CodeWarrior™ Development Tools

PowerPlant Constructor™ User Guide

Revised: 8/12/03

Metrowerks reserves the right to make changes to any product described or referred to in this document without further notice. Metrowerks makes no warranty, representation or guarantee regarding the merchantability or fitness of its products for any particular purpose, nor does Metrowerks assume any liability arising out of the application or use of any product described herein and specifically disclaims any and all liability. **Metrowerks software is not authorized for and has not been designed, tested, manufactured, or intended for use in developing applications where the failure, malfunction, or any inaccuracy of the application carries a risk of death, serious bodily injury, or damage to tangible property, including, but not limited to, use in factory control systems, medical devices or facilities, nuclear facilities, aircraft navigation or communication, emergency systems, or other applications with a similar degree of potential hazard**.

How to Contact Metrowerks

Corporate Headquarters	Metrowerks Corporation 7700 West Parmer Lane Austin, TX 78729 U.S.A.
World Wide Web	`http://www.metrowerks.com`
Sales	Voice: 800-377-5416 Fax: 512-996-4910 Email: sales@metrowerks.com
Technical Support	Voice: 800-377-5416 Voice: 512-996-5300 Email: support@metrowerks.com

Table of Contents

6 Editing Bitmaps 95

1

Introduction

Welcome to the *PowerPlant Constructor User Guide*. Constructor is the visual interface builder for PowerPlant. PowerPlant is Metrowerks' Macintosh application framework. Using PowerPlant and the CodeWarrior IDE®, you create Mac OS applications that build on the PowerPlant framework.

Constructor lets you build a visual interface for a PowerPlant program. This manual is the user guide for learning to use Constructor.

NOTE To use Constructor, you need to use the Mac OS. There is not a Windows-based version of Constructor.

The sections in this introduction are:

- Read the Release Notes—where to find last-minute information not covered in this manual
- Welcome to Constructor—a brief introduction to Constructor
- What's New in This Release—a brief description of changes for this release
- Bugs in this Release—known limitations of this release
- System Requirements—hardware and software requirements
- Installing Constructor—putting it together so you can use it
- Starting Points—an overview of the chapters in this manual
- Where to Learn More—other sources of Constructor information

Read the Release Notes

Before using Constructor, please read the release notes. They contain important information about new features, bug fixes, and any late-breaking changes that could not make it into the manual before press deadlines.

Welcome to Constructor

Constructor is an interactive, graphical, resource editor. With Constructor you can create, modify, and save a variety of resources for windows, window content, controls, buttons, scroll bars, bitmaps, text appearance, menus, and much more.

Constructor is designed to work primarily with PowerPlant, Metrowerks' application framework for Mac OS programming. A PowerPlant-based application uses the information stored in Constructor resources to create the visual appearance of the application at runtime. Therefore, think of Constructor as a visual interface builder for Mac OS programs.

You'll find all the information you need starting with "Constructor Overview."

What's New in This Release

This release of Constructor introduces a few minor changes and new features. The biggest changes are:

Flattened resource file support in PowerPlant Constructor

Mac OS X runs on many different disk formats, including non-traditional Mac HFS formats. This means that some disk formats (UFS and NFS, for example) do not handle the resource fork of a Mac OS file. Since Constructor is a resource editor, it is designed to alter data in the resource fork of a file. Apple Computer has introduced flattened resource files which are data files whose contents are what was traditionally in the resource fork of a file. Constructor allows the editing and saving of the traditional resource files, and also the new flattened resource files.

These flattened resources contain all the resource information within the data fork of the file instead of the resource fork. There is a new checkbox in the **Save As** dialog box under the **File** menu that controls this feature.

“LGA” classes no longer covered

Most “LGA” classes, except LGADialog, LGAFocusBorder, LGAColorSwatchControl, are no longer supported by PowerPlant Constructor since their look and feel are already provided by the Mac OS Appearance Manager.

Bugs in this Release

There are many drawing and control layout-related bugs on Mac OS X. These should hopefully be cleaned up in later versions of the OS.

System Requirements

If you can run CodeWarrior, you can run Constructor.

- A Macintosh-compatible computer with a PowerPC 601 or better processor.
- Mac OS System 8.1 or later with CarbonLib 1.0.4 or later installed.
- A minimum of 32 MB RAM to use the CodeWarrior IDE along with Constructor.
- A CD-ROM drive to install CodeWarrior software, documentation, and examples

Constructor is part of a complete package that also includes the CodeWarrior IDE, and the PowerPlant application framework (source code or library files). You need all of these elements to use Constructor effectively.

Constructor has been Carbonized to work with Mac OS X, as well as Mac OS 8.1 and later as long as CarbonLib 1.0.4 or later is installed. If you don’t have CarbonLib, you can find it at:
`http://www.apple.com/support/`

Installing Constructor

Installing Constructor is very simple. You may choose one of two methods: automatic or manual.

The CodeWarrior Installer automatically installs Constructor with any installation that also includes PowerPlant. If you install PowerPlant, you also install Constructor.

To install Constructor manually, locate the Constructor file on the CodeWarrior Tools CD. It is in the `PowerPlant Constructor` folder. Copy the file to your hard drive. There are no other files required, just the application itself.

Starting Points

This manual is a user's guide to Constructor. When you have a question about the interface, or about how to accomplish a particular task, refer to the chapter that covers the topic in which you are interested.

Each chapter begins with an overview. In addition to this introduction, you can read:

- "Constructor Overview"—an overview of Constructor, including a discussion of the Constructor project window
- "Managing Resources Overview"—creating, copying, modifying, and deleting resources in a Constructor project
- "Editing Windows and Views Overview"—creating and editing a visual interface
- "Editing Menu Bars and Menus Overview"—creating and editing menus and menu bars
- "Editing Bitmaps Overview"—creating and editing bitmaps of all kinds, including icons, patterns, cursors, and pictures
- "Editing Text Traits Overview"—creating and editing text traits
- "Editing String Lists Overview"—editing STR# resources
- "Editing Custom Pane Types Overview"—editing custom pane type resources
- "Pane Properties Overview"—a reference to each pane property inspector
- "Constructor Menu Reference Overview"—a reference to each Constructor menu item

If you have never used Constructor, then you should begin with "Constructor Overview" before proceeding to other topics. This chapter covers the various elements of the Constructor interface, and how they work together.

“Managing Resources Overview” is also excellent for new Constructor users. It discusses how to create new resources.

Use the remaining chapters in this manual as reference material. They discuss how to edit resources of various types.

Where to Learn More

This manual assumes you are familiar with Mac OS programming in general, and with PowerPlant in particular.

This manual does not explain the PowerPlant application framework, or the visual elements you find in a PowerPlant application. For more information on the various visual elements discussed in this manual (such as panes, views, controls, and the individual types of panes), see *The PowerPlant Book*.

The PowerPlant Book also contains a series of PowerPlant programming tutorials. Several of these tutorials include work in Constructor, including creating custom pane type resources. These tutorials show you how to use Constructor in real programming situations. Examining the Constructor resource files in those tutorials will help you become familiar with Constructor. These tutorials run on Mac OS computers.

To learn more about Mac OS programming, consider Discover Programming for Macintosh, available from Metrowerks. It contains the complete text of several books on Mac OS programming (in electronic form).

2

Constructor Fundamentals

This chapter is an introduction to Constructor. It gives you a high-level overview of what Constructor is and how to use it effectively.

Constructor Overview

Constructor is a resource editor designed to create and edit several resource types.

Constructor was originally created as a PPob resource editor to accompany PowerPlant. So it has become the premier visual interface builder for the Mac OS.

Over time, Constructor has come to understand more resource types, most of which have no direct connection with PowerPlant. For example, Constructor can edit 'MENU' and 'MBAR' resources, and many different kinds of bitmap-based resources. These resources do not have to be inside a PowerPlant-based project.

You can use Constructor to edit these resources in any kind of file.

The topics in this overview include:

- What is a Resource?—what resources are for and Mac OS programming
- Constructor's Resource Types—resource types that Constructor understands
- Constructor Project Files—the files Constructor works with and how to manage them
- Project Window—the display for a project file
- Editor Windows—how individual resource editors relate to the project window
- Property Inspector Window—examining properties for resources, panes, menus, and menu items

- User Interface Features—a quick look at the user interface

What is a Resource?

A resource is a structured format for storing data. Any program that understands the format can then locate and use data stored in the resource.

Mac OS uses resources heavily. There are dozens if not hundreds of standard resources recognized by many applications, and creating your own resource formats for proprietary use is easy. Resources are such an integral part of Mac OS programming that it is built right into the Mac OS 8.x and 9.x file system. Every file has two parts, a data fork and a resource fork. In common usage, the data fork stores data preserved in application-unique formats (such as the information in a spreadsheet program). The resource fork stores data preserved in standard formats such as window descriptions, buttons, menu text, pictures, cursors, icons, and other data structures.

On Mac OS X, Constructor supports flattened resources, allowing you to save all the resource information in the data fork of the file instead of the resource fork. This is because not all filesystems support a separate resource fork and data fork in a file like the Mac OS 8.x and 9.x systems do. For information on saving your file in this flattened format, refer to “A Word on Flattened Resources.”

Constructor’s Resource Types

Fundamentally, Constructor is an interactive resource editor primarily devoted to the visual interface supported by the PowerPlant application framework. The resource types that Constructor manages include:

- PPob—describes a view and its contents
- MBAR—describes a menu bar
- MENU—describes an individual menu
- Mcmd—contains command numbers for menu items
- bitmaps—a wide variety of resources that use bitmaps, including icon resources, pictures, and patterns

- Txtr—describes the appearance of text
- STR#—an indexed list of strings
- CTYP—describes custom visual objects
- WIND—created in association with some PPob resources
- RidL—a resource ID list used by PowerPlant to manage message broadcasting, among other things.

Constructor has no editor for WIND or RidL resources. They are created, maintained, and managed by Constructor automatically. You modify Mcmd resources using the MENU editor.

Views A typical PowerPlant application has several visual elements such as windows, buttons, check boxes, and so forth. In PowerPlant terminology, these are all types of panes. Views and controls are sub-categories of panes. For more on the precise meaning of the terms pane, view, and control, see *The PowerPlant Book*..

With Constructor you design a window and its contents interactively right on the screen. With Constructor you see and manipulate windows and panes directly. When you're finished designing the window, Constructor saves the results in a PPob (PowerPlant object, pronounced pea-pob) resource.

Like the WIND, DLOG, and DITL resources used by the Macintosh Toolbox, the PPob resource allows you to specify the location and appearance of each element in a window. However, the PPob format is more comprehensive. The PPob also contains information about the behavior and relationships of the panes. The PPob describes a window, its location, its contents, the location of the contents within the window, and the visual containment hierarchy of the contents.

PowerPlant's UReanimator class uses the information in the PPob resource to create the visual appearance of your program. For example, when your program calls the function `LWindow::CreateWindow()` with a resource ID, it reads the corresponding PPob resource and generates the window based on the panes described there.

Constructor is first and foremost a visual editor for a PPob resource.

Menus A typical PowerPlant application also has menus. PowerPlant uses three menu-related resources, the MBAR, MENU, and Mcmd resources.*The PowerPlant Book* describes how PowerPlant code uses these three resources. With Constructor you can create, manage, and edit all three menu-related resources interactively. The resources do not have to be in a PowerPlant-based application.

Bitmaps Bitmaps abound in Mac OS programming. Constructor has a single bitmap editor you can use to create and modify bitmaps in a wide variety of resource types, including icon suites, cursors, pictures, patterns, and more. The resources do not have to be in a PowerPlant-based application.

Text Traits A typical PowerPlant application uses text in many ways. It might have static text items, buttons with labels, and so forth. PowerPlant describes the characteristics of text such as font, size, alignment, and color in a Txtr (text traits) resource. With Constructor you can create, manage, and edit Txtr resources.

String Lists Constructor also lets you create and edit an STR# resource that contains an indexed list of strings.

Custom Types The predefined PowerPlant objects may not be enough for your work. You are likely to define custom objects with special features. With Constructor you can create and define the attributes of custom visual objects with a CTYP resource.

Constructor Project Files

This section discusses how Constructor stores information in files, and how you work with those files. Topics include:

- Managing Files with Constructor
- A Word on Flattened Resources

Managing Files with Constructor

Constructor works on resources in Constructor project files. By convention, Constructor project file names end with the extension `.ppob`. This reflects Constructor's primary purpose as a visual interface builder. Information about windows and their contents is stored in the PPob resource in the Constructor file. However, the file may contain other resources.

Constructor can recognize and edit many (but not all) resource types. Constructor ignores non-Constructor resource types, but retains them in the file. You cannot see or edit non-Constructor resource types in Constructor.

You may prefer to keep Constructor resources in a Constructor file, and most other resources in a separate resource file. Double-clicking the file (either in the Finder or the CodeWarrior project window) then opens the appropriate resource editor: Constructor for Constructor-related resources, and your favorite resource editor for other resources.

TIP Some panes reference an icon or picture resource. If you have the resource in the same file as the PPob resource, Constructor displays the picture in the layout window. See "Displaying Pane Features."

Use the standard **File** menu commands to manage project files.

Table 2.1 Managing a Constructor project file

To...	Do this
Create a new file	Choose **New Project File** in the **File** menu.
Open a file	Choose **Open Project File** in the **File** menu.
Close a file	Choose **Close** in the **File** menu, or click in the close box of the project window.
Save a file	Choose **Save** in the **File** menu.
Save a file with a new name	Choose **Save As…** in the **File** menu. If you want to save the file as flattened resources in the data fork of the file instead of the resource fork, check the **Save as flattened resource file** check box, as shown in Figure 2.1. Refer to the section "A Word on Flattened Resources" for more information.

Your ability to save changes will be affected by the status of the file. If the file is locked or read-only, you can't save any changes. If the file has modified read-only status from a source code control system, you may not be able to check the file back into the system.

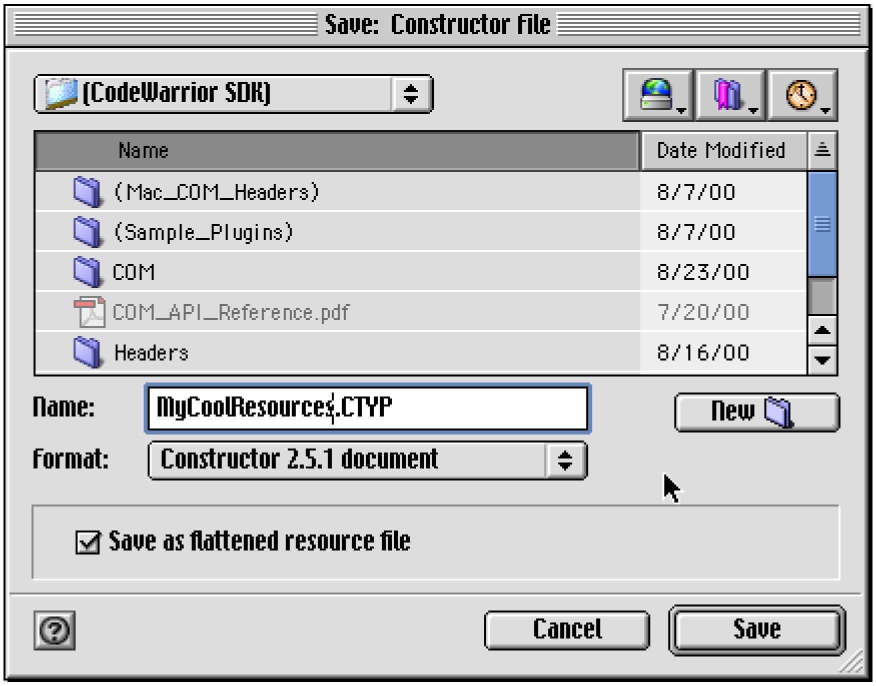
For more information on this topic, see the CodeWarrior IDE User Guide section on file privileges.

A Word on Flattened Resources

Mac OS X runs on many different disk formats, including non-traditional Mac HFS formats. This means that some disk formats (UFS and NFS, for example) do not implement the resource fork of a Mac OS file. Since Constructor is a resource editor, it is designed to alter data in the resource fork of a file. Apple Computer has introduced flattened resource files which are data files whose contents are what was traditionally in the resource fork of a file. Constructor allows the editing and saving of the traditional resource files, and also the new flattened resource files.

