (classes, properties, events, functions)            | The Browser (OLE tab)                               |
|                  | Display the inheritance hierarchy for objects                                      | The Browser                                         |
| Reporting        | Print the contents of an object                                                    | Library painter (or the appropriate object painter) |
|                  | Print an object's hierarchy, properties, instance variables, events, and functions | The Browser                                         |
|                  | Copy object information to the clipboard or save it as an RTF file                 | The Browser                                         |

| Situation | When you want to                                     | Use                                                                                              |
|-----------|------------------------------------------------------|--------------------------------------------------------------------------------------------------|
|           | Report on object interdependencies in an application | PowerBuilder Cross Reference (Advanced PowerBuilder Utilities)                                   |
|           | Report on obsolete function usage                    | Migration Assistant                                                                              |
| Searching | Search for text in objects                           | Library painter (Entry>Search) or PowerBuilder Object Searcher (Advanced PowerBuilder Utilities) |
|           | Search for objects in libraries                      | PowerBuilder Object Searcher (Advanced PowerBuilder Utilities)                                   |

Managing objects and libraries

| When you want to                                         | Use                                             |
|----------------------------------------------------------|-------------------------------------------------|
| Create a new library                                     | Library painter, Application painter            |
| Copy, move, and delete objects                           | Library painter                                 |
| Generate a PBD or DLL from a single PowerBuilder library | Library painter                                 |
| Create PBDs or DLLs for all the PBLs in an application   | Project painter                                 |
| Export an object to source                               | Library painter                                 |
| Import an object from source                             | Library painter                                 |
| Control the objects listed in the Library painter        | Library painter Options dialog (Design>Options) |

Using source control

| When you want to                | Use                                            |
|---------------------------------|------------------------------------------------|
| Check an object out             | Library painter (native PowerBuilder facility) |
| Check an object in              | Library painter (native PowerBuilder facility) |
| Specify a source control system | Library painter                                |

| <b>When you want to</b>                              | <b>Use</b>                                                                            |
|------------------------------------------------------|---------------------------------------------------------------------------------------|
| Advanced source control features, such as versioning | ObjectCycle or one of the other third-party source control interfaces to PowerBuilder |

## Regenerating objects

| <b>When you want to</b>                                                         | <b>Use</b>                                                                              |
|---------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------|
| Regenerate selected objects                                                     | Library painter (Entry>Regenerate)<br>FOR INFO See the <i>PowerBuilder User's Guide</i> |
| Regenerate objects in an inheritance hierarchy                                  | Browser                                                                                 |
| Regenerate (rebuild) any objects affected by changes in the current application | Library painter (Design>Incremental Rebuild)                                            |
| Regenerate (rebuild) all objects in the current application                     | Library painter (Design>Full Rebuild)                                                   |
| Optimize a PBL's internal structure                                             | Library painter (Library>Optimize)                                                      |

## Testing and deployment

### Running

| When you want to                                                                                        | Use                                                                                                             |
|---------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------|
| Execute the application from the development environment                                                | Run facility                                                                                                    |
| Execute a specific window from the development environment                                              | Run Window facility                                                                                             |
| Preview an object as you paint it                                                                       | The Preview facility in the appropriate painter                                                                 |
| Execute a report in print preview mode                                                                  | Run Report facility                                                                                             |
| Test and analyze SQL statements interactively                                                           | The Database Administration painter                                                                             |
| Examine the SQL statements generated by a DataWindow before they are sent to the database for execution | The SQLPreview event of the appropriate DataWindow control<br><br>FOR INFO See the <i>PowerScript Reference</i> |
| Trap execution-time errors in DataWindow data expressions or property expressions                       | The Error event of the appropriate DataWindow control                                                           |
| Trap database retrieval or update errors encountered by a DataWindow                                    | The DBError event of the appropriate DataWindow control                                                         |
| Trace the interactions between your application and a database it accesses                              | The Database Trace tool or the ODBC Driver Manager Trace<br><br>FOR INFO See <i>Connecting to Your Database</i> |
| Examine values in your test database                                                                    | The Database Administration painter, Data Manipulation painter, or Query painter                                |

