Customizing PowerBuilder

A

In PowerBuilder you have the ability to customize the development environment. This is useful because it can minimize time spent on performing repetitive tasks. You also can speed up your development by creating additional toolbar buttons to give you quick access to commonly used features and objects. Most of these settings can be made from the different PowerBuilder painters. PowerBuilder stores the majority of these customizations in the PB.INI file. This appendix discusses what is contained in the PB.INI file and how to customize the PowerBuilder environment.

The PB.INI File

PowerBuilder's initialization file (PB.INI) contains a number of different sections that correspond to different PowerBuilder painters. Each section has a number of keywords defined for it that store the individual settings. The major sections are as follows:

- Application—Changes all application preferences.
- Window—Changes options used in the Window painter.
- Menu—Changes options used in the Menu painter.
- DataWindow—Changes options used in the DataWindow painter.
- Database—Changes options used in the Database painter.
- Library—Changes options used in the Library painter.
- Debug—Changes options used during debug.
- PowerBuilder—Changes systemwide PowerBuilder options.

There are also smaller sections for the SQL painter, database profiles, the Object Browser, and toolbar settings. While you can edit the PB.INI file directly, most settings contained in here are automatically saved as you make changes in PowerBuilder. To edit the PB.INI, use your favorite text editor or press Shift+F6 from within PowerBuilder to open the File Editor. This appendix looks at many of the keywords for each of the different INI sections.

NOTE

In previous versions of PowerBuilder, many of the settings could be maintained using the Preferences painter. In Version 5.0, the Preferences painter has been eliminated and the places where you modify settings are now located in the different object painters.

Application Section

The application section affects the application object's setup. The keywords for the application section are listed in Table A.1.

Table A.1. Application keywords.

Variable	*Description*
`DefLib`	Default PBL to store objects unless a different one is specified during a save
`AppLib`	The fully qualified PBL containing the current application
`AppName`	The current application name

NOTE

For all previously accessed applications, the library search path for that application is displayed under the `AppName` keyword. Each entry is prefaced with `$c`, followed by the library containing the application object. Each library that is included in that application's library search path is listed with each entry separated by a semicolon (;).

The Window Section

The keywords listed in the window section indicate the defaults used in the Window painter. These variables control the default prefixes for window control names and how a grid is placed. The grid keywords can also be set via the Window painter, but the object names can only be changed in the INI file. These keywords are presented in Table A.2.

Table A.2. Window keywords.

Variable	*Description*
`CheckBox`	The `CheckBox` default prefix (`cbx_`).
`CommandButton`	The `CommandButton` default prefix (`cb_`).
`DataWindow`	The `DataWindow` default prefix (`dw_`).
`DropDownListBox`	The `DropDownListBox` default prefix (`ddlb_`).
`EditMask`	The `EditMask` default prefix (`em_`).
`Graph`	The `Graph` default prefix (`gr_`).

continues

Table A.2. continued

Variable	*Description*
`GroupBox`	The `GroupBox` default prefix (`gb_`).
`HScrollBar`	The `HScrollBar` default prefix (`hsb_`).
`Line`	The `Line` default prefix (`ln_`).
`ListBox`	The `ListBox` default prefix (`lb_`).
`MultiLineEdit`	The `MultiLineEdit` default prefix (`mle_`).
`OLEControl`	The `OLEControl` default prefix (`ole_`).
`Oval`	The `Oval` default prefix (`oval_`).
`Picture`	The `Picture` default prefix (`p_`).
`PictureButton`	The `PictureButton` default prefix (`pb_`).
`RadioButton`	The `RadioButton` default prefix (`rb_`).
`Rectangle`	The `Rectangle` default prefix (`r_`).
`RoundRectangle`	The `RoundRectangle` default prefix (`rr_`).
`SingleLineEdit`	The `SingleLineEdit` default prefix (`sle_`).
`StaticText`	The `StaticText` default prefix (`st_`).
`UserObject`	The `UserObject` default prefix (`uo_`).
`VScrollBar`	The `VScrollBar` default prefix (`vsb_`).
`Default3D`	Control borders default to 3D and the window's background color. The values are listed below.
`GridOn`	The "snap to grid" option. `0` means off; `1` means on.
`GridShow`	Whether to show the grid. `0` means hide the grid; `1` means display it.
`GridX`	The width of the grid, in pixels, if displayed. Default is `8`.
`GridY`	The height of the grid, in pixels, if displayed. Default is `8`.
`Status`	This is an old PowerBuilder 3.0 option that controlled the display of the status box.