These flattened resources contain all the resource information within the data fork of the file instead of the resource fork. There is a new checkbox in the **Save As** dialog box that controls this feature, as shown in Figure 2.1. The **File** menu is where you find the Save As command.

Figure 2.1 Flattened Resources



Name	Date Modified
(Mac_COM_Headers)	8/7/00
(Sample_Plugins)	8/7/00
COM	8/23/00
COM_API_Reference.pdf	7/20/00
Headers	8/16/00

Name: MyCoolResources.CTYP

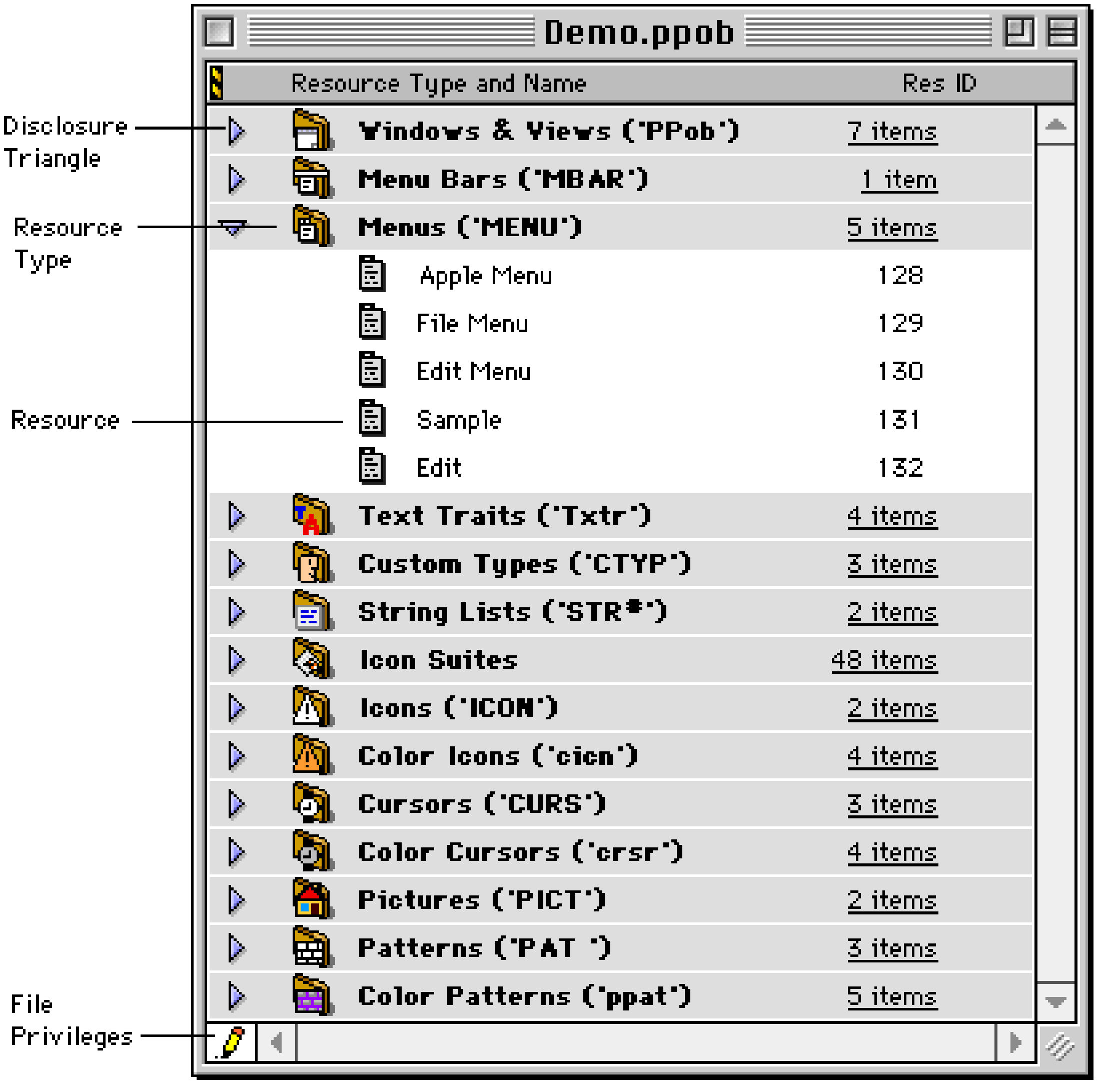
Format: Constructor 2.5.1 document

☑ Save as flattened resource file

Project Window

When you open a Constructor project file, Constructor displays a window that lists each type of resource, and the individual resources available for each type. Figure 2.2 shows this window.

Figure 2.2 A Constructor project window



Demo.ppob

Resource Type and Name	Res ID
Windows & Views ('PPob')	7 items
Menu Bars ('MBAR')	1 item
Menus ('MENU')	5 items
Apple Menu	128
File Menu	129
Edit Menu	130
Sample	131
Edit	132
Text Traits ('Txtr')	4 items
Custom Types ('CTYP')	3 items
String Lists ('STR#')	2 items
Icon Suites	48 items
Icons ('ICON')	2 items
Color Icons ('cicn')	4 items
Cursors ('CURS')	3 items
Color Cursors ('crsr')	4 items
Pictures ('PICT')	2 items
Patterns ('PAT ')	3 items
Color Patterns ('ppat')	5 items

The project window is the central or main Constructor window where you create new resources and manage existing resources. "Managing Resources Overview" introduces this topic.

Each grey bar in the project window reflects a resource type. There are many resource types, including:

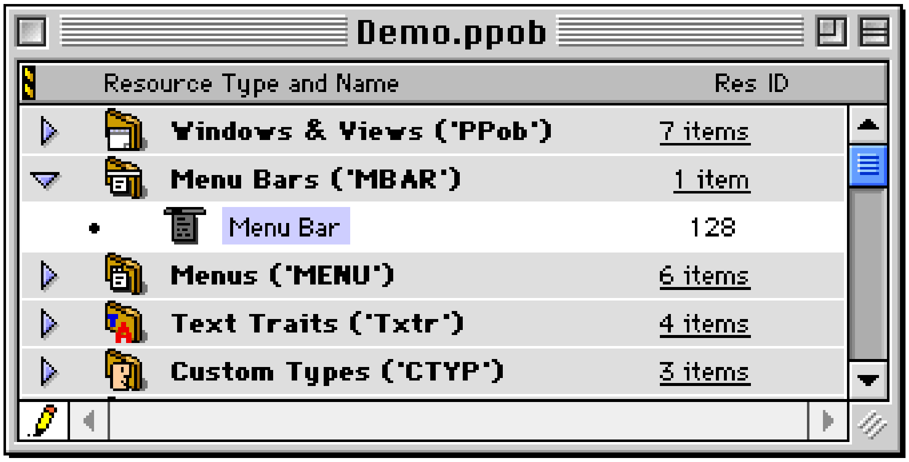
- Windows and Views for PPob resources
- Menu Bars for MBAR resources
- Menus for MENU (and Mcmd) resources
- Text Traits for Txtr resources

- Custom Pane Types for CTYP resources
- Various bitmap resources

To the left of each type heading is an disclosure triangle. Open or close the list of resources of that type by clicking on the triangle.

Whenever any resource listed in the project window has changed and has not been saved, a mark appears to the left of that resource, as shown in Figure 2.3, where the menu bar resource has changed.

Figure 2.3 A changed resource



Demo.ppob

Resource Type and Name	Res ID
Windows & Views ('PPob')	7 items
Menu Bars ('MBAR')	1 item
• Menu Bar	128
Menus ('MENU')	6 items
Text Traits ('Txtr')	4 items
Custom Types ('CTYP')	3 items

You may have multiple projects open simultaneously, so you can copy resources from one to another.

The file privileges icon in the lower left corner of the project window indicates the read/write status of the file, usually with respect to a source code control system. The file privileges icon works just like the same icon in the CodeWarrior IDE. For information on this topic, see the *IDE User Guide* section on file privileges.

Editor Windows

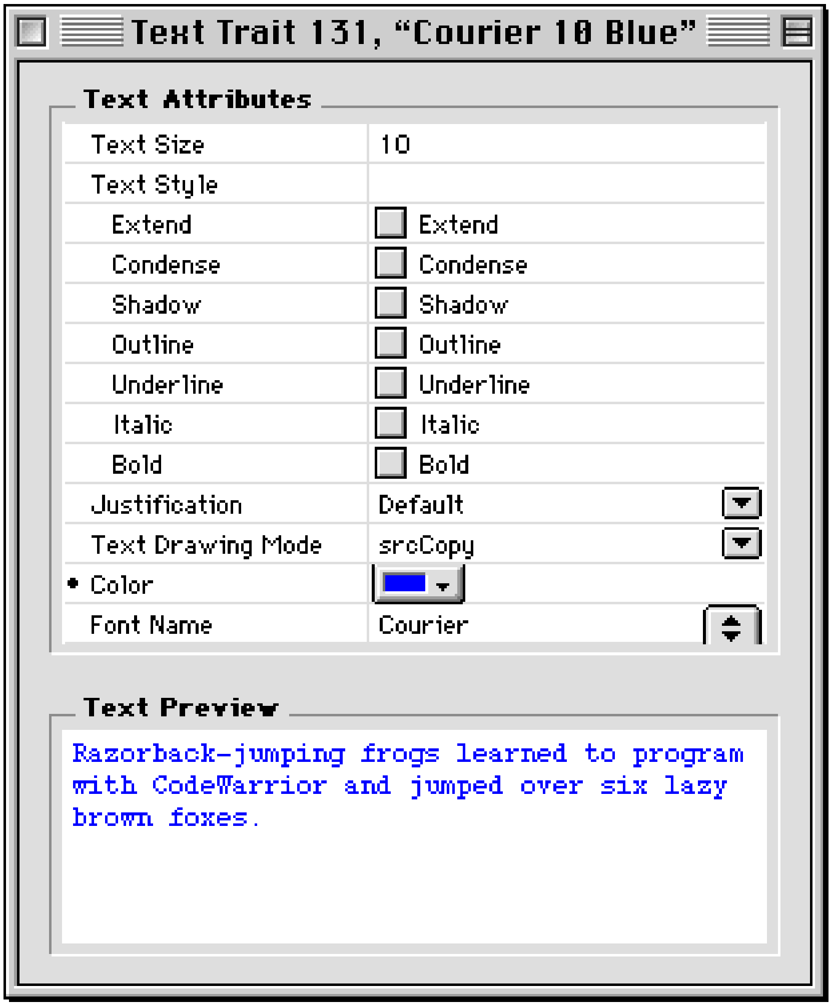
Each type of resource has a resource editor. To open an individual resource for editing, use one of the following techniques:

- double-click the resource in the project window

- select the resource(s) and press the Return key
- select the resource(s) and choose **Edit Resource** from the Edit menu

Whichever method you choose, Constructor opens an editor window that allows you to modify the resource. Each kind of resource has a different editor window. Figure 2.4 shows the Txtr editor window.

Figure 2.4 The text traits editor window



Text Trait 131, “Courier 10 Blue”

Text Attributes

Text Size	10
Text Style	
Extend	☐ Extend
Condense	☐ Condense
Shadow	☐ Shadow
Outline	☐ Outline
Underline	☐ Underline
Italic	☐ Italic
Bold	☐ Bold
Justification	Default
Text Drawing Mode	srcCopy
• Color	🟦
Font Name	Courier

Text Preview

Razorback-jumping frogs learned to program with CodeWarrior and jumped over six lazy brown foxes.

TIP You can close or save all Constructor windows by holding down the Option key and choosing **Close All** or **Save All** from the **File** menu.

See these topics for a discussion of the individual editors:

- Editing Windows and Views Overview
- Editing Menu Bars and Menus Overview
- Editing Bitmaps Overview
- Editing Text Traits Overview
- Editing String Lists Overview
- Editing Custom Pane Types Overviewr

Property Inspector Window

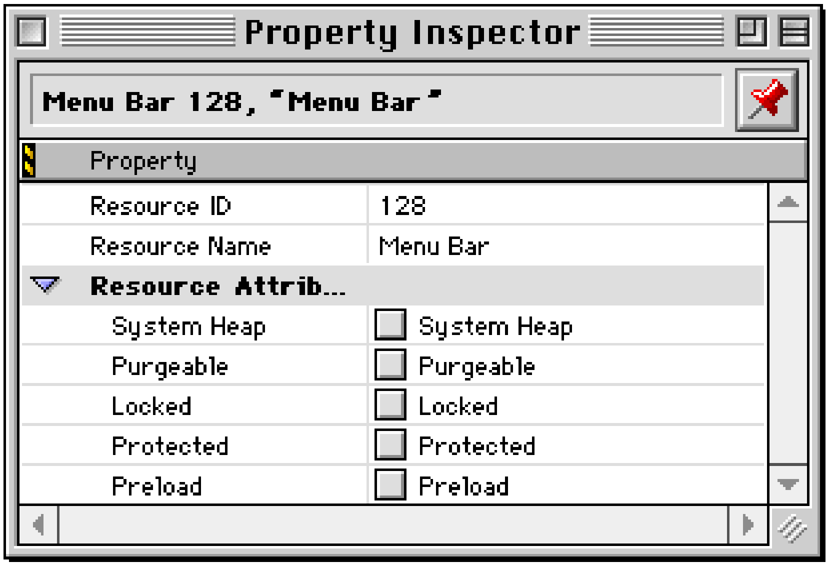
To open the Property Inspector Window, choose **Property Inspector** from the **Window** menu.

The Property Inspector Window displays attributes of a selected item, which might be:

- a resource in the Project Window
- a pane in the Layout Window or Hierarchy Window
- a menu or menu item in the Menu Bar Editor Window
- a class or data item in a Custom Type Editor Window

Whenever you select one of these items, its properties appear and can be edited in the Property Inspector Window. Figure 2.5 shows you this window displaying properties for a resource.

Figure 2.5 Resource properties



Property Inspector

Menu Bar 128, "Menu Bar"

Property	
Resource ID	128
Resource Name	Menu Bar
Resource Attrib...	
System Heap	□ System Heap
Purgeable	□ Purgeable
Locked	□ Locked
Protected	□ Protected
Preload	□ Preload

The top of the window identifies the item whose properties are on display. In many cases there are groups of properties, such as the Resource Attributes group in Figure 2.5. To display or hide a group of attributes, click on the disclosure triangle.

The Property Inspector Window stays in synch with whatever other view is open. If you modify the data in the Property Inspector, all other views on the same data update simultaneously. If you modify the data in some other view (such as in the Project Window, the Property Inspector updates its data at the same time.

For a discussion of how the Property Inspector works with various items in Constructor, see

- Resources—"Modifying Name, ID, and Attributes"
- Panes—"Setting Pane Properties"
- Menus—"Setting Menu Properties"
- Menu items—"Setting Menu Item Properties"
- Custom pane class—"Setting Class Properties"
- Custom pane data item—"Setting Data Item Properties"

User Interface Features

Constructor has many of the interface features you expect in a good interactive program. These include features that support:

- Selection—selecting items
- Moving, Copying, and Deleting—relocating and copying items
- Lists—managing lists of items
- Text Editing—modifying text in an item
- Undo—support for undo and redo

The figures in this section use the project window to illustrate the interface. The same techniques work in individual editor windows, when the technique is appropriate for the kind of data displayed in the window. Interface features are not available in all windows under all circumstances. In some situations, particular features are not appropriate.