## Debugging

| When you want to                                                                                                                                                                                                                                                                                                                                                                  | Use                                                           |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------|
| Test run the application in <b>debug mode</b> —to be able to: <ul style="list-style-type: none"> <li>◆ <i>Stop</i> it at breakpoints</li> <li>◆ <i>Step</i> through its script code line by line</li> <li>◆ <i>Look</i> at the contents of variables</li> <li>◆ <i>Modify</i> the contents of variables for test purposes</li> <li>◆ <i>Watch</i> how variables change</li> </ul> | Debugger                                                      |
| Jump into the debugger while running an application in the development environment                                                                                                                                                                                                                                                                                                | The Debug button on the Terminate This Application dialog box |
| Trap serious errors when running the application (so that you can handle them with code of your own instead of relying on the PowerBuilder default execution-time error processing)                                                                                                                                                                                               | The SystemError event of the Application object               |

## Tracing and profiling

| Situation   | When you want to                                               | Use                                                                                                                      |
|-------------|----------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------|
| Application | Trace an entire application                                    | System Options dialog box, Profiling tab                                                                                 |
|             | Trace a portion of an application (programmatic control)       | StartTrace, EndTrace, and other trace-related PowerScript functions<br><br>FOR INFO See the <i>PowerScript Reference</i> |
|             | Trace a portion of an application (graphical interface)        | The w_starttrace window (provided with the Application Profiler)                                                         |
|             | Analyze and display trace results                              | The Application Profiler (profile.exe)                                                                                   |
|             | Create a text-based trace file                                 | The execution trace facility (can be activated on the System Options dialog box)                                         |
| Database    | Trace the interactions between your application and a database | Database Trace tool or ODBC Driver Manager Trace                                                                         |

Generating the executable application

| When you want to                                                                      | Use                                                           |
|---------------------------------------------------------------------------------------|---------------------------------------------------------------|
| Generate an executable file and dynamic libraries                                     | Project painter<br>FOR INFO See <i>Application Techniques</i> |
| Generate dynamic libraries                                                            | Library painter                                               |
| Generate user objects for use on the client side of a distributed application         | Project painter                                               |
| Include bitmaps, icons, and cursors with the executable file and/or dynamic libraries | PowerBuilder resource (PBR) files                             |
| Generate a C++ interface for by C++ client programs                                   | Project painter                                               |

Installing the application

| When you want to                                                                    | Use                                                                                                       |
|-------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------|
| Build a complete setup program                                                      | InstallShield Express or PBSetup (Windows only)                                                           |
| Install the appropriate PowerBuilder Deployment Kit                                 | InstallShield Express or PBSetup (Windows only)                                                           |
| Install database client software                                                    | DBMS-specific client software installation procedure                                                      |
| Install database interfaces                                                         | InstallShield Express or PBSetup (Windows only)                                                           |
| Configure ODBC drivers, updating registry or INI files (ODBCINST.INI and ODBC.INI ) | InstallShield Express or PBSetup (Windows only)                                                           |
| Distribute updated deployment modules or a new version of your application          | The Synchronizer                                                                                          |
| Install the PowerBuilder window plug-in or DataWindow plug-in                       | InstallShield Express or PBSetup (Windows only)                                                           |
| Install the PowerBuilder window ActiveX                                             | InstallShield Express or PBSetup (Windows only) or use the CODEBASE attribute to install it automatically |
| Install the PowerBuilder automation server component                                | PBGenReg program                                                                                          |



## Updating deployed files

| When you want to                                                            | Use                                                                  |
|-----------------------------------------------------------------------------|----------------------------------------------------------------------|
| Update a machine with the latest versions of specified files (such as DLLs) | The Synchronizer<br>FOR INFO See the <i>Synchronizer Users Guide</i> |