The values for the `Default3D` variable are listed in Table A.3.

Table A.3. Values for the `Default3D` keyword.

Value	*Meaning*
`0`	The default window background is white and controls are not set to 3D.
`1`	The default window background is gray and controls are set to 3D.

The `Status` variable enabled you to have a Select Object Status window displayed in the lower-right corner during window development. This option, when turned on, displays information about the object selected. PowerBuilder has replaced the window with the information displayed in the MicroHelp area at the bottom of the Window painter when an object is selected, but the `Status` entry in the INI file has remained.

The Menu Section

Only one keyword is found in the `Menu` section: the `Prefix` variable. This keyword value is the prefix for menu item names (`m_`) when they are created in the Menu painter. Like control prefix names, the prefix for a menu item can be 1 to 16 characters in length.

The DataWindow Section

The keywords found in the DataWindow section are used as the defaults in the DataWindow painter. These variables are listed in Table A.4.

Table A.4. DataWindow keywords.

Variable	*Description*
`GridOn`	The "snap to grid" option. `0` means off; `1` means on.
`GridShow`	Option for whether to show the grid. `0` means hide the grid; `1` means display it.
`GridX`	The width of the grid, in pixels, if displayed. Default is `8`.
`GridY`	The height of the grid, in pixels, if displayed. Default is `8`.
`new_default_datasource`	The default data source for a new DataWindow.
`new_default_presentation`	Indicates the default presentation style.
`new_form_color`	The default background color for a freeform DataWindow.
`new_form_column_border`	The default column border for a freeform DataWindow.
`new_form_column_color`	The default column color for a freeform DataWindow.
`new_form_text_border`	The default text border for a freeform DataWindow.
`new_form_text_color`	The default text color for a freeform DataWindow.
`new_grid_color`	The default background color for a grid DataWindow.
`new_grid_column_border`	The default column border for a grid DataWindow.
`new_grid_column_color`	The default column color for a grid DataWindow.
`new_grid_text_border`	The default text border for a grid DataWindow.

continues

Table A.4. continued

Variable	*Description*
`new_grid_text_color`	The default text color for a grid DataWindow.
`new_label_color`	The default background color for a label DataWindow.
`new_label_column_border`	The default column border for a label DataWindow.
`new_label_column_color`	The default column color for a label DataWindow.
`new_label_text_border`	The default text border for a label DataWindow.
`new_label_text_color`	The default text color for a label DataWindow.
`new_tabular_color`	The default background color for a tabular DataWindow.
`new_tabular_column_border`	The default column border for a tabular DataWindow.
`new_tabular_column_color`	The default column color for a tabular DataWindow.
`new_tabular_text_border`	The default text border for a tabular DataWindow.
`new_tabular_text_color`	The default text color for a tabular DataWindow.
`Outline_Objects`	Whether DataWindow objects are outlined. `1` means yes; `0` means no.
`PreviewOnNew`	Preview a new DataWindow on data source selection. Values are `yes` and `no`.
`PreviewRetrieve`	Retrieve each time you preview a window.
`Preview_RetainData`	Save retrieval arguments between previews.
`PrintOnNew`	Preview a new DataWindow after creating a new DataWindow definition. Values are `yes` and `no`.
`PrintPreviewRulers`	Show rulers on Print Preview window. Values are `yes` and `no`.
`PrintPreviewZoom`	Zoom percentage for the Print Preview window.
`Ruler`	Show the ruler in the DataWindow painter. Values are `yes` or `no`.
`Status`	This is an old PowerBuilder 3.0 option that controlled the display of the status box.
`stored_procedure_build`	How the result set is built. `0` means manual result set; `1` means PowerBuilder will generate the result set.
`DefaultFileOrLib`	Indicates which files can be opened in the Report painter.