Selection

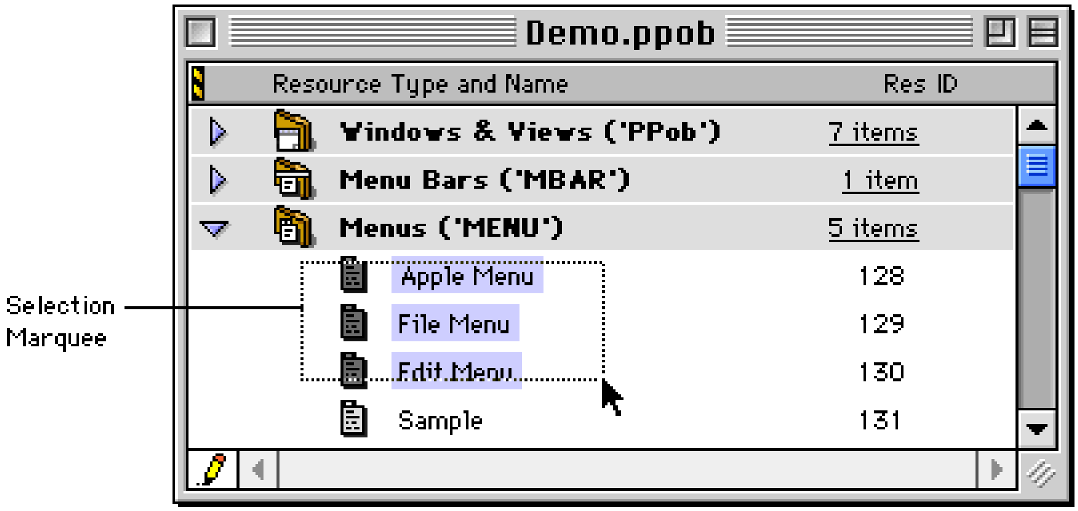
Constructor follows the standard behavior with respect to selecting items. In general, you first select an item or group of items, then you perform an operation on the item(s).

To select an item, you click it.

To extend a selection (select an additional item while keeping previously selected items), you press the Shift key and click.

To select contiguous items with one gesture, click and drag the mouse. Items that intersect the area defined by the resulting selection marquee become selected when you release the mouse button. Figure 2.6 illustrates a selection marquee.

Figure 2.6 Selecting multiple items with a marquee



Demo.ppob

Resource Type and Name	Res ID
Windows & Views ('PPob')	7 items
Menu Bars ('MBAR')	1 item
Menus ('MENU')	5 items
Apple Menu	128
File Menu	129
Edit Menu	130
Sample	131

For marquee selection to work, the drag must begin in an empty area of the window. If you click and drag on an item, you move or copy the item instead of starting a marquee.

TIP For a layout window, a drag that begins in the top-level view and not inside an individual pane starts a marquee selection. If you start your drag inside an individual pane, you select the pane and then move or resize it. To get around this problem, hold down the Control key when you begin a drag inside a pane.

To select all items in a window, choose the **Select All** item in the **Edit** menu.

To deselect all selected items, click outside of any item. For example, you can click the simulated desktop in a layout window to deselect all selected items.

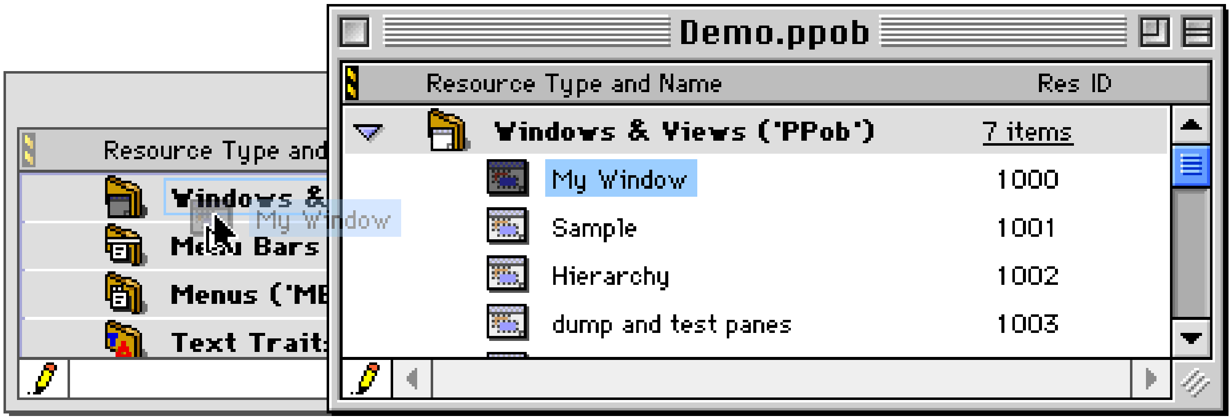
In the layout window only, you may also press the Control key while clicking an item. See "Layout Window."

Moving, Copying, and Deleting

To move or copy an item, you click and drag it. If you drag an item to another location in the same window, you move it.

If you drag an item from one window and drop the item in another window, Constructor makes a copy of the item in the destination window. The item is *not* removed from the originating window. Figure 2.7 illustrates copying a PPob resource by dragging from one project window into another.

Figure 2.7 Copying a resource by dragging



To copy an item you move within a window, press the Option key when beginning the drag or when dropping the item.

NOTE Dragging to create a copy does not work in the project window. Use the **Duplicate** command (see below).

Drag and drop also works in the bitmap editor, in both the Editing View and the Sample Views.

You can drop bitmaps onto either the Editing View or the Sample Views. If you drop on the Editing view, the original bitmap is scaled to fit. If you drop on the editing view, the bitmap does not scale.

You can drag from the Sample Views to other windows in Constructor and applications that support drag and drop, including the Finder desktop and the Scrapbook.

In addition to drag and drop, the standard **Cut**, **Copy**, and **Paste** commands in the **Edit** menu work in the traditional manner at all appropriate times.

You may also choose the **Duplicate** command in the **Edit** menu to make a copy of an item in the same window.

To delete an item, select it. Then press the Delete key, or use the **Clear** command from the **Edit** menu.

TIP When copying, duplicating, moving, or deleting a pane that contains other panes, you work on the container and its contents.

Lists

Some Constructor windows, like the project window, display lists of information. You can navigate through lists of items by typing the up or down arrow keys. Each time you do, the next or previous item in the list becomes selected.

Constructor uses disclosure triangles to open and close lists. Click on the triangle to change the state of the associated list. If the triangle is next to a selected item, you may also type the right or left arrow keys to open or close a list, respectively.

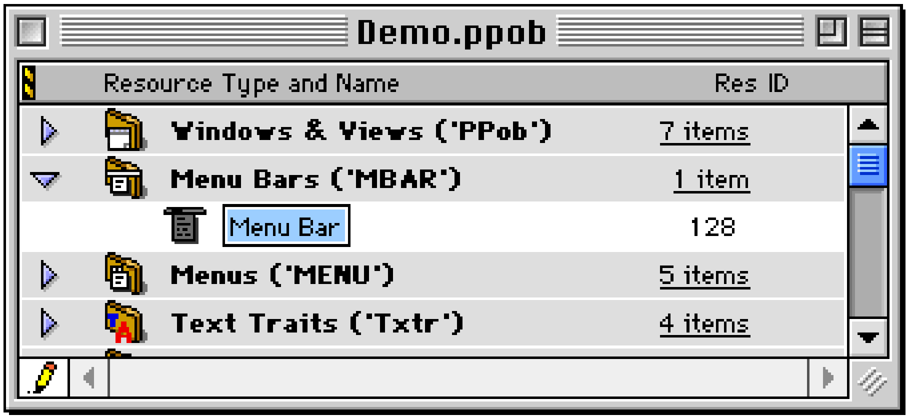
You can extend a selection in a list by pressing the Shift key and typing the up or down arrow key.

Text Editing

Textual items appear in many places in Constructor windows. For example, lists of items have names. Many windows have editable text fields.

To activate a text item for editing, click on the item. Then either move the cursor off the text, or wait a second. You know the item is editable when a frame appears around the text, the text in the item becomes selected, or when the text insertion cursor appears in the item. Figure 2.8 shows how you can edit items in the project window.

Figure 2.8 Editing text in-place in the project window



Demo.ppob

Resource Type and Name	Res ID
Windows & Views ('PPob')	7 items
Menu Bars ('MBAR')	1 item
Menu Bar	128
Menus ('MENU')	5 items
Text Traits ('Txtr')	4 items

You navigate through the text in an item by using the arrow keys. You select text by dragging with the mouse in the text item. You can select a word by double-clicking the word.

To move from one item to the next in a series of text fields, use the Tab key. For example, if you are editing the name of a resource in the project window, if you press the Tab key the ID field becomes selected and editable. You can then modify the ID number if you wish.

Undo

Constructor has single-level undo. Most operations can be undone (and subsequently redone) using the **Undo** or **Redo** items in the **Edit** menu.

If undo is unavailable after a given action, the menu item will read **Can’t Undo**.

3

Managing Resources

This chapter discusses how to create, copy, modify, open, and delete resources in a Constructor project.

Managing Resources Overview

You manage resources from the Constructor project window. For details on the project window, see “Project Window.” In general, you select the resource or resources you want to work with, and then do something with or to that resource.

The topics discussed in this chapter include:

- Making a New Resource
- Copying a Resource
- Modifying Name, ID, and Attributes
- Opening a Resource for Editing
- Deleting a Resource

For the most part, you perform each of these tasks in the same way regardless of the type of resource involved.

Making a New Resource

To make a new resource, choose **New Resource** from the **Edit** menu. What happens next depends upon the selection in the project window.

Based on the selection, if Constructor knows for sure what kind of resource you want, it creates an untitled resource of the correct type, and gives it the next available ID number. If Constructor does not know what type of resource you want, Constructor displays a dialog box that asks you to specify additional information.

Other chapters in this manual discuss how to edit the contents of each individual type of resource. This section simply discusses how to work with the resource as a whole.

The topics in this section are:

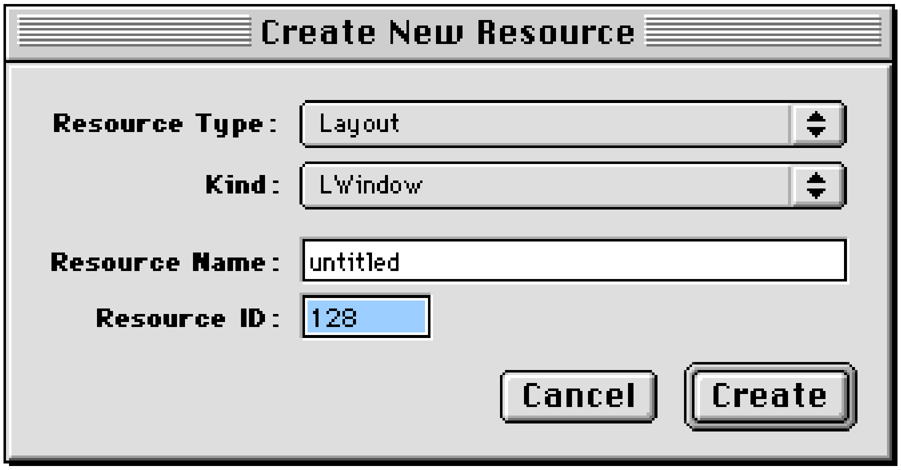
- Creating Any Constructor Resource
- Creating a Layout Resource
- Creating a Menu Bar Resource
- Creating a Menu Resource
- Creating a Bitmap Resource
- Creating a Text Traits Resource
- Creating a String List Resource
- Creating a Custom Type Resource

Creating Any Constructor Resource

In the project window, if you have no resource selected, a layout resource selected, or resources of different types selected, then Constructor does not know for sure what kind of resource you want to create. Constructor presents the Create New Resource dialog, as shown in Figure 3.1.

TIP To see this dialog no matter what resource (if any) is selected, use Command-Option-K.

Figure 3.1 The Create New Resource dialog



Resource Type: Layout

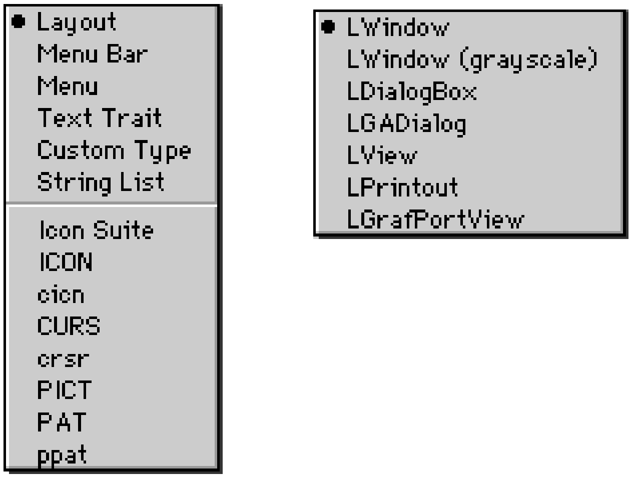
Kind: LWindow

Resource Name: untitled

Resource ID: 128

You select the desired resource type from the Resource Type pop-up menu, as shown in Figure 3.2.

Figure 3.2 Available resource types



If the resource type is a layout, you select the top-level view type from the Kind pop-up menu. If the resource type is not a layout, the Kind pop-up menu is disabled. You may also specify the resource name and resource ID number. Then click the **Create** button to create the new resource.

Use this technique to make a Constructor resource of any type. Each type also has one or more methods for creating that specific type.

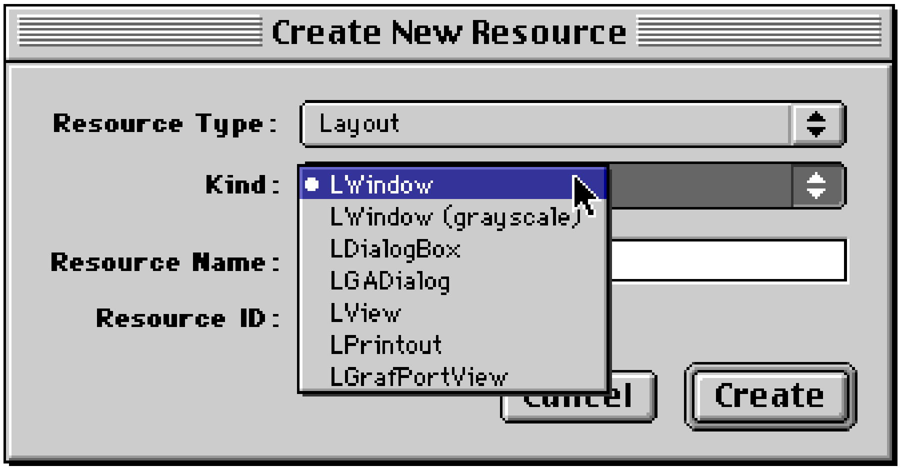
Creating a Layout Resource

There are three ways to create a new layout resource.

- Follow the instructions in “Creating Any Constructor Resource”.
- Select an existing Windows and Views resource in the project window (or the grey resource type header bar), and then choose **New Resource** from the **Edit** menu.
- Drag a view as described in the section “Catalog Window”.

If you have a layout resource selected in the project window, when you choose **New Resource** the Create New Resource dialog illustrated in Figure 3.3 appears. You must also choose the view type you wish to be the top-level view for this layout resource.

Figure 3.3 Creating a view with the new resource dialog



Resource Type: Layout

Kind: LWindow

Resource Name:

Resource ID:

You may choose from LWindow, LDialogBox, LView, LPrintout, and LGrafPortView. Choose the top-level view type you wish this layout resource to have, and click the **Create** button.

For detailed information on these view classes, and when it is appropriate to use them, see *The PowerPlant Book*. Table 3.1 lists the usual circumstances for which you use each type of view.

Table 3.1 Choosing the top level view in a PPob

View	Use for
LWindow	most windows, including floating palettes
LWindow (grayscale)	same as LWindow, but has a grayscale background
LDialogBox	dialogs, with default OK and Cancel buttons
LGADialog	same as LDialogBox, but has a grayscale background
LView	generic view without a window
LPrintout	printing
LGrafPortView	an external with a container application, such as PhotoShop and HyperCard

Alternatively, you can drag *any* pane from the Catalog Window into the project window.

TIP Creating a new PPob by dragging means that you aren't limited to the five views available in the Create New Resource dialog. For example, you can drag a custom window class (derived from LWindow) from the Catalog Window and use it as a top-level view.

Creating a Menu Bar Resource

There are two ways to create a new menu bar resource.

- Follow the instructions in "Creating Any Constructor Resource".
- Select an existing menu bar resource in the project window (or the grey resource header bar), and then choose **New Resource** from the **Edit** menu.

Constructor creates a new, untitled menu bar resource with the next available ID number.

NOTE A typical Constructor project file has only one MBAR resource. PowerPlant expects that resource to have ID number 128.

Creating a Menu Resource

There are two ways to create a new menu resource.

- Follow the instructions in “Creating Any Constructor Resource”.
- Select an existing menu resource in the project window (or the grey resource header bar), and then choose **New Resource** from the **Edit** menu.

Constructor creates a new, untitled menu resource with the next available ID number.