## Migrating libraries

| When you want to                                   | Use                                 |
|----------------------------------------------------|-------------------------------------|
| Migrate libraries to a new release of PowerBuilder | Library painter<br>(Design>Migrate) |

## Managing releases

| Situation                 | When you want to                                                                                | Use                                                                                                                                                                                                        |
|---------------------------|-------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Beginning a new release   | Set up libraries and source control files to begin developing a new version of an application   | If using a source control system: Library painter (Source>Create New Release)<br><br>If <i>not</i> using a source control system: the operating system (to make new copies of the application's PBL files) |
| Restoring a prior release | Recreate libraries from a previous version of an application (if using a source control system) | Project painter<br>(Design>Restore Libraries)                                                                                                                                                              |



# Index

## A

- ActiveX automation 68
- ActiveX controls
  - about 123, 160
  - PowerBuilder window ActiveX 8, 63
- ActiveX, samples 52
- AIX, version supported 19
- ancestor events, returning values 44
- AncestorReturnValue variable 44
- AppleScript scripts
  - editing 120
  - executing 120
- application
  - closing 138
  - coding final processing for 137
  - coding initial processing for 137
  - coding system error processing for 137
  - cross-platform 139
  - distributed 129
  - fonts for 137
  - icon 160
  - library search path for 137
  - opening 138
  - organizing objects for 96
  - specifying logistics for 137
  - tiers 129
  - variable types for 137
  - version control 42
- application context 138
- application framework
  - about 101
  - enhancing with special-purpose libraries 103
- Application objects, source control 42
- Application painter
  - specifying library search path in 137
  - using to create a new library 156
- Application Profiler 159
- application programming interface 86
- Arabic language, support for 23

- Art Gallery 151
- ASCII, saving as 37
- asynchronous calls
  - about 4
  - and server push 3
- automation
  - about 125
  - ActiveX 68
  - OLE 122

## B

- base classes 101
- border painting, and DataWindows 37
- breakpoints, setting 159
- Browser, new features 40
- Button object 34

## C

- C++ Class Builder 128
- charting tool 53
- CheckBox controls 79
- checkboxes, centering 35
- child windows 76
- class definition information 46
- class library
  - acquiring 103
  - developing 102
  - PowerBuilder Foundation Class Library 103
- clipboard 119, 132
- Close event for an application
  - about 138
  - coding 138
- code generation 144
- CommandButton controls 79
- compiling, options for 144
- Component Gallery for Powersoft Tools 52, 151



errors  
 execution-time 158  
 handling application system errors 137

events  
 about 88  
 ancestor return values 44  
 Close for an application 138  
 coding scripts for 89  
 defining your own 90  
 Open for an application 138  
 SystemError for an application 138, 159

Excel5, SaveAs support 37

executable files  
 about 145  
 delivering 160

executable version of an application  
 compile options for 144  
 what goes in it 145, 160

execution  
 database trace tool 158  
 debug mode 159  
 ending an application 137  
 library list 145  
 regular mode 158  
 role of application object 137  
 starting an application 137

exporting files from an application 132

**F**

files  
 executable 145  
 reading 132  
 writing 132

font mapping 19

functions, external  
 using to execute database stored procedures 128  
 using to execute DLL functions 127

functions, PowerScript  
 about 92  
 DoScript 120  
 file manipulation 132

functions, user-defined 90

**G**

garbage collection 51  
 GenerateCSS property 11  
 generators in Project painter 18  
 Graph controls 79  
 graphing tool 53  
 GroupBox controls 80  
 GroupBox object 35

**H**

Hebrew language, support for 23

Help  
 files, delivering with an application 160  
 providing application Help to users 120

HOW Learning Edition 52

HP-UX, version supported 19

HScrollBar controls 79

HTML  
 DataWindow enhancements 10  
 forms 13  
 from DataWindow 133  
 tables 12