Many of the keywords in Table A.4 have several values that can be specified. The `new_default_datasource` keyword contains the value of the last data source selected when you

created a new DataWindow (for example, SQL Select or stored procedure data source). The values are listed in Table A.5.

Table A.5. Values of the `new_default_datasource` keyword.

Value	*Meaning*
1	SQL Select
2	Query
3	Stored Procedure
4	Script
5	Quick Select

The `new_default_presentation` keyword contains the value of the last presentation style selected when you created a new DataWindow (for example, tabular or freeform). The values are listed in Table A.6.

Table A.6. Values for the `new_default_presentation` keyword.

Value	*Meaning*
1	Tabular
2	Freeform
3	Grid
4	Label
5	N-Up
6	Crosstab
7	Graph
8	Group

The `PreviewRetrieve` keyword is used to determine whether a retrieve is performed on a preview. This also affects when data is retrieved in the Database painter if the Data Manipulation option is selected. A value of `1` indicates that data should be retrieved immediately when you preview a DataWindow. A value of `0` forces you to click the Retrieve icon or select Retrieve from the Rows menu in order to view data.

The `Preview_RetainData` keyword is used to determine whether PowerBuilder caches retrieved data on a DataWindow preview. A value of `1` indicates that the retrieved data is cached between previews. This is useful because database access is not required every time you enter Preview, which therefore saves you time and resources. In addition, you will not be prompted for retrieval

arguments after the first preview. A value of `0` means that the data will be retrieved from the database again. Also, you will be prompted for retrieval arguments each time you select Preview.

NOTE

When you change the SQL for the DataWindow, data is automatically retrieved on Preview. The value of the `Preview_RetainData` variable has no effect in this case.

The `DefaultFileOrLib` variable indicates which files can be opened in the Report painter. The values are listed in Table A.7.

Table A.7. Values for the `DefaultFileOrLib` variable.

Value	*Meaning*
`0`	DataWindow objects (reports) in PBLs only
`1`	Powersoft-created reports in PSR files only
`2`	Objects in PBLs and PSR files

The Database Section

The keywords found in the Database section are used as the defaults for database processing during development. The keywords used are dependent on the DBMS you are using. The variables are listed in Table A.8.

Table A.8. Database preferences.

Variable	*Description*
`Vendors`	The name of your DBMS vendor (ODBC, SYB, SQL Server v4.*x*, and so on).
`DBMS`	The current database management system vendor (ODBC).
`ServerName`	The name of the connected server (Falcon).
`Database`	The name of the current database (Powersoft Demo DB).
`UserID`	Your user ID (DBA).
`DatabasePassword`	Your database password.
`LogId`	Your server logon ID.
`LogPassword`	Your server password.
`DbParm`	Database-specific parameters.

Variable	*Description*
`AutoCommit`	Controls how transactions are processed. Values are `TRUE` or `FALSE`.
`AutoQuote`	How quotes are placed around `Where Criteria` expressions.
`Columns`	The number of columns displayed when you open a table before a vertical scroll bar appears.
`ForeignKeyLineColor`	Displays color of lines between foreign key symbol and table.
`IndexKeyLineColor`	Displays color of lines between index symbol and table.
`PrimaryKeyLineColor`	Displays color of lines between primary key symbol and table.
`Prompt`	Prompts for database connect information. `1` equals yes; `0` equals no.
`Lock`	The database isolation level (database-dependent).
`NoCatalog`	Indicates catalog access to PowerBuilder data repository tables.
`ReadOnly`	Used to limit database access to the data repository.
`ShowIndexKeys`	Shows index keys. The values are `0` for no and `1` for yes.
`ShowReflnt`	Shows referential integrity in the Database painter.
`StayConnected`	Indicates when PowerBuilder disconnects from the database.
`TableDir`	Shows the table directory when you enter the Database painter.
`TableListCache`	Contains a time (in seconds) until the table list is refreshed.
`TableSpace`	Specifies the name of the table space (database-dependent).
`TerminatorCharacter`	A character that indicates the end of SQL statements (;).
`HideComments`	Shows column comments when table is opened. `1` for Hide; `0` for Show.
`TableHeaderColor`	The background color of the table header.
`TableHeaderTextColor`	The display color of the text in the table header.
`TableDetailColor`	The background color of the detail area.
`TableDetailTextColor`	The display color of the detail area text.
`TableColumnNameTextColor`	The display color of the detail area column names.