Creating a Bitmap Resource

You can use constructor to create a variety of resources that contain bitmap data. The available resources are:

- icon suites—‘icl8’, ‘icl4’, ‘ICN#’, ‘ics8’, ‘ics4’, ‘ics#’, ‘icm8’, ‘icm4’, and ‘icm#’ representing large icons, small icons, and icon masks in 8-bit color, 4-bit color, and black and white
- ‘ICON’—black and white icon
- ‘cicn’—color icon
- ‘PICT’—bitmaps, not vector-based drawing commands
- ‘PAT ’—black and white pattern
- ‘ppat’—color pattern
- ‘CURS’—black and white cursor
- ‘crsr’—color cursor

There are two ways to create a new bitmap resource:

- Follow the instructions in “Creating Any Constructor Resource”.
- Select an existing bitmap resource in the project window (or the grey resource header bar), and then choose **New Resource** from the **Edit** menu.

Constructor creates a new, untitled bitmap resource of the selected type, with the next available ID number.

Creating a Text Traits Resource

There are two ways to create a new text traits resource.

- Follow the instructions in "Creating Any Constructor Resource".
- Select an existing text traits resource in the project window (or the grey resource header bar), and then choose **New Resource** from the **Edit** menu.

Constructor creates a new, untitled text traits resource with the next available ID number.

Creating a String List Resource

There are two ways to create a new string list resource.

- Follow the instructions in "Creating Any Constructor Resource".
- Select an existing string list resource in the project window (or the grey resource header bar), and then choose **New Resource** from the **Edit** menu.

Constructor creates a new, untitled string list resource with the next available ID number.

Creating a Custom Type Resource

There are two ways to create a new custom type resource.

- Follow the instructions in "Creating Any Constructor Resource".
- Select an existing custom type resource in the project window (or the grey resource header bar), and then choose **New Resource** from the **Edit** menu.

Constructor creates a new, untitled custom type resource with the next available ID number.

Copying a Resource

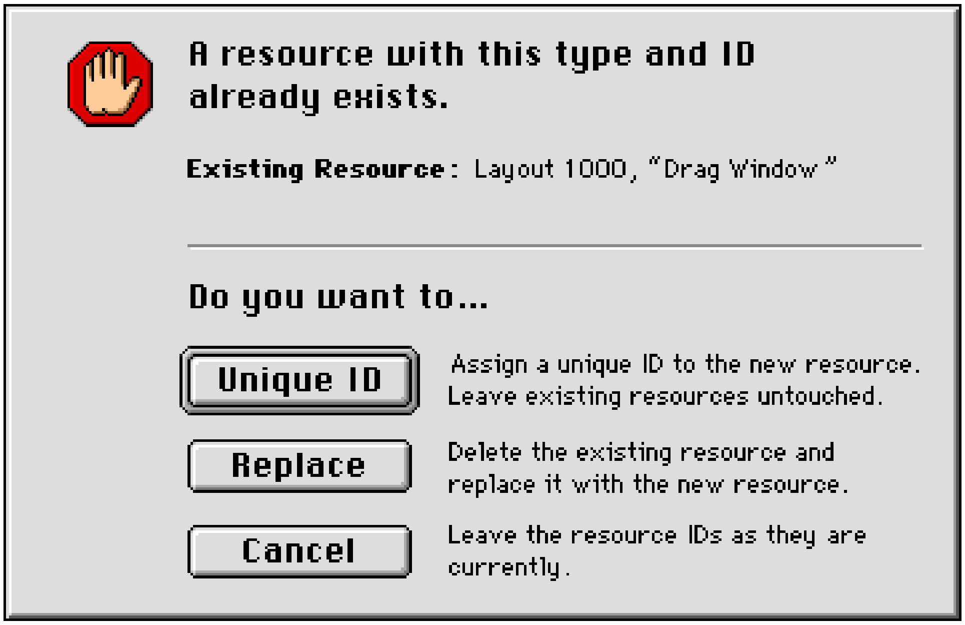
When you create a new resource, you must then edit the resource to define its contents properly. Sometimes it is more efficient to make a copy of an existing resource, and then make minor modifications to the copy.

There are three ways to copy a resource. First, you select one or more resources. Then you can:

- use the **Copy** and **Paste** commands in the **Edit** menu
- use the **Duplicate** command in the **Edit** menu
- drag the resource(s) into another project

If you copy and paste in the same project, or if the destination project has a resource of the same type with the same ID number as the copy, Constructor may complain about a duplicate ID number. The Duplicate Resource ID dialog appears.

Figure 3.4 Duplicate Resource IDs dialog



A resource with this type and ID already exists.

Existing Resource: Layout 1000, "Drag Window"

Do you want to...

Assign a unique ID to the new resource. Leave existing resources untouched.

Delete the existing resource and replace it with the new resource.

Leave the resource IDs as they are currently.

Constructor lets you assign new a new ID number, replace the existing resource, or cancel the operation. The new ID number will be the next available ID number for that type.

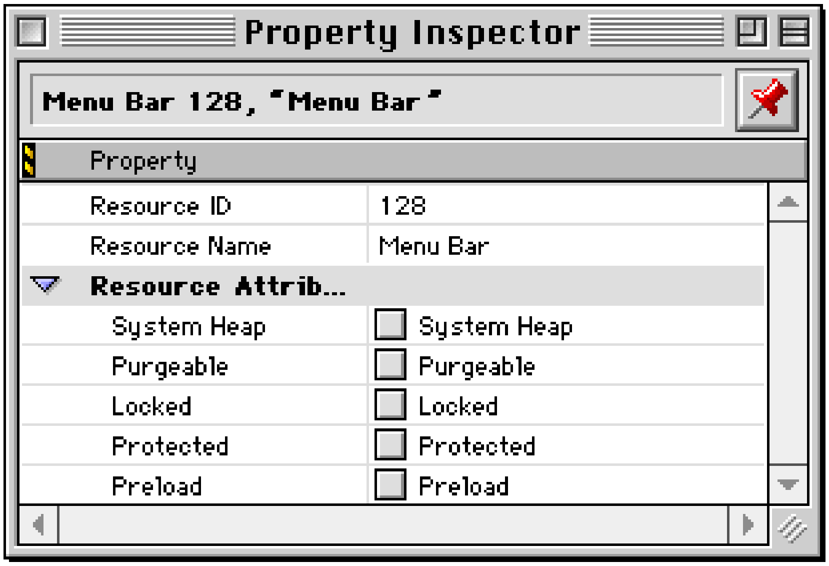
Modifying Name, ID, and Attributes

Each resource has a name, ID number, and attributes. The name is optional, but helps identify the actual purpose of the resource.

The name and ID can be edited in place in the Project Window. Simply click on the text or ID number, and wait a moment. A frame appears around the item, and the text becomes selected inside the frame. You can then edit the text. Use the Tab key to move between the name and ID number.

To edit resource attributes as well as name and ID, select the resource and use the Property Inspector Window. Figure 3.5 shows the resource properties in an inspector window.

Figure 3.5 Resource properties



Property Inspector

Menu Bar 128, "Menu Bar"

Property	
Resource ID	128
Resource Name	Menu Bar
Resource Attrib...	
System Heap	□ System Heap
Purgeable	□ Purgeable
Locked	□ Locked
Protected	□ Protected
Preload	□ Preload

You cannot change the resource type. However, you can modify the name, ID number, and attributes.

By default, all attributes for Constructor resources are off. PowerPlant works correctly with the attributes off. Typically, there is no need to turn on any attribute, and you should not do so unless you are familiar with the consequences.

Opening a Resource for Editing

After you select a resource, you can open an editor to modify the contents of the resource. To open the editor associated with a resource, you can:

- double-click the resource
- select the resource(s) and press the Return key

- select the resource(s) and choose **Edit Resource** from the **Edit** menu

You may open and edit several resources of any type at the same time. If you open them simultaneously, Constructor staggers the editor windows on the monitor. Details of the various editor windows can be found in the appropriate chapters of this manual.

Deleting a Resource

To delete a resource or resources, first select the resource(s).

You may then delete the resource(s) using one of these techniques:

- choose **Clear Resource** from the **Edit** menu
- press the Delete key
- drag the resource(s) to the trash

You can Undo a delete until you perform another action.

4

Editing Windows and Views

This chapter discusses the tools you use and the tasks you perform to modify the contents of a view (window) using Constructor.

Editing Windows and Views Overview

We are at the core of what Constructor is all about. You want to quickly and easily specify the appearance of a view and all of its contents. You can do so with Constructor.

This chapter discusses how to manage and modify the *contents* of windows and views described in a PPob resource. For information on how to create, copy, modify, open, and delete PPob resources as a whole, see “Managing Resources Overview.”

The terminology in this chapter is important, and can be confusing. You will be using windows to create windows. You will have views that contain views. To minimize this confusion, here are a few terms and how they are used in this chapter.

- window—a display feature of Constructor! We will use the term “window” to refer only to Constructor’s own interface.
- PPob—the overall visual element you build with Constructor, represented as a PPob resource. Typically, this is a window that will appear in your own program. Nevertheless, we will call this a PPob to distinguish it from Constructor’s own windows.
- pane—individual items inside a PPob. These individual items might be LView objects—visual representations of PowerPlant view classes. However, we will use the term “pane” to refer to views, panes, and controls generically. They are, in fact, all LPane objects. The only exception to this will be the term “top-

level view" used to describe the pane that encloses all other panes in the PPob.

With the definition of terms out of the way, we can turn to the real subject at hand. Conceptually, this chapter consists of two sections: what you see and what you do. The what you see section explains the Constructor windows involved in editing a view. The what you do section discusses the tasks you perform. The topics are:

- Windows for Views—the windows you use to display and set PPob-related information
- Building a Visual Interface—the tasks you perform to build a PPob and its contents

Windows for Views

This section discusses the Constructor windows involved in creating a PPob and its contents. The principal features and characteristics of each window are discussed in detail. This section covers what you see in Constructor. The topics include:

- Catalog Window—the window that contains the panes you can add to any PPob in any project
- Layout Window—the window that displays your PPob's visual appearance
- Hierarchy Window—the window that displays the containment hierarchy for the panes in your PPob
- Layout and Hierarchy Compared—differences and similarities between these two windows

In addition, you examine pane properties in the Property Inspector Window. There is one Catalog Window and one Property Inspector Window for Constructor shared by all PPob resources. However, each PPob resource has its own Layout Window and Hierarchy Window.

Catalog Window

The PowerPlant Catalog Window contains every class related to visual objects—that is, objects you can display. This window contains standard PowerPlant classes and custom display classes. It is the catalog you use as a source for new panes when building a PPob. See “Adding Panes to a PPob.”

To open this window, choose the **Catalog** item in the **Window** menu. This window shows the class name and the class ID for each pane.

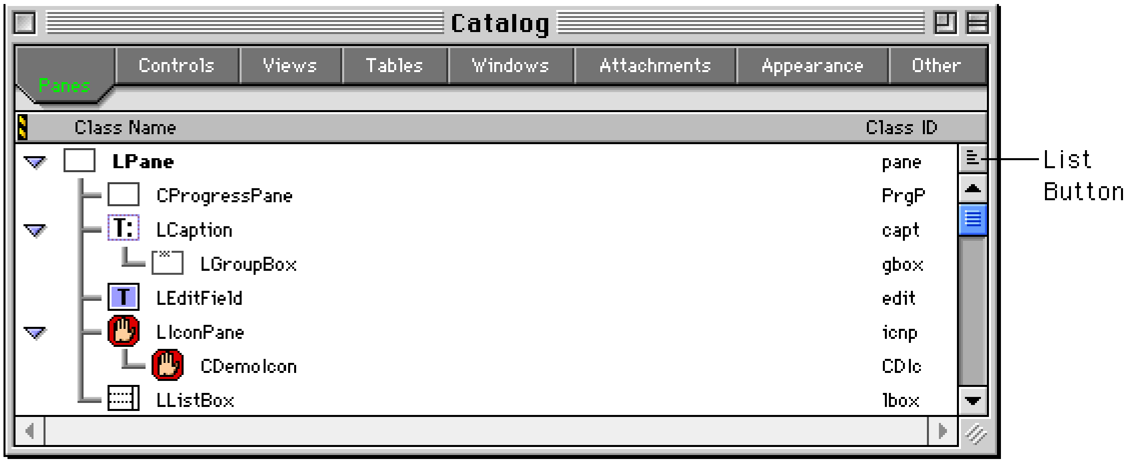
The panes are organized into groups defined by a PowerPlant base class. The groups are:

- Panes
- Controls
- Views
- Tables
- Windows
- Attachments

To see the contents of a particular group of panes, click the corresponding tab in the scrolling bar across the top of the window.

You can view the panes by class hierarchy, as shown in Figure 4.1, or sort the panes by name. Click the List Button at the top of the vertical scroll bar to switch between hierarchy and name order.

Figure 4.1 The PowerPlant Catalog window



Class Name	Class ID
LPane	pane
CProgressPane	PrgP
LCaption	capt
LGroupBox	gbox
LEditField	edit
LIconPane	icnp
CDemoIcon	CDIc
LListBox	lbox

List Button

Layout Window

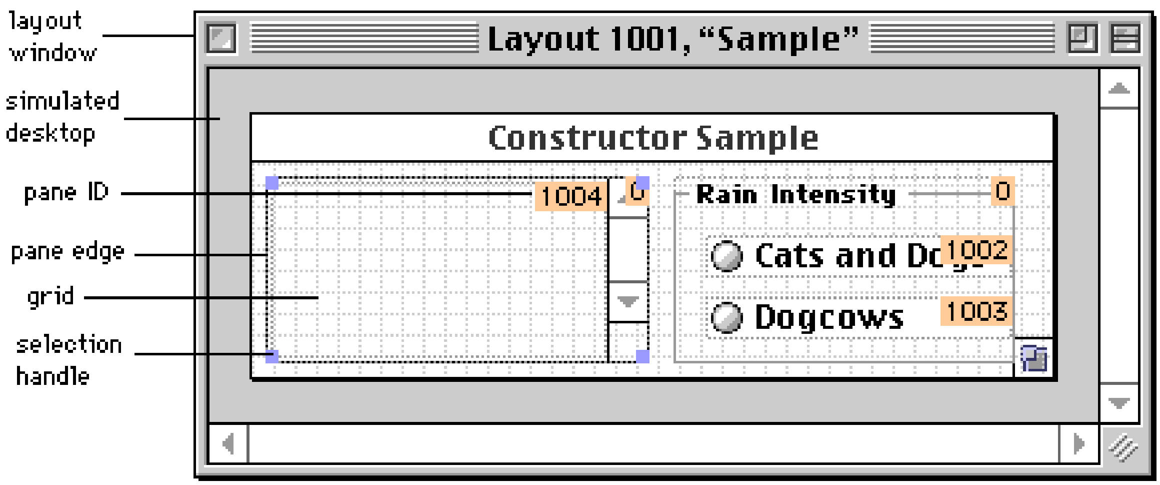
The layout window is where you do most of your visual work in Constructor. This is the window in which you arrange your panes for functional and aesthetic appearance.

When you open a PPob resource, the layout window for that resource appears. You may open the layout window in three ways:

- double click the PPob resource in the project window
- select the PPob resource and press the Return key
- select the PPob resource and choose Edit Resource from the Edit menu

Figure 4.2 shows a layout window. The PPob appears on a simulated desktop inside the layout window.

Figure 4.2 The layout window



The title bar of the layout window shows the nature of the resource, the resource ID number, and the name of the resource. In this case it is a PPob resource, ID number 1001, and named “untitled.”

Within the layout window there is a PPob. In this case the top-level view is an LWindow. If you have a PPob whose top-level view has a title bar (such as an LWindow), you can see the title in the layout view. You can change the size of a top-level view by dragging a selection handle.

The nature of the top-level view depends on the view type you select for the PPob resource when you create it. Typically the top-level view is an LWindow, LView, LPrintout, LGrafPortView, LDialogBox, or LGADialog. See “Creating a Layout Resource.”

TIP You can’t change the top-level view type after you have created the PPob resource. If you want to change the top-level view, make a new resource with the correct view type. Then copy the contents of the original view to the new view.

As you add panes to the PPob, you can rearrange them in the layout window, modify them, resize them, and so forth. See “Building a Visual Interface.”