**I**

IBM AIX, version supported 19

ICO files, delivering as resources 160

icon, application 160

Image Editor 151

indexes, defining 149

inheritance 94

INI files  
 delivering with an application 160  
 usage 143

IntelliMouse Pointing Device 42

Internet  
 DataWindow plug-in 66  
 PowerBuilder window ActiveX 63  
 PowerBuilder window plug-in 65  
 Web.PB 62

Internet service 15, 138

Internet Tools, new features 7

## J

- Japanese DBCS, support for 22
- just-in-time debugging 32

## K

- keys, defining 149
- keyword service 15, 138

## L

- last-compiled timestamp 41
- libraries
  - about 96
  - class libraries 102
  - cross-platform use of 97
  - files for 96
  - listing those an application is to use 137
  - special-purpose 103
- Library painter
  - about 96
  - new features 29, 41, 42
  - using to create a new library 156
- library search path
  - defining 137
  - use in executable application 145
- Line controls 79
- ListBox controls 79
- Listview controls 79
- load balancing 5
- local databases, installing for users 160
- localized deployment kits 21
- logic
  - coding in scripts 89
  - coding in user-defined functions 90
- logistics of an application 137
- Lotus Notes, PowerBuilder Library for 103

## M

- machine code 144
- mail system, accessing 121
- main windows 74

- maintenance of an application, delivering updated
  - application files 160
- MAPI 121
- MDI 83
- MDI frames
  - about 77
  - defining toolbars for 82
- MDI sheets 77
- menus 81, 82
- message boxes 75
- migrating tables within or between databases 135, 150
- Migration Assistant 156
- MultiLineEdit controls 79
- multitier applications 60, 129

## N

- Name Server 5
- nonvisual classes 100
- n-up row selection, highlighting 38

## O

- Object Browser, new features 40
- object management, with libraries 96
- ObjectCycle 157
- object-oriented programming
  - class library usage 101
  - how objects work 94
  - PowerBuilder implementation 98
- objects
  - about 94
  - class definition information 46
  - in an executable file 145
  - previewing 158
  - storing 96
- OCX controls
  - about 81, 122
  - samples 52
- ODBC 108
- OLE
  - about 122
  - ActiveX controls 123
  - automation 122, 125

- columns in DataWindows 122
  - compound documents 123
  - control 81, 122
  - DataWindow objects 124
  - DataWindow presentation style 122
  - new features 26
  - objects 122
  - structured storage 126
  - OLE custom controls 81, 122
  - OLE GenReg utility 51
  - online books 148
  - Open event for an application 138
  - organizing objects 96
  - Oval controls 79
- P**
- packaging an application
    - compile options for 144
    - what goes in the executable version 145, 160
  - painters
    - previewing objects in 158
    - storing work from 96
  - PBD files, delivering 160
  - PBFNT60.INI 19
  - .pbfmt60.ini 19
  - PBL files
    - about 96
    - cross-platform use of 97
  - PBUs, change in 20
  - Pcode
    - for 16-bit deployment 47
    - for an executable application 144
    - in a PBL (library) file 97
  - performance analysis 30
  - persistent state
    - DataWindows 5
    - shared objects 2
  - PFC *see* PowerBuilder Foundation Class Library
  - Picture controls 79
  - PictureButton controls 79
  - PictureListBox controls 79
  - piping data between data sources 135, 150
  - platforms
    - choosing for an application 139
    - portability of libraries across 97
  - plug-ins
    - DataWindow 66
    - PowerBuilder window 65
  - polymorphism 94, 100
  - PopulateError function 50
  - popup menus 82
  - popup windows 76
  - portability, libraries 97
  - POST, and asynchronous calls 4
  - PowerBuilder automation server component 68
  - PowerBuilder Cross Reference utility 156
  - PowerBuilder Extended Attribute Reporter 150
  - PowerBuilder Foundation Class Library
    - about 103
    - new features 38
  - PowerBuilder Library for Lotus Notes 103
  - PowerBuilder Object Searcher 156
  - PowerBuilder units, change in 20
  - PowerBuilder window ActiveX 8, 63
    - context feature 15, 138
    - secure mode 13
  - PowerBuilder window plug-in 65
    - context feature 15
    - new features 7
  - PowerBuilder window plug-in, context feature 138
  - PowerDesigner 149
  - PowerScript 92
  - Powersoft database interfaces 108
  - Powersoft Font Preferences 19
  - Powersoft online books 148
  - previewing
    - objects 158
    - reports 131
  - print preview
    - about 131
    - scrollbars 35
  - printing
    - about 131
    - DataWindow 134
    - RTE control 134
  - processing logic
    - coding in scripts 89
    - coding in user-defined functions 90
  - Profiler 159
  - profiling 30