The AutoQuote keyword is used to determine whether single quotation marks are automatically placed around strings found in the Expression box of the Where Criteria window in the Query or Select painter. If the variable is set to 1, quotes are automatically added to strings in the Expression box. If the variable is set to 0, no quotes are added.

The NoCatalog keyword affects usage of the data repository. If the PowerBuilder repository tables do not exist and this variable is set to No, PowerBuilder automatically creates the tables the first time a user connects to the database. If this variable is set to Yes, PowerBuilder will not automatically create the tables. PowerBuilder will not reference the PowerBuilder repository tables with a value of Yes, but it will permit DDL and DML (CREATE, INSERT, or DELETE) statements.

The ReadOnly keyword is used to limit database access. If the PowerBuilder repository tables do not exist and this variable is set to 0, PowerBuilder automatically creates the tables the first time a user connects to the database. With a value of 1, PowerBuilder does not care if the repository tables already exist and just uses the default values for columns. Users cannot modify information in the data repository.

The ShowReflnt variable indicates whether referential integrity is displayed in the Database painter. Referential integrity is displayed in the form of foreign keys, primary keys, and the lines associated with them. If the variable is set to 1, referential integrity is displayed. If it is 0, referential integrity is not displayed.

The StayConnected variable indicates when PowerBuilder disconnects from the connected database. If the variable is set to 0, PowerBuilder will disconnect from the database when you exit the Database painter. You will have to reconnect each time you enter the painter, which may be a long process. If the variable is set to 1, PowerBuilder will remain connected to the database until you exit PowerBuilder.

The TableDir variable indicates whether the table directory should be displayed when you enter the Database painter. If this variable is set to 1, PowerBuilder automatically opens the Select Table dialog box when you open the Database painter. If this variable is set to 0, the list is not displayed until you click on the Tables icon.

The Library Section

The keywords found in the Library section are used as the defaults for the Library painter. These variables are listed in Table A.9.

Table A.9. Library preferences.

Variable	*Description*
`ApplicationExplosion`	Include application explosion in report. `1` means Yes; `0` means No.
`ApplicationScripts`	Include application scripts in report. `1` means Yes; `0` means No.
`CondensedFont`	Font used for control's text in the window picture on an entity report.
`NormalFont`	Contains the font used when printing developer reports.
`DeletePrompt`	Prompt for library or entry deletion. `1` means Yes; `0` means No.
`DisplayComments`	Show comments in the library list. `1` means Yes; `0` means No.
`DisplayDates`	Show last update date in library list. `1` means Yes; `0` means No.
`DisplaySizes`	Show entry file sizes in the library list. `1` means Yes; `0` means No.
`IncludeApplications`	Include application objects in a browse. `1` means Yes; `0` means No.
`IncludeDataWindows`	Include DataWindows in a browse. `1` means Yes; `0` means No.
`IncludeFunctions`	Include user functions in a browse. `1` means Yes; `0` means No.
`IncludeMenus`	Include menus in a browse. `1` means Yes; `0` means No.
`IncludePipeLines`	Include pipelines in a browse. `1` means Yes; `0` means No.
`IncludeQueries`	Include queries in a browse. `1` means Yes; `0` means No.
`IncludeStructures`	Include structures in a browse. `1` means Yes; `0` means No.
`IncludeUserObjects`	Include user objects in a browse. `1` means Yes; `0` means No.
`IncludeWindows`	Include windows in a browse. `1` means Yes; `0` means No.
`MenuAttributes`	Include menu attributes in report. `1` means Yes; `0` means No.
`MenuScripts`	Include menu explosion in report. `1` means Yes; `0` means No.

continues

Table A.9. continued

Variable	*Description*
`SaveBackupsOnOptimize`	Create a backup file when library is optimized. `1` means Yes; `0` means No.
`SourceVendor`	Contains ID of source control vendor being used (such as PVCS).
`UserID`	Contains the Check In/Out user ID.
`WindowAttributes`	Include window attributes in report. `1` means Yes; `0` means No.
`WindowObjects`	Include window objects during report print. `1` means Yes; `0` means No.
`WindowObjectsAttributes`	Include window object attributes in report. `1` means Yes; `0` means No.
`WindowObjectsScripts`	Include window object scripts in report. `1` means Yes; `0` means No.
`WindowPicture`	Include picture of window in report. `1` means Yes; `0` means No.
`WindowScripts`	Include window scripts in report. `1` means Yes; `0` means No.