Hierarchy Window

The hierarchy window is an alternative but critical method of visualizing a PPob and its contents. Panes in a PPob are arranged both spatially and hierarchically. A pane that is visually inside another pane may or may not be contained hierarchically within that pane.

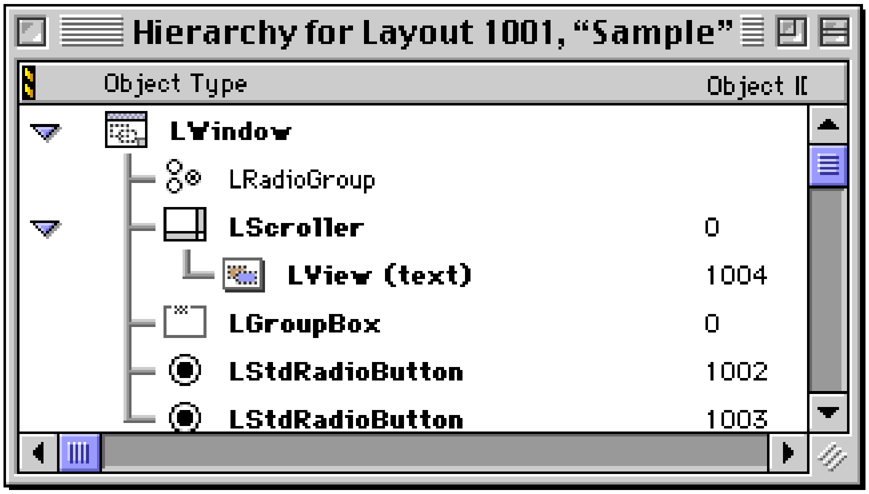
The containment hierarchy is an important aspect of PowerPlant programming. Sometimes it is vital that a pane be hierarchically contained inside another pane. For example, if you have a picture that you want to scroll, the LPicture pane *must* be hierarchically located inside a scrolling view pane.

The containment relationships among panes is stored in the PPob resource, along with all the other pane-related information. PowerPlant use this information to set up the superview/subview relationships in the PowerPlant objects at runtime.

The hierarchy window displays the containment hierarchy for a PPob. To display a PPob’s hierarchy window, choose the **Show Object Hierarchy** item from the **Layout** menu, when that PPob’s layout window is active. The layout window must be open to view the hierarchy window.

Figure 4.3 shows a hierarchy window.

Figure 4.3 The hierarchy window



The top-level view is always at the top of the hierarchy window. When you select an item, the hierarchy lines become darker. In this case, the LTextEdit pane is contained within an LScroller pane, and that is inside the LWindow pane that forms the top-level view.

NOTE The hierarchy window also shows you the order in which panes will be drawn. Your application will draw from the top of the hierarchy down. Panes further down the list may hide those higher up if they overlap and draw over the same area.

You use the hierarchy window to rearrange the hierarchical order of the panes in a PPob. See “Arranging Panes Hierarchically.”

Layout and Hierarchy Compared

Constructor maintains a single PPob resource for each top-level view. The PPob contains all the data on all the panes. The layout and hierarchy windows are simply different views on the same data set.

The layout window specializes in viewing the spatial relationships among the panes in a PPob resource. You can see what they look like relative to each other, move them around, and so forth. You use the layout view to arrange the visual appearance of a PPob.

The hierarchy window specializes in viewing the containment relationships among the panes in the PPob resource. You can see which panes actually contain other panes.

Either window provides access to the data in the PPob resource. Each looks at the data a little differently. However, you can use either view—layout or hierarchy—as a source or destination when copying panes between projects. For example, you can copy a pane from a layout window in one project, and paste it into the hierarchy window in another project.

Except that each window has a special way of viewing the PPob data, the layout and hierarchy windows are interchangeable. If you cut an item in the hierarchy window, it disappears from the layout. If you copy an item in the layout, it appears in the hierarchy. Constructor keeps both windows updated and consistent at all times.

Building a Visual Interface

This section discusses the individual tasks you must perform when editing a PPob resource. You must add panes to a PPob, arrange them visually and hierarchically, create certain groups, and set properties for the panes. For information on how to create a new PPob resource, see “Creating a Layout Resource.”

These tasks are not sequential. You perform these tasks concurrently, and in virtually any order that fits your personal style. However, at one time or another you will perform each task while creating a PPob.

The topics in this section are:

- Adding Panes to a PPob—putting a new pane into a PPob
- Displaying Pane Features—how to display pane edges, invisible panes, pane IDs, and pictures inside panes
- Arranging Panes Visually—organizing panes for an aesthetic display
- Arranging Panes Hierarchically—organizing panes in a containment hierarchy

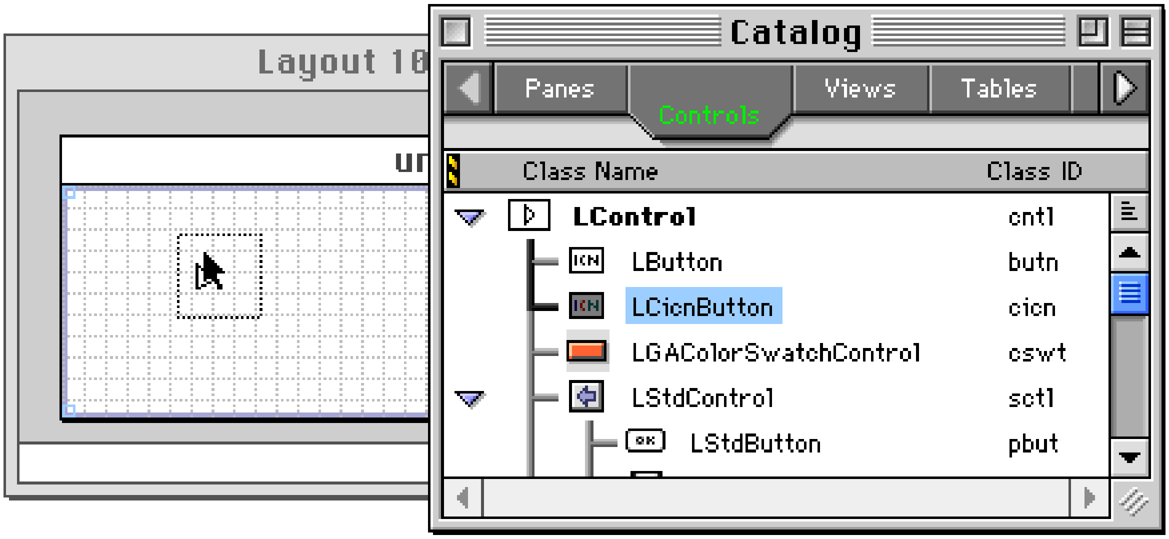
- Working With Radio Groups—organizing groups of radio buttons
- Working With Tab Groups—organizing groups of text fields so the user can tab from one to the next
- Setting Pane Properties—specifying the contents and appearance of an individual pane
- Removing Panes—taking a pane out of a PPob

Adding Panes to a PPob

To add a pane to a PPob, you copy a pane from some other source and move it into the PPob. The most common method is to drag a pane from the Catalog Window into a PPob. Figure 4.4 illustrates the process of adding an LCicnButton to a PPob.

You can drop panes only into a PPob. Constructor does not let you drop them on the simulated desktop around the PPob.

Figure 4.4 Adding a pane to a PPob



Class Name	Class ID
LControl	cntl
LButton	butn
LCicnButton	cicn
LGAColorSwatchControl	cswt
LStdControl	sctl
LStdButton	pbut

A new pane created by dragging from the Catalog Window always starts out with a default size and default attributes. For example, panes designed for text are usually empty—that is, they have no text in them. You modify the pane after adding it to the PPob. See "Setting Pane Properties."

NOTE You cannot copy from the Catalog Window and paste into a PPob. You must drag from one to the other.

You can copy and paste panes within a PPob or between PPobs. You use the standard techniques for copy and paste. See "Moving, Copying, and Deleting."

Displaying Pane Features

When working in the layout window, you can control the display of certain pane characteristics. Use menu commands for:

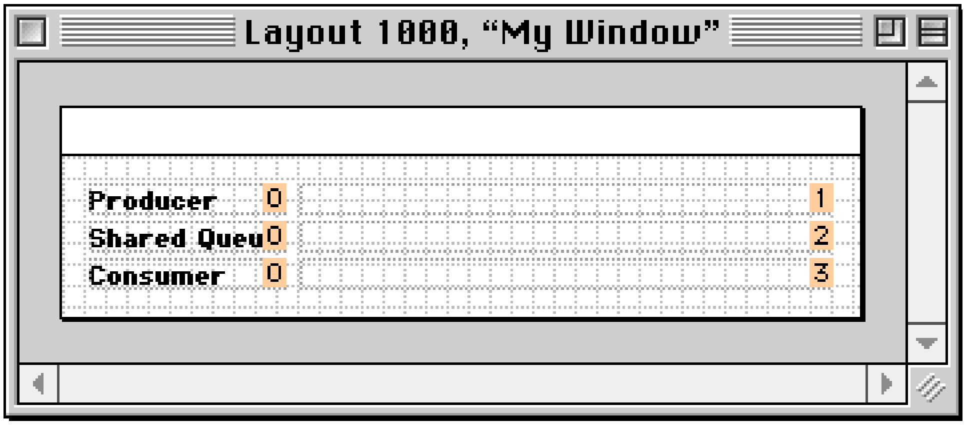
- Displaying pane ID—toggle the display of pane ID numbers
- Displaying pane edges—toggle the display of pane boundaries
- Displaying invisible panes—toggle the display of panes that are invisible at runtime
- Displaying icons and pictures—show the actual contents of certain panes in the layout window

Displaying pane ID

Each pane has a pane ID. PowerPlant uses the pane ID to find the pane object in memory. You set the pane ID in the Property Inspector Window.

To display ID numbers, choose the **Show Pane IDs** command in the **Layout** menu. The ID of each pane appears. When the IDs are being displayed, the menu item text changes to **Hide Pane IDs**. Choose **Hide Pane IDs** to conceal the ID numbers. Figure 4.5 shows pane IDs on display.

Figure 4.5 Pane IDs on display



See *The PowerPlant Book* for information on how to use pane IDs.

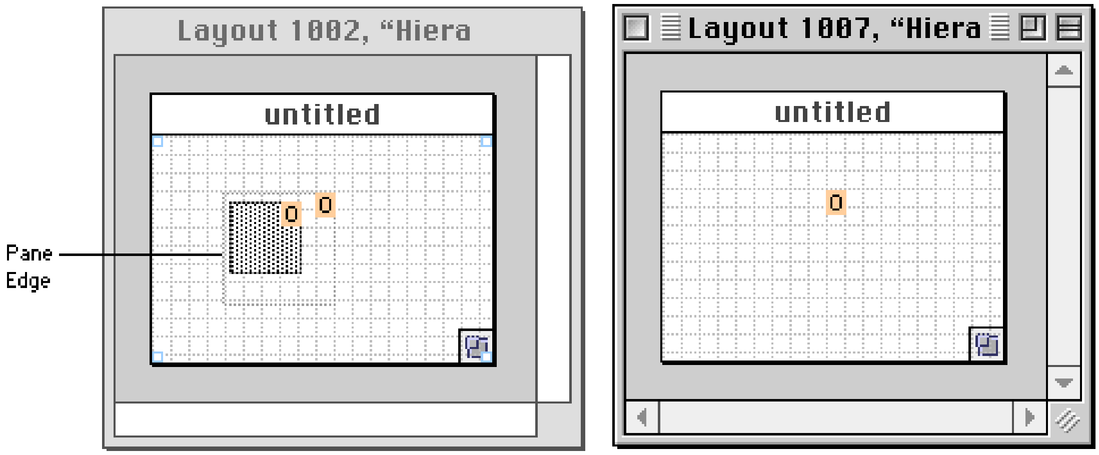
Displaying pane edges

Every pane has a boundary or edge. However, some pane types have no actual visual representation. For example, an LView pane is simply a container for other panes. If pane edges are hidden, you cannot see an LView object in the layout window. If you show pane edges, you can see the dotted outline of the LView pane.

To display pane edges, choose the **Show Pane Edges** command in the **Layout** menu. The bounds of each pane appear as a dotted outline. When edges are being displayed, the menu item text changes to **Hide Pane Edges**. Choose **Hide Pane Edges** to conceal the edges.

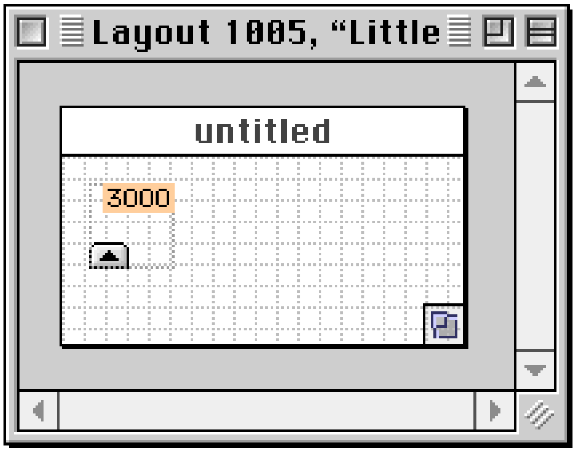
Figure 4.6 shows two PPobs, each with an LView object with ID zero. The ID number is on display. In the window on the left, the pane edge is visible. In the window on the right, the pane edge is hidden, so the pane bounds are invisible. If you hide both pane ID and edges, the LView object would be completely hidden. You could still select it, but you couldn't see it except for the selection handles while it is selected.

Figure 4.6 Pane edges



Showing edges is also useful for panes that contain an icon or picture, or for panes that do not have a visual frame. The image might be smaller than the actual pane size. If you display pane edges, you can see the actual size of the pane. Figure 4.7 illustrates this feature. This PPob contains an LButton pane that uses a very small picture.

Figure 4.7 A picture smaller than a pane



Displaying invisible panes

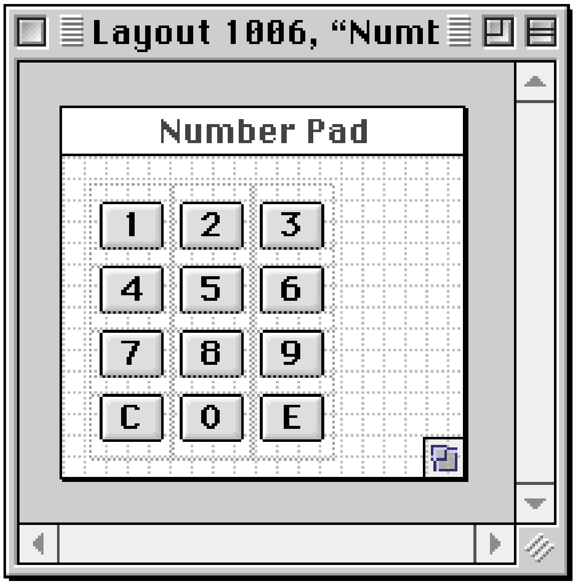
Each pane has a *Visible* check box in the Property Inspector Window. You use this pane attribute to determine whether a pane is visible or not at runtime. However, Constructor reads the same attribute and allows you to show a pane even if it is intended to be invisible at runtime.

To show non-visible panes, choose the **Show Invisible Panes** command in the **Layout** menu. Choose **Hide Invisible Panes** to conceal the non-visible panes once again.

Displaying icons and pictures

Some panes contain an icon or picture. For example, an LButton pane references a set of PICT resources. If you have the icon or picture resource in the same file as the PPob resource, Constructor displays the actual icon or picture in the layout window. Figure 4.8 shows some pictures on display inside button panes.

Figure 4.8 Some icon buttons



Keeping icon and picture resources in the same file as the PPob can make editing the layout window a lot more intuitive. You can see the contents of the item, not just an empty rectangle. You can even edit the icons or pictures in Constructor. See "Editing Bitmaps Overview."

Arranging Panes Visually

This section discusses the various techniques you can use to arrange your panes in a visually pleasing and aesthetic pattern within a PPob. Constructor has features for:

- Selecting an obscured pane—select a pane behind another
- Moving a pane—relocate panes by dragging
- Nudging a pane—move a pane pixel-by-pixel
- Using a grid—display and/or snap to a definable grid
- Aligning panes—align multiple panes in various ways
- Distributing panes—spread panes evenly
- Resizing a pane—change a pane's size

Most operations require that you first select the pane or panes on which you wish to work, and then do something to them. You select panes in the usual ways: click, Shift-click to extend a selection, or click and drag using the marquee. You can begin a marquee selection inside an existing pane if you hold down the Control key when you start the drag. See "Selection."