- programs
  - configuring 160
  - installing for users 160
- Project painter, new features 18, 47
- PSChart 53
- PSR files 133

## R

- RadioButton controls 79
- Rectangle controls 79
- registry 144, 160
- regular mode, executing in 158
- remote procedure calls 128
- reports 131
- resources
  - delivering 160
  - in an executable file 145
- response windows 75
- RichTextEdit controls 79
- right-to-left operating systems, support for 23
- RoundRectangle controls 79
- RowFocusChanging event 36
- RPC 128

## S

- SaveAs function, new features 37
- SaveAsAscii function 37
- SCC API 29
- scripts
  - about 89
  - ingredients of 92
- scrollbars, print preview support 35
- SDI 83
- secure mode
  - plug-ins 13
  - PowerBuilder window ActiveX 9, 13
- security, defining for databases 149
- server databases 106
- server push 3
- shared objects 2
- SignalError function 50
- SingleLineEdit controls, about 79

- single-tier applications 57, 129
- Solaris, version supported 19
- source control
  - and Application objects 42
  - new API 29
- SQL Anywhere 149
- SQL Central utility 149
- SQL statements
  - about 109
  - embedding in scripts 92
  - sending to the DBMS for execution 150
- standard class user objects 101
- state information, DataWindows 5
- StaticText controls 79
- storages 126
- Stored Procedure Update for DataWindows utility 153
- streams 126
- structured storage 126
- submenus 81
- subroutines
  - implementing with user events 90
  - implementing with user-defined functions 90
- synchronization
  - application files 48
  - DataWindows 5
- Synchronizer 48
- synchronous calls, and server push 3
- system error processing for an application 137
- SystemError event for an application 138, 159

## T

- Tab controls
  - about 80
  - control property array 43
- Table painter 149
- tables
  - defining 149
  - manipulating data in PowerBuilder 150
  - migrating within or between databases 135, 150
  - overview of accessing 106
- testing an application
  - tracing and profiling 30
  - tracing database execution 158
- timers, nonvisual 43



- timestamp, last-compiled 41
- Timing object 43
- toolbars
  - defining for MDI frames 82
  - new look 41
- tracing 30
- Translation Toolkit 24, 155
- TreeView controls 79
- two-tier applications 58, 129
  - displaying controls in 78
  - displaying menu bar for 81
  - displaying popup menus in 82
  - role of 74
  - testing one in isolation 158
  - types of 74

## U

- Unicode Standard, support for 21
- user events 90
- user interface design
  - basics in PowerBuilder 74
  - SDI versus MDI 83
- user objects
  - using to access external controls 81
  - using to define your own controls 80

## V

- variable declarations 92
- variables, working with while debugging 159
- version control
  - and Application objects 42
  - new API 29
- views
  - defining in Database painter 149
  - in Debugger 32
- visual classes 100
- VScrollBar controls 79

## W

- Web, jumping to from PowerBuilder 17
- Web.PB
  - about 62
  - new features 7
  - wizard on PowerBar 51
- windows
  - about 74