The majority of the keywords have to do with three areas: what information is printed on any entry report, what objects are included for viewing in the Library painter, and what information is displayed for each entry.

The Debug Section

The keywords found in the Debug section are used when you are debugging an application. The keywords are listed in Table A.10.

Table A.10. Debug keywords.

Variable	*Description*
`VariablesWindow`	Show Variables window during debug. `1` means Show; `0` means Hide.
`WatchWindow`	Show Watch window during debug. `1` means Show; `0` means Hide.
`Stopn`	Contains a stop point that was set in debug.
`Watchn`	Contains any watch variables set.

The `n` in the `Stopn` keyword is the number of the stop. For example, if you have seven stops, you have `stop1` through `stop7` preference variables. The format of the information stored in the `Stopn` keyword is listed in Table A.11.

Table A.11. Values for the `Stopn` keyword.

Option	*Description*
`State`	Indicates the state of the stop (enabled or disabled)
`ObjectName`	Contains the name of the object
`ControlName`	Contains the name of the control
`EventName`	Contains the name of the event
`LineNo`	Contains the line number in the script where the stop is set

For example, an enabled stop for window `w_product_test` at line 4 in the `Clicked` event of the `cb_go` command button would look like this:

```
e,w_product_test,cb_go,clicked,4
```

The same process applies to the `Watchn` keyword, where `n` indicates the order of the variables specified in the Watch window in the debugger.

The PowerBuilder Section

The keywords found in the PowerBuilder section (the section is actually titled `[PB]`) are systemwide keywords that are not tied to one painter. The variables are listed in Table A.12.

Table A.12. PowerBuilder system keywords.

Variable	*Description*
`CompilerWarnings`	Show compiler warnings during compile. `1` means Show; `0` means Suppress.
`DashesInIdentifiers`	Dashes are allowed in identifiers. `1` means Allow; `0` means Prohibit.
`DatabaseWarnings`	Show database warnings during compile. `1` means Show; `0` means Suppress.
`FontBold`	Indicates whether text should default to bold. `1` means Yes; `0` means No.
`FontFixed`	Use a fixed font. `1` means Fixed; `0` means Variable fonts.
`FontHeight`	Contains the font height in points.

continues

Table A.12. continued

Variable	*Description*
`FontName`	Contains the default font name.
`Maximized`	Maximizes main PowerBuilder window when opened. `1` means Yes; `0` means No.
`PromptOnExit`	Exit confirmation message to leave PowerBuilder. `1` means Yes; `0` means No.
`SharedIni`	Contains the name of the shared INI file, if any.
`StripComments`	Strip statement comments from DBA Notepad SQL statements before execution. `1` means Strip; `0` means Keep.
`ToolbarFontHeight`	Contains height of toolbar text.
`ToolbarFontName`	Contains font name of toolbar text.
`Window`	Contains display size and position for PowerBuilder windows.
`EditorTabWidth`	Contains the tab width used in all scripts.
`EditorFontHeight`	Contains the PowerScript font height being used.
`EditorFontName`	Contains the name of the PowerScript font being used.
`EditorFontBold`	Indicates whether PowerScript font is bold. `1` means Yes; `0` means No.
`EditorFontFixed`	Converts variable to fixed font if not found. `1` means Yes; `0` means No.
`EditorFontItalic`	Indicates whether PowerScript font is italicized. `1` means Yes; `0` means No.
`EditorFontStrikeOut`	Indicates whether PowerScript font has strike-outs. `1` means Yes; `0` means No.
`EditorFontUnderline`	Indicates whether PowerScript font is underlined. `1` means Yes; `0` means No.
`EditorColor`	Indicates the background color of the PowerScript Painter.
`EditorColorn`	Indicates the color of the type of script statement indicated by `n`.
`EditorDropDowns`*x*	Indicates which drop-down list boxes are displayed in the PowerScript painter where *x* indicates the object type: App is application, Win is window, GFun is Global functions, LFun is local functions and Menu.
`EditorIndenting`	Indicates whether auto-indentation is used in the PowerScript painter. `1` means Yes; `0` means No.
`EditorTabSize`	Indicates how many spaces are used when the Tab key is pressed and for auto-indentation.