Selecting an obscured pane

You select a pane using standard selection techniques. See "Selection." However, you may have panes stacked on top of each other such that one pane completely covers (and thus obscures) a pane behind it.

If you have one pane on top of another, you can select the item *behind* the currently selected item by holding down the Command key while clicking. To select the next item in the stack, Command-click again. This cycles through the stacked panes, including the top-level view, without affecting any other panes. Continue Command-clicking until you have selected the desired pane.

NOTE If you have Command-clicked to select an obscured pane, you must hold down the Option key before dragging, resizing, or double-clicking that pane to prevent Constructor from changing the selection.

Moving a pane

To move a pane, drag the selection with the mouse and drop it at the desired location. You can select and drag multiple panes as well. You may also use the Property Inspector Window to change the pane’s location.

Hold down Shift and/or Command keys while dragging to constrain the movement of the pane.

To constrain the movement to the same horizontal or vertical line, press the Shift key *after* you start dragging. This allows you to move a selection in a straight line. If you let go of the Shift key, the constraint is released and you can move the selection freely.

NOTE Holding down the Shift key while selecting allows you to extend the selection.

To constrain the movement to a grid, you choose the **Snap to Grid** command in the **Layout** menu. To reverse the snap setting temporarily, press the Command key *after* you start dragging. If you let go of the Command key, the snap setting is determined from the menu.

NOTE Holding down the Command key while selecting allows you to select a pane behind the front pane.

There is one pane you cannot move. You cannot move a top-level view. The top-level view stays anchored in the layout window. You can reposition the location at which your top-level view will appear in the desktop when you run your application. You do that by setting the location in the Property Inspector Window for the top-level view. See *The PowerPlant Book* for details on positioning windows.

Nudging a pane

Sometimes you want to move a pane by one or two pixels. It can be difficult to be this precise dragging with the mouse.

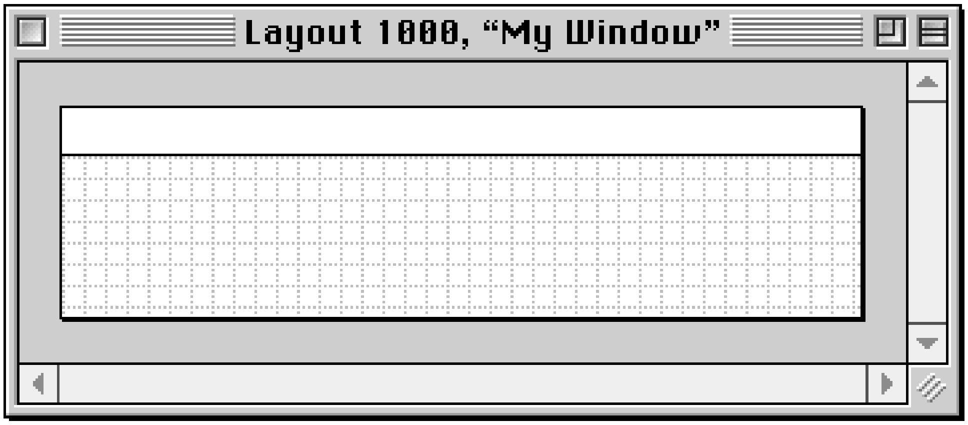
To nudge a pane, first select the pane. Then use the cursor control keys (the arrow keys) to move the pane in the desired direction: up, down, left, or right. Each time you type an arrow key, the pane moves by one pixel in the corresponding direction.

Using a grid

Constructor has a grid you can use to arrange panes. You can display the grid, snap to the grid, and modify the grid spacing.

To see the grid, choose **Show Grid** from the **Layout** menu. To conceal the grid, choose **Hide Grid**. Figure 4.9 shows a 10-pixel square grid in an empty PPob.

Figure 4.9 The Constructor grid on display



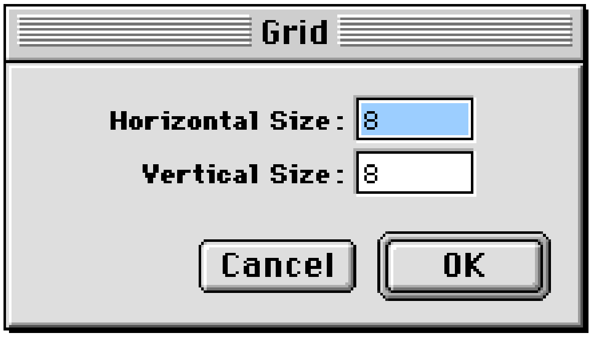
If you want all panes to align along grid lines, choose the **Snap To Grid** command from the **Layout** menu. To allow free movement, choose **Don't Snap To Grid**. These commands have no effect on existing panes. The snap to grid comes into play when you drag panes. If the snap feature is on, the drag outline snaps to the grid while you drag an item. The snap feature works whether or not the grid is visible.

TIP You can reverse the snap to grid setting when dragging panes. Hold down the Command key after you start dragging.

You can set the grid spacing (in pixels). Choose the **Edit Grid…** item in the **Layout** menu. The Grid dialog appears as shown in Figure

4.10. Enter the grid spacing you want. You can have different values for horizontal and vertical spacing.

Figure 4.10 The Grid dialog



Grid

Horizontal Size: 8

Vertical Size: 8

Aligning panes

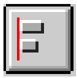
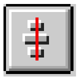
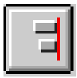
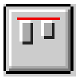
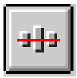
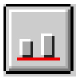
You can align selected panes in the layout very easily. Constructor has three mechanisms for aligning panes: menu commands, a palette, and a dialog. All three mechanisms have similar functionality for aligning panes.

First, select two or more panes you wish to align. Then issue the command for how you want to align the panes. You may:

- choose an alignment command from the Arrange menu
- click a button in the alignment palette
- open a dialog by choosing Arrange Objects… from the Arrange menu

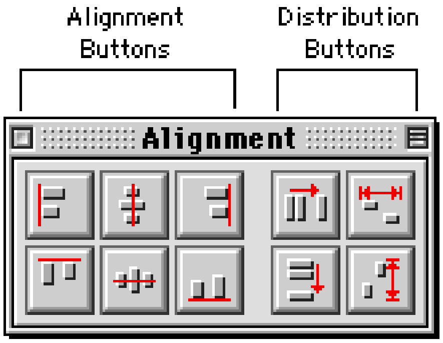
The alignment options, corresponding icons, and results are listed in Table 4.1. These correspond to menu commands in the **Arrange** menu, and buttons in the palette and the Arrange Objects dialog.

Table 4.1 Constructor pane alignment options

Option	Icon	Alignment Line
Align left edges		the left of the leftmost selected pane
Align horizontal centers		halfway between the left of the leftmost selected pane and the right of the rightmost selected pane
Align right edges		the right of the rightmost selected pane
Align top edges		the top of the topmost selected pane
Align vertical centers		halfway between the top of the topmost selected pane and the bottom of the bottommost selected pane
Align bottom edges		the bottom of the bottommost selected pane

To see the alignment palette, choose **Alignment Palette** from the **Window** . Figure 4.11 shows the alignment palette, with the six alignment buttons on the left side of the palette. The other buttons are for distributing panes. See “Distributing panes.”

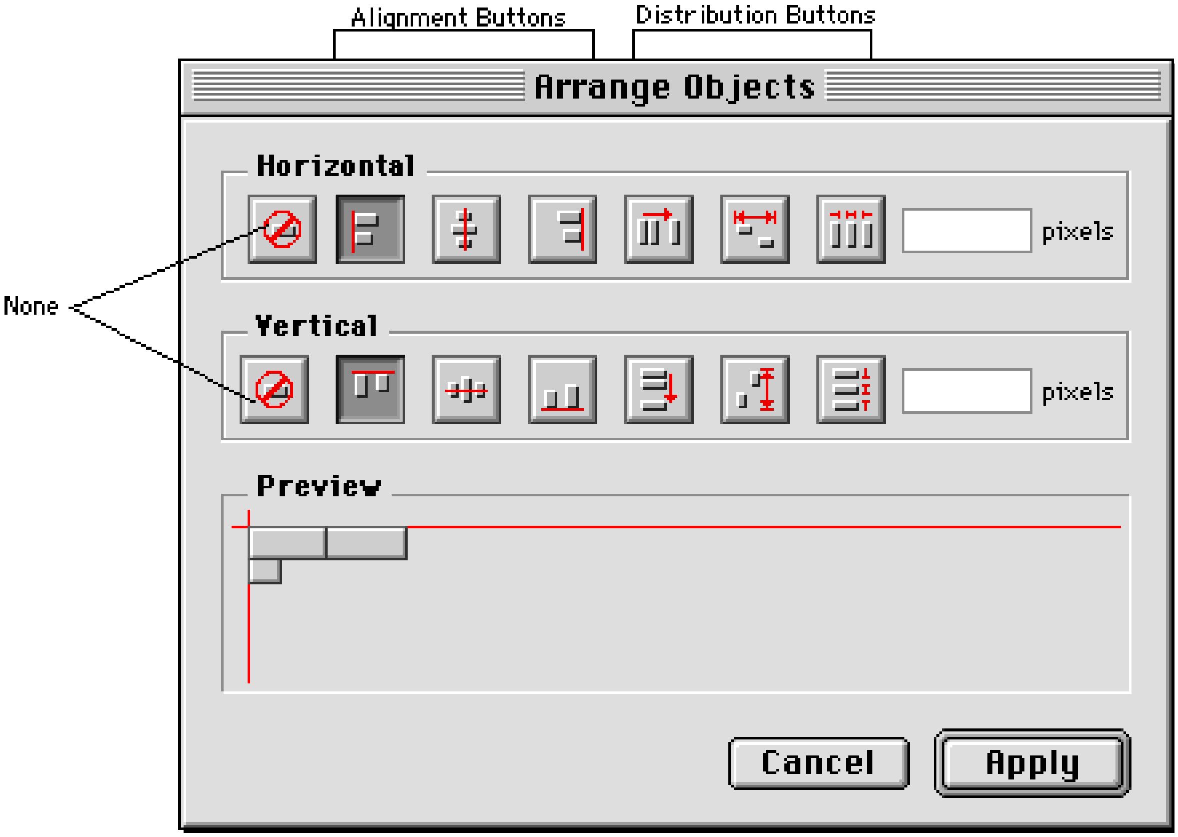
Figure 4.11 The Alignment palette



Simply click the appropriate button to obtain the desired alignment. The effect is immediate. Like most operations in Constructor, you can undo the action if you wish.

The third alternative for aligning panes in Constructor is the Arrange Objects dialog. When you choose **Arrange Objects…** from the **Arrange** menu, Constructor displays the dialog shown in Figure 4.12. This dialog gives you a few alignment options not available with the alignment palette.

Figure 4.12 The Arrange Objects dialog



The Arrange Objects dialog lets you set both a horizontal and vertical alignment simultaneously, and lets you preview the effect of a command. The preview in the illustration shows objects aligned along top left edges.

Simply click the desired horizontal and vertical alignment. When you have set the desired alignment option, click the Apply button. The dialog disappears and your command is applied to the selected panes. Constructor remembers the previous settings in the Arrange Objects dialog and restores them when you open the dialog again.

Distributing panes

Distributing panes means to spread them out evenly across a certain distance. Evenly means that after distribution, the gap between each pair of panes is the same.

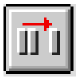
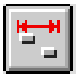
You can distribute selected panes in the layout very easily. Constructor has three mechanisms for distributing panes: menu commands, a palette, and a dialog. The dialog has additional features not available from menu commands or the alignment palette.

First, select two or more panes you wish to distribute. Then issue the command for how you want to distribute the panes. You may:

- choose a distribution command from the **Arrange** menu
- click a button in the alignment palette
- open a dialog by choosing **Arrange Objects…** from the **Arrange** menu

The distribution options, icons, and results are listed in Table 4.2. Some of these correspond to menu commands in the **Arrange** menu, and buttons in the alignment palette. The Arrange Objects dialog has all six options.

Table 4.2 Constructor pane distribution commands

Option	Icon	Range of spread
Spread horizontally		between the left and right selected panes
Spread horizontally in container		from the left edge to the right edge of the container that holds all the panes
Spread horizontally by specified gap		from the left selected pane, with the specified gap between panes
Spread vertically		between the top and bottom selected panes

Option	Icon	Range of spread
Spread vertically in container		from the top edge to the bottom edge of the container that holds all the panes
Spread vertically by specified gap		from the top selected pane, with the specified gap between panes

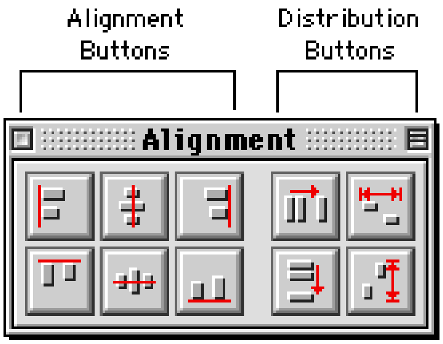
When distributed, panes maintain their original left to right or top to bottom ordering, based on the left or top edges respectively.

You cannot specify a gap using the alignment palette or menu commands. You can specify a gap in the Arrange Objects dialog.

When distributing in a container, you must pay attention to the containment hierarchy. If all panes are in the same container (such as a scrolling view, for example) then it is that container's bounds that serve as the range within which the panes are distributed. If panes are in multiple containers, the bounds of the container that holds all the selected panes controls distribution.

Figure 4.13 shows the alignment palette, with four distribution-related buttons on the right side of the palette. The other buttons are for aligning panes. See "Aligning panes."

Figure 4.13 The Alignment palette

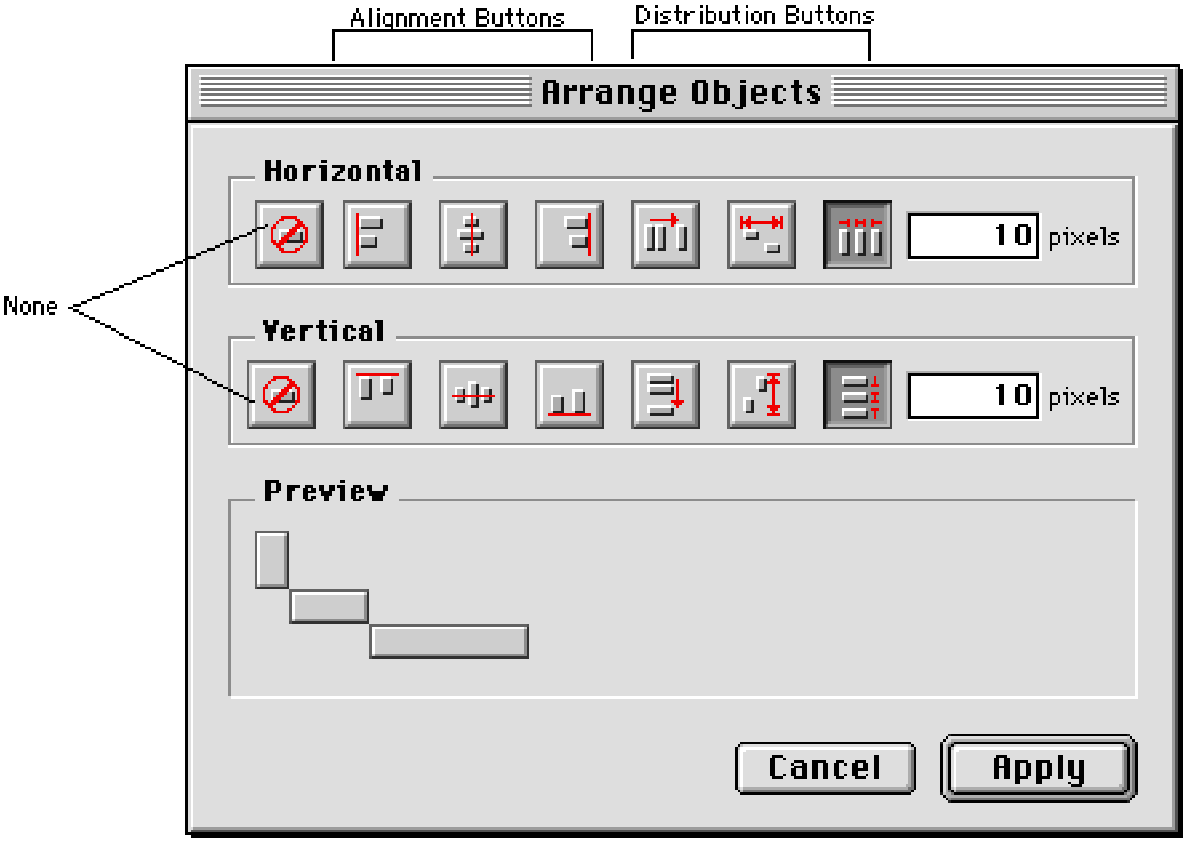


Simply click the appropriate button to obtain the desired distribution. The effect is immediate. Like most operations in Constructor, you can undo the action if you wish.

The third alternative for distributing panes in Constructor is the Arrange Objects dialog. When you choose **Arrange Objects…** from the **Arrange** menu, Constructor displays the dialog shown in Figure 4.14. This dialog gives you a few distribution options not available with the alignment palette.

The Arrange Objects dialog lets you set both a horizontal and vertical distribution simultaneously, and lets you preview the effect of a command. The preview in the illustration shows objects distributed with a 10-pixel gap both horizontally and vertically.

Figure 4.14 The Arrange Objects dialog



You may also provide a set gap or space to use between panes. To specify a gap, click the appropriate button. When you do, the *pixels* field associated with that button becomes enabled so you can enter the desired spacing.

When you have set the desired distribution options, click the Apply button. The dialog disappears and your command is applied to the selected panes. Constructor remembers the previous settings in the Arrange Objects dialog and restores them when the dialog opens.

Resizing a pane

To resize a pane or panes, click on one of the selection handles and drag it in any direction. You may also open the Property Inspector Window to change the width and height. You can change the size of a top-level view just like any other pane.

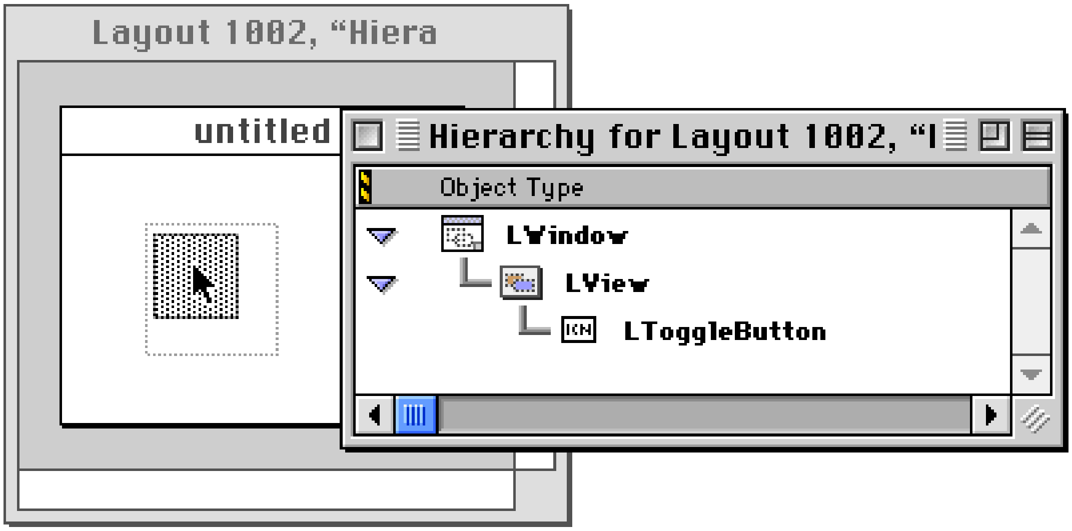
TIP Binding affects a pane both at runtime and in the layout editor. If binding is on for any given side of a pane, that side of the pane will remain the same distance from the edge of the enclosing view when you resize the view.

Arranging Panes Hierarchically

When you add a pane to a layout, if the drop point is inside a containing pane (a view pane of some type such as LScroller or LView), then Constructor automatically places the new pane inside the container, not only visually but hierarchically as well.

Figure 4.15 shows the result of dropping an LToggleButton pane inside an existing LView object. Notice in the hierarchy window how the button pane appears hierarchically inside the LView pane.

Figure 4.15 Dropping a pane in a container

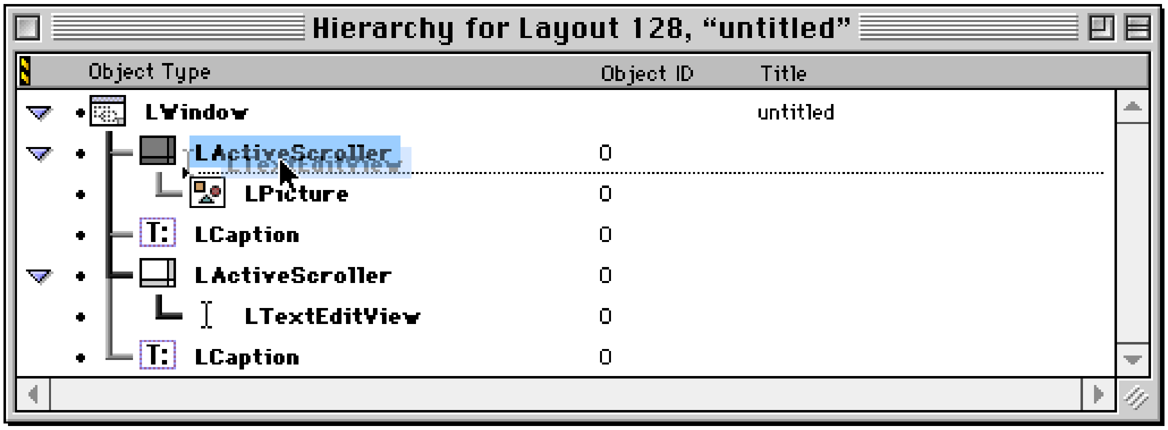


Notice that in the layout window you cannot tell whether the LToggleButton is actually contained in the view. They could simply overlap. The hierarchy window shows you the containment relationship. There is no way you can tell the hierarchical relationship of these two panes by looking at the layout window.

On the other hand, the hierarchy window is designed so that you can easily see and modify the pane hierarchy.

To change a pane's position in the containment hierarchy, select the pane in the hierarchy window, and drag it to a new position anywhere in the hierarchy. While you drag, a small focus bar and arrow indicate the current destination, as shown in Figure 4.16. The focus bar flashes to enhance visibility.

Figure 4.16 Changing hierarchy order



Hierarchy for Layout 128, “untitled”

Object Type	Object ID	Title
LWindow		untitled
LActiveScroller	0	
LPicture	0	
LCaption	0	
LActiveScroller	0	
LTextEditView	0	
LCaption	0	

In this illustration, the user is changing the selected object’s container from one view to another view. This does *not* change the object’s spatial position in the layout. However, because Constructor draws objects clipped to the containing view, the object may appear to have vanished from the layout view. To see the object, increase the size of the new container view to include the area of the object you moved.

You can reorganize multiple panes simultaneously by selecting and dragging them as a unit.

NOTE Dragging a container operates on the container and subpanes as a unit, rather than just the container. Constructor outlines the subpanes as well as the container, so you can tell what you are moving.

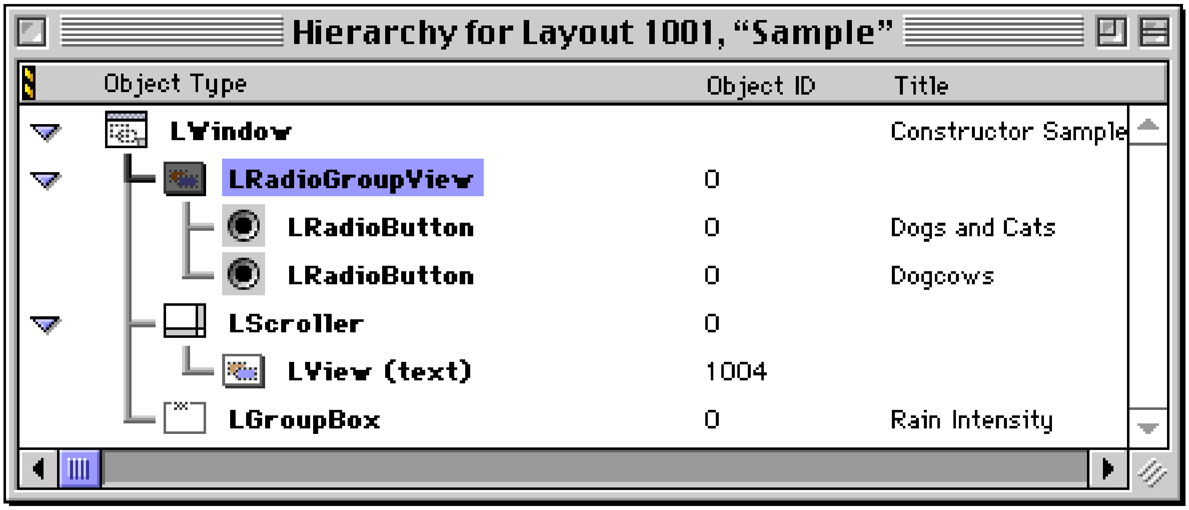
Working With Radio Groups

You create a radio group to manage a set of mutually-exclusive buttons, such as standard radio buttons, in which one and only one button must be on at one time. See *The PowerPlant Book* for information about how to use radio groups in PowerPlant code.

To create a radio group, drag an LRadioGroupView object from the Catalog Window and drop it onto its containing view, then drag and drop radio button objects into the LRadioGroupView. Constructor will automatically arrange the radio button objects within the view’s hierarchy so that they are in the LRadioGroupView object.

You can work in either the Layout Window or the Hierarchy Window when creating the group. You can see the radio group in the hierarchy window as shown in Figure 4.17.

Figure 4.17 A radio group in a hierarchy window



Object Type	Object ID	Title
LWindow		Constructor Sample
LRadioGroupView	0	
LRadioButton	0	Dogs and Cats
LRadioButton	0	Dogcows
LScroller	0	
LView (text)	1004	
LGroupBox	0	Rain Intensity

Alternatives to LRadioGroupView

Although these methods are supported, using a LRadioGroupView object is the recommended way to create a radio group.

Alternative ways to create radio group boxes:

- choose the **Make Radio Group** item from the **Arrange** menu
- drag an LRadioGroup object from the Catalog Window

Seeing and modifying radio group contents

To see which buttons are contained within a group, use the Hierarchy Window for the radio group, shown in Figure 4.17.

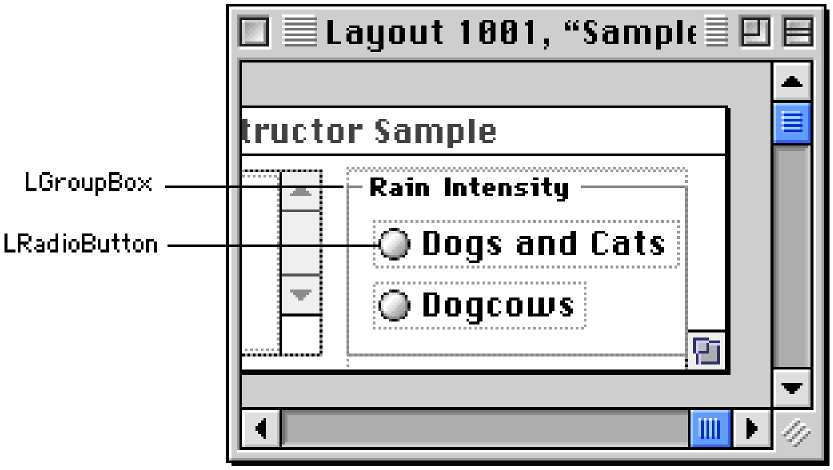
You can add radio buttons to and remove buttons from an existing group. To add a button, select an existing radio button entry in the Property Inspector Window and choose New Radio Button ID from the Edit menu. Then enter the ID number of the radio button you wish to add. To remove a button, simply select it and delete it.

If you create a radio group by dragging the object from the Catalog Window, the group will be empty (even if you have radio buttons selected). To add buttons to the group, use the technique described above.

Radio groups and group boxes

In many PPob resources, a radio group is visually placed inside an LGroupBox pane, as in Figure 4.18. The LGroupBox pane is simply a visual feature of the layout. It has nothing to do with radio groups.

Figure 4.18 Radio buttons inside an LGroupBox pane



You may also note from the illustration that Constructor does not show any button in the group as being on. PowerPlant takes care of this at runtime. You can specify which button should be on by default in the Property Inspector Window. See *The PowerPlant Book* for details.

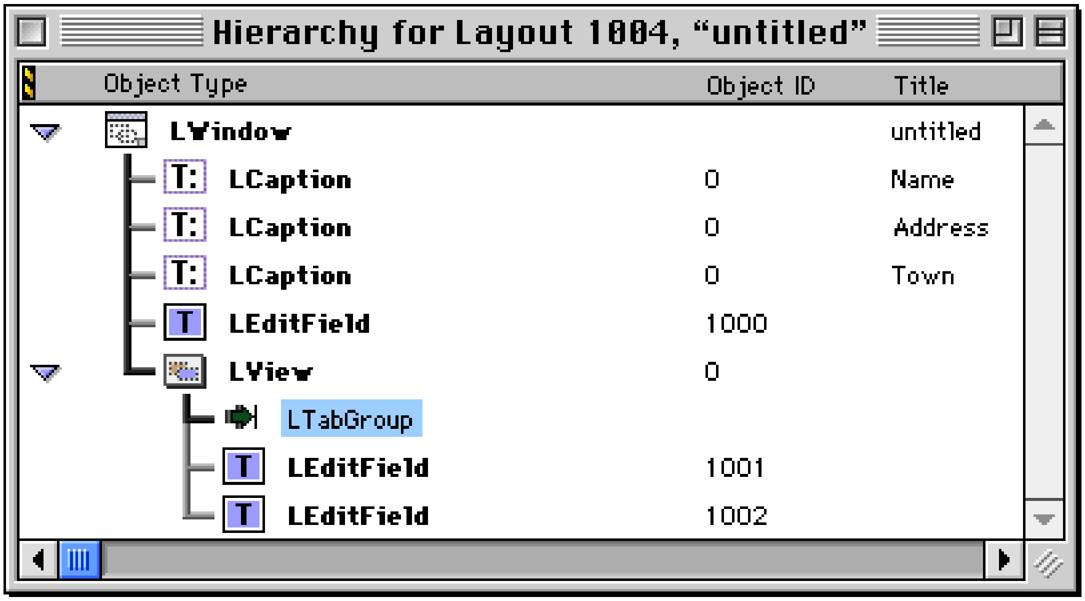
Working With Tab Groups

You create a tab group to manage a set of editable text fields when you want to allow the user to move from one field to the next by typing the Tab key. This is a common feature of dialog boxes that have editable text fields.

To create a tab group, choose the **Make Tab Group** item from the **Arrange** menu. The tab group object does not appear in the Layout

Window because is has no visual counterpart. It does appear in the Hierarchy Window as shown in Figure 4.19.

Figure 4.19 A tab group in a hierarchy window



Hierarchy for Layout 1004, "untitled"

Object Type	Object ID	Title
LWindow		untitled
LCaption	0	Name
LCaption	0	Address
LCaption	0	Town
LEditField	1000	
LView	0	
LTabGroup		
LEditField	1001	
LEditField	1002	

Drag the tab group object to any location in the hierarchy. A tab group automatically encompasses all objects at the same level in the visual hierarchy—its siblings. The only way to control membership within a tab group is to put both the desired member panes and the tab group inside some other visual container so they are all on the same level, and other objects are not. For example, in Figure 4.19 two edit fields are in a tab group, but one is not.

A tab group includes *all* panes at the same level, but it only has impact on panes that are also commanders, such as text-related panes like LEditField and LTextEdit.