Variable	*Description*
`EditorColoring`	Indicates whether the PowerScript painter uses color-coding. `1` means Yes; `0` means No.
`FindCase`	Indicates case sensitivity on the Find window in the PowerScript painter. `1` means Yes; `0` means No.
`FindRegExp`	Indicates whether a regular expression is used on the Find window in the PowerScript painter. `1` means Yes; `0` means No.
`FindWrap`	Indicates a wrap at the beginning or end in the Find window in the PowerScript painter. `1` means Yes; `0` means No.
`Object(n)`	Contains last four objects referenced. Listed on the File menu.

There are a number of new entries in the PowerBuilder section that correspond to the new enhancements in the PowerScript painter. The values for `EditColorn` are

Value	*PowerScript Statement Type*
0	Whitespace
1	PowerScript keyword
2	PowerScript data type
3	Integer literal
4	Float literal
5	Date literal
6	Time literal
7	String literal
8	Symbol
9	Invalid text
10	Identifier
11	Jump label
12	Comment
13	Invalid string
14	Enumeration

Miscellaneous Sections

There are several additional sections in the PB.INI file such as the SQL painter, Object Browser, DBMS profiles, and toolbars. The SQL painter section stores a few settings such as which table information (comments, labels, and data types) are displayed in the painter. The Object Browser

section stores the Browser's position and size, as well as a number of Boolean flags indicating whether the hierarchy, legend, and inheritance are displayed. DBMS profiles keep track of all databases that have been connected to PowerBuilder. Last of all, there are numerous sections that keep track of the toolbar configurations for customization of the available PowerBuilder toolbars.

PowerBuilder Toolbars

PowerBuilder has a number of different toolbars available to the developer. In Version 5.0, these toolbars can be extended and new toolbars can be created to fit the developer's needs. PowerBuilder supplies several default toolbars (see Figure A.1):

- PowerBar—Used to open painters and other tools.
- StyleBar—Used to change attributes of text.
- PainterBar—Used to modify components of the current painter. For example, in the Script painter, buttons include Cut, Paste, Copy, Delete, Comment, Uncomment, Undo, Select All, SQL Paste, Paste Statements, Browse Objects, and Return.

FIGURE A.1.
Default toolbars.

Manipulating Toolbars

To manipulate any of the toolbars, right-click on any of the toolbars and choose one of the options displayed (see Figure A.2). You can change a toolbar's location, whether text is shown under each icon for all toolbars, and whether the toolbars are displayed. A toolbar's location can also be changed by clicking on the toolbar and dragging it to a new location.

FIGURE A.2.
The toolbar pop-up menu.

New to PowerBuilder 5.0 is the ability to dock toolbars, which means you can take an existing toolbar and drop it in an existing toolbar. To see an example, Figure A.3 shows the toolbars before being docked, and Figure A.4 shows the docked toolbars.

FIGURE A.3.
Toolbars in their initial state.

FIGURE A.4.
Docked toolbars.

Another way to access the toolbars is to select the Toolbars option from the Window menu (or File menu when no painters are open). When you choose this option, the Toolbars dialog box is displayed (see Figure A.5).

FIGURE A.5.
The Toolbars dialog box.

From this dialog, you can specify if toolbar text or PowerTips are displayed for all shown toolbars, the size of the font when the Show Text option is on, which toolbars are visible, where each toolbar is located, and whether to customize a toolbar.

Customizing a Toolbar

To change a toolbar, click the Customize button in the Toolbars dialog box or select the Customize option from the toolbar pop-up menu. The Customize dialog box is opened (see Figure A.6).

The Customize dialog box allows you to add icons to any toolbar using predefined buttons or creating your own custom buttons. You can also change the buttons' order, add spacing, or remove buttons.

To modify your toolbar (select either the PainterBar or PowerBar radio button), all you need to do is drag and drop. The Current toolbar palette displays the toolbar with its existing buttons. To change the order, click on a button and drag it to its new location. The Selected Palette list box displays a number of buttons with predefined commands. To find out what each button

is, click on the button to display a description at the bottom of the dialog box. Once the desired button has been located, click it and drag it to the current toolbar palette. Using the buttons on the right hand side of the dialog, you can reset the current toolbar to its defaults or clear the toolbar of all buttons.

FIGURE A.6.
The Customize dialog box.

You can also create your own buttons to perform specific actions. Click on the Custom radio button, which displays a multitude of buttons that you can add to your toolbar. To create a new button, click on the custom button you want to add and drag it onto the Current toolbar. When the button is dropped, the Toolbar Item Command dialog box is opened (see Figure A.7).

FIGURE A.7.
The Toolbar Item Command dialog box.

The Toolbar Item Command dialog box enables you to specify what the new custom button will do. Depending on which toolbar you are changing or which painter you are in, a custom button can be added to do the following:

- Execute a PowerBuilder menu item.
- Run an executable outside of PowerBuilder.
- Run a query.
- Run a report.

- Select a user object for placement in a window or a custom user object (available in the Window and User Object painters only).
- Assign a display format to a column in a DataWindow object (available in the DataWindow and Report painters only).
- Create a computed field in a DataWindow object (available in the DataWindow and Report painters only).

The following sections outline how to accomplish each of these commands.

Executing a PowerBuilder Menu Item

To execute a PowerBuilder menu item, use the following syntax in the Command Line box:

```
@MenuBarItem.MenuItem
```

For example, to create a Compile button to be used in the Script painter, the command line would be

```
@Compile.Script
```

You can also refer to menu items by number. This is necessary when a menu item does not have a text caption but is displayed in the form of a picture. When using numbers for menu items, you must remember that each line separator in the menu also counts as a menu item.

An example of this is the Window Functions option available under the Declare menu in the Window painter. The command line to create a custom button to open the Window Function painter would look like this:

```
@Declare.5
```

This option is the fifth menu item. Here is the list of menu items:

- Global variables
- Shared variables
- Instance variables
- Line separator
- Window functions

Running an Executable

To run an executable outside of PowerBuilder, type the fully qualified name of the executable in the Command Line box (if it is not in the path). You can also click the Browse button, which will enable you to search for the executable via the Browse dialog box.

For example, to add a button to run the Windows calculator, type the following:

```
C:\WINDOWS\CALC.EXE
```

Running a Query

To run a predefined query object, click the Query button. The Select Query dialog box will appear (see Figure A.8). Select the query object you want to assign to this button and click OK.

FIGURE A.8.
The Select Query dialog box.

For example, if you want to know what the maximum error-message number is from an error-message table so that you can create the next error message, you can create a query for this purpose. To execute the query, write a command line like this:

```
Query /l c:\projects\cis\contacts.pbl /o q_max_error_number /r
```

The options for this syntax are as follows:

- `Query`—Indicates to PowerBuilder that this is a query.
- `/l`—Indicates that the next field is the library name of the query. The library does not have to be in your application's search path.
- `/o`—Indicates that the next field is the object name of the query.
- `/r`—Tells PowerBuilder to run the query.

Running a Report

To run a report (a non-editable DataWindow), click the Report button to open the Select Report dialog box (see Figure A.9) and then select the report you want to assign to this button.

FIGURE A.9.
The Select Report dialog box.

For example, if you want to display a list of all error messages so that you can utilize an already created error message, you can create a report for this purpose. To execute the report, write a command line like this:

```
Report /l c:\projects\cis\sales.pbl /o d_error_message_report /ro
```

The options for this syntax are as follows:

- `Report`—Indicates to PowerBuilder that this is a report.
- `/l`—Indicates that the next field is the library name of the report.
- `/o`—Indicates that the next field is the object name of the report.
- `/r`—Tells PowerBuilder to run the report.
- `/ro`—Tells PowerBuilder to run the report but not to provide a design mode to change the report.
- `/a "arguments"`—Specifies the arguments passed to the report.

Placing a User Object

To create a custom button that selects a user object for placement in a window or a custom user object, click the User Object button (only available in the Window and User Object painters). The Select User Object dialog box will appear (see Figure A.10). Select the user object you want to assign to this button and click OK.

For example, if you are always placing the same user object on your windows, such as a DataWindow control, you might want a custom button for this purpose. This can save considerable time if you are using a framework because you can replace the standard window controls with framework controls. The following command line will accomplish this:

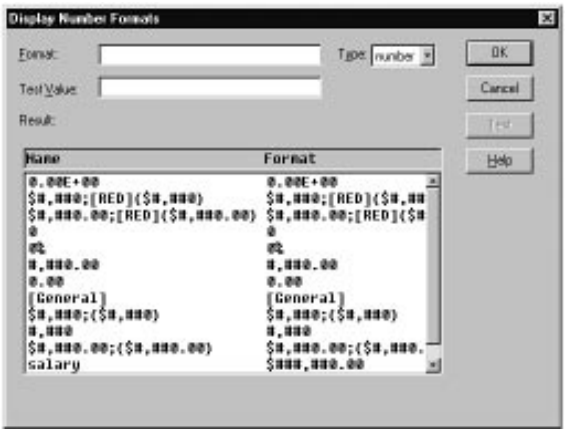
```
UserObject u_dw
```

FIGURE A.10.
The Select User Object dialog box.

Assigning Display Formats and Computed Fields

The Toolbar Item Command dialog box for a DataWindow contains two extra command buttons: the Format button and the Function button. To assign a display format to a column in a DataWindow object, click the Format button. The Display Number Formats dialog box appears (see Figure A.11). Select the format you want to assign to this custom button and click OK.

FIGURE A.11.
The Display Number Formats dialog box.



If you always use the phone format on strings, you can create a phone format custom button. The command line to do this is the following:

```
Format s,(@@@) @@@-@@@@
```

The `s` indicates to PowerBuilder that this is a string format.

To create a computed field in a DataWindow object, click the Function button. The Function for Toolbar dialog box appears (see Figure A.12). Select the computed function that you want to assign to this custom button and click OK.

FIGURE A.12.
The Function for Toolbar dialog box.

For example, if you always create a computed column using the `Max()` function, you would select `Max()` from the dialog. The command line to do this would look like the following:

```
Function max
```

Creating Additional Toolbars

New to PowerBuilder 5.0 is the ability to create additional custom PainterBars and PowerBars. These customized toolbars can be used instead of or in addition to the existing PainterBars and PowerBars. To create a new toolbar, select the New menu item on the toolbar pop-up menu (refer to Figure A.2). This opens the New Toolbar dialog box (see Figure A.13).

FIGURE A.13.
The New Toolbar dialog box.

In this dialog box, select the toolbar you are interested in and click OK. This opens the Customize dialog box shown in Figure A.6. The dialog box behaves exactly the same as when you customize an existing toolbar. In this dialog box, drag any existing toolbar buttons you want to include as well as any custom toolbars you wish to create from the Selected Palette list box to the Current Toolbar list box. Once you have specified all the icons you desire onto the new toolbar, click OK, which displays the new toolbar. This toolbar (PowerBar or PainterBar) will display every time you run PowerBuilder.

A useful example is creating a new PainterBar in the Window painter. Rather than just have a toolbar that includes the standard PowerBuilder controls, create a new PainterBar that maps to your standard visual user objects. This way, you will always have quick access to your user objects instead of going through the manual process of selecting a user object each time.

To remove one of the created toolbars, select Customize from the toolbar pop-up menu to open the Customize dialog box (refer to Figure A.6). Drag all the toolbar icons out of the current toolbar list box back to the Selected Palette list box. Once all the icons have been removed, click OK, which removes the toolbar and makes it available in the New Toolbar dialog box again (refer to Figure A.13).

Summary

PowerBuilder development can be customized to minimize the amount of time spent doing menial tasks. You can do this by setting preferences in many of the painters or editing the PB.INI file. You can also greatly customize your toolbars to perform frequently completed tasks.