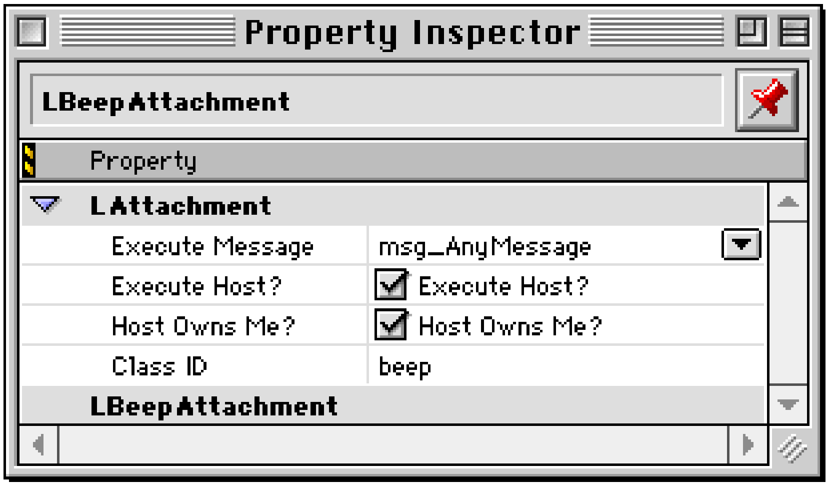
WARNING! The text field in the tab group gets the tab key first. Unless it passes the keypress to its supercommander, (typically the tab group object), the tab group will not work correctly. LTextEdit, LEditField, LTextEditView, and LEditText do this properly unless the key filter selected for the pane is *None*.

Working With Attachments

You can connect attachments to panes in your layout. Simply drag the correct attachment from the Catalog Window and drop it into either the Layout Window or Hierarchy Window at the appropriate spot. For example, drop the attachment onto the pane to which you want it attached.

To set the attachment's properties, use the Property Inspector Window show in Figure 4.20. See *The PowerPlant Book* chapter "Periodicals and Attachments" for information on using attachments.

Figure 4.20 Inspecting attachment properties



Execute Message	msg_AnyMessage
Execute Host?	☑ Execute Host?
Host Owns Me?	☑ Host Owns Me?
Class ID	beep

Setting Pane Properties

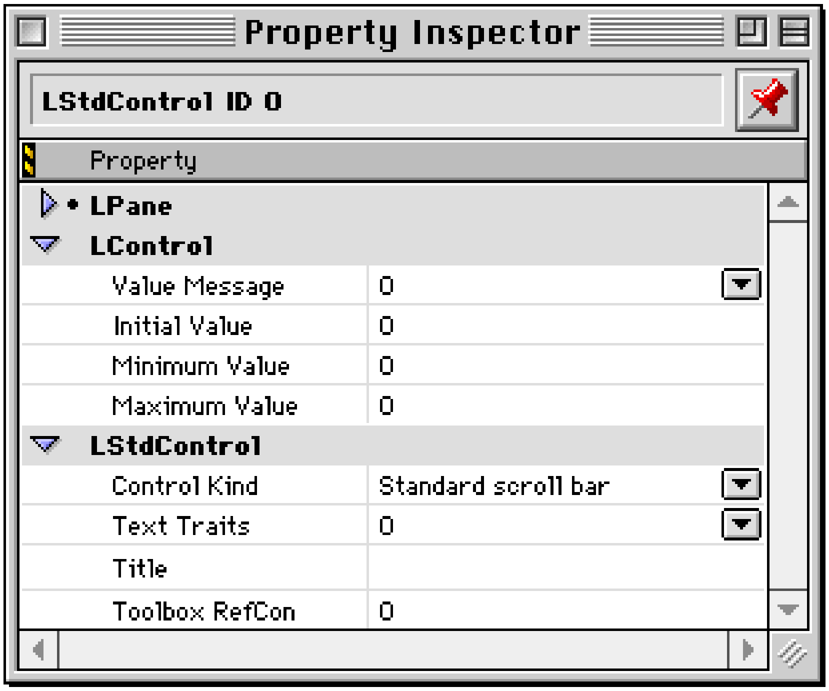
The Property Inspector Window allows you to see and modify an individual pane's data, such as location, size, content, and so forth. You open a property inspector using one of these methods:

- double-click a pane
- select a pane and press the Return key
- choose Property Inspector from the Window menu

You can perform these tasks in either the Layout Window or the Hierarchy Window. Once the Property Inspector is open, it displays the information for the currently selected pane. If multiple panes are selected, it displays information for the first pane selected.

Because each different class of pane has different properties, the precise information in the window depends upon the nature of the selected pane. For example, Figure 4.21 shows the property inspector for the LStdControl class.

Figure 4.21 The LStdControl properties



Property	
LPane	
LControl	
Value Message	0
Initial Value	0
Minimum Value	0
Maximum Value	0
LStdControl	
Control Kind	Standard scroll bar
Text Traits	0
Title	
Toolbox RefCon	0

The information in the Property Inspector is grouped according to the base class. For example, the LStdControl class derives from LControl ,which derives from LPane. For an LStdControl object, the Property Inspector displays data from each of the three classes, grouped by class.

The Property Inspector Window presents all the customizable properties for that particular kind of pane. When a particular property is irrelevant to a pane, the Property Inspector does not display that property. For example, LStdControl has a minimum and maximum value. LStdButton derives from LStdControl, but a button does not use the minimum or maximum values. So those properties are not available in the inspector window for LStdButton.

See “Property Inspector Window.”

For information on the effect or purpose of the various settings, see *The PowerPlant Book* and “Pane Properties Overview.”

Removing Panes

To remove a pane from a PPob resource, first select the pane or panes you want to remove. Then you may:

- press the Delete key
- drag the pane to the trash
- choose the Clear command from the Edit menu
- choose the Cut command from the Edit menu

Items dragged to the trash are deleted immediately. They cannot be recovered from the trash. However, a delete operation can be undone until you issue a new command.

Items that are cut from the PPob are placed on the clipboard (remain in memory available for pasting) until a subsequent cut or copy operation.

5

Editing Menu Bars and Menus

This chapter discusses Constructor's menu bar and menu editors.

Editing Menu Bars and Menus Overview

Menu bars and menus are closely intertwined, both in Constructor and in a real application. An MBAR resource is little more than a list of MENU resources to include in the menu bar at runtime. Constructor follows the same pattern. The menu bar editor displays a collection of menus and allows you to modify them. The two editors—menu bar and menu—are virtually identical.

This chapter discusses how to manage and modify the *contents* of menus. For information on how to create, copy, modify, open, and delete menu bar and menu resources as a whole, see "Managing Resources Overview."

The major sections in this chapter are:

- Menu Command Numbers—a few words on menu command numbers and their significance
- Windows for Menus—the Constructor interface for menu editing
- Editing a Menu Bar—how to edit a menu bar
- Editing Menu Items—how to create and edit menu items

Menu Command Numbers

PowerPlant adds menu command numbers to menu items. Each menu item has a unique command number. The command number strategy allows you to rearrange menus without having to change your menu dispatch code, as long as the command number moves along with the menu item.

Command numbers are stored in an Mcmd resource. The resource ID of the menu and associated Mcmd are the same. Constructor creates and maintains the required Mcmd resources automatically based on the data you provide in the menu editors.

NOTE Some menus cannot be described ahead of time. Font menus, for example, depend on the fonts installed on the machine. For these kinds of menus, PowerPlant uses a concept called the synthetic command number. See *The PowerPlant Book* chapter on commanders and menus for a thorough explanation.

Standard menu items like the **New** item in the **File** menu have standard command numbers in PowerPlant. In the Constructor menu editors, PowerPlant command numbers are displayed with their symbolic name rather than a numerical value. In the property inspector for menu items, the pop-up menu for command numbers lists all defined PowerPlant menu commands.

You are responsible for ensuring that the command number you assign to a menu item is handled properly in your code. In general, this means using predefined PowerPlant command numbers whenever possible. These values are declared in the `PP_Messages.h` file in the PowerPlant source files. PowerPlant reserves and uses command numbers –999 to 999 for the standard menu commands.

TIP Use the ppob file from the PowerPlant stationery project as a reference for standard menus and command numbers.

This chapter does not discuss the PowerPlant menu strategy, or how to use menu command numbers to best effect, any further. For information on PowerPlant’s use of menus and menu commands, see *The PowerPlant Book* chapter on commanders and menus.

Windows for Menus

The menu bar and the menu resource each have their own editor window. This section discusses both:

- Menu Bar Editor Window
- Menu Editor Window

In addition, you can use the Property Inspector Window to examine and set the properties of either a menu or a menu item. See "Setting Menu Properties" and "Setting Menu Item Properties."

See also "Property Inspector Window."

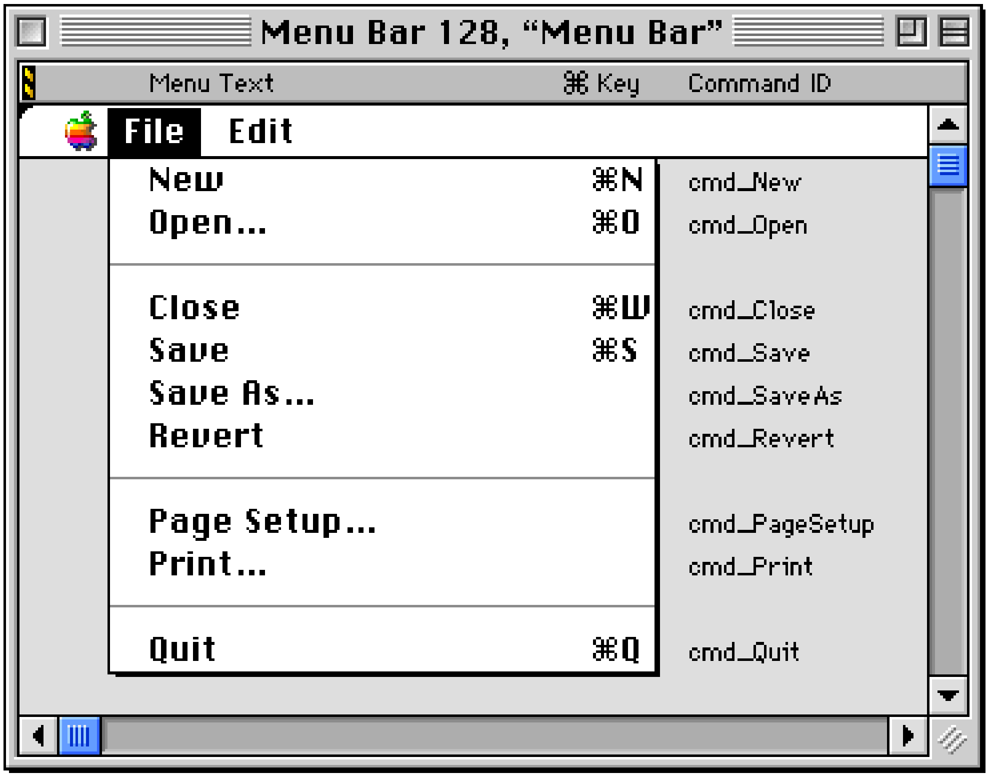
Menu Bar Editor Window

The menu bar editor displays a collection of menu resources contained in the menu bar resource. To open the menu bar editor window, you can:

- double click a menu bar resource in the project window
- select the menu bar resource and press the Return key
- select the menu bar resource and choose **Edit Resource** from the **Edit** menu

When you do, the window shown in Figure 5.1 appears. This illustration shows a typical menu bar resource.

Figure 5.1 The menu bar editor



The grey bar across the top of the window contains the column heads for the data displayed in the window. You see menu text, the command key associated with each menu item, and the PowerPlant menu command number for each menu item.

Like the layout window, the menu bar editor displays a simulated desktop. The white bar that contains menu titles is analogous to an actual menu bar on the Mac OS desktop, right down to the curved corner at the top left of the title bar.

In the grey area below the title bar, Constructor displays the actual menu items in each menu. The menu items appear just as they would if you pulled the menu down in a running application. The Constructor menu bar editor is WYSIWYG—what you see is what you get—with two exceptions. Constructor does not display submenus interactively, although it does display the triangle indicator for a hierarchical menu item. Constructor also does not display an icon if the menu item has an icon attached.

A menu bar is really just a collection of menu resources. The Constructor menu bar editor allows you to see and edit not only which menus are in the menu bar, but the contents of each menu as well.

Menu Editor Window

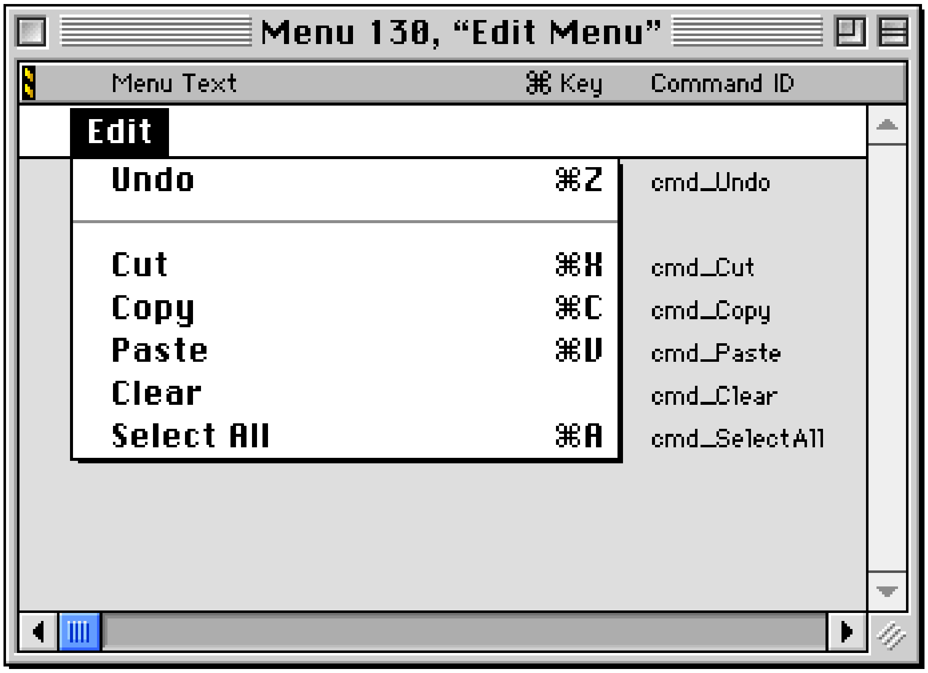
The menu editor window is virtually identical to the menu bar editor. The only difference is that the menu editor displays a single menu. The menu bar editor can contain several menus.

To open a menu editor window, you:

- double click a menu resource in the project window
- select the menu resource and press the Return key
- select the menu resource and choose **Edit Resource** from the **Edit** menu

When you do, the window shown in Figure 5.2 appears.

Figure 5.2 Menu editor window



Menu Text	⌘ Key	Command ID
Edit		
Undo	⌘Z	cmd_Undo
Cut	⌘X	cmd_Cut
Copy	⌘C	cmd_Copy
Paste	⌘V	cmd_Paste
Clear		cmd_Clear
Select All	⌘A	cmd_SelectAll

Editing a Menu Bar

This section discusses the tasks you perform to manage and edit a menu bar resource. All of these tasks are performed within the menu bar editor. The topics discusses in this section are:

- Making a New Menu—creating a new menu in the menu bar
- Adding an Existing Menu—adding an existing MENU resource to the menu bar
- Modifying a Menu—changing a menu's title and other characteristics
- Setting Menu Properties—using the Property Inspector Window with menus
- Changing Menu Order—changing the order of appearance of individual menus in the menu bar
- Removing a Menu—deleting a menu from the menu bar

For related topics, see:

- "Creating a Menu Bar Resource"—how to create a menu bar resource
- "Menu Bar Editor Window"—how to open the menu bar editor
- "Editing Menu Items"—how to edit particular menu items

Making a New Menu

To make a new menu, choose the **New Menu** item in the **Edit** menu. When you do, a new menu title appears in the menu bar editor to the right of the currently selected menu title. If there is no selection, the new menu is added to the end of the menu bar.

If the new menu is one of the first three menus added to the menu bar, Constructor automatically assigns the correct menu title: Apple icon, File, or Edit. For subsequent menus, the initial menu title is "untitled."

When you create a new menu, the menu title is placed in editing mode so you can modify the text. To end editing mode, click in an empty place in the editor window, or press the Tab or Return keys.

When you create a new menu, Constructor also creates the required MENU resource. The new resource will appear in the Menus section of the project window. Constructor assigns consecutive resource ID numbers to new MENU resources.

Creating a new menu does not add items to the menu. For more information on this topic, see "Making a New Menu Item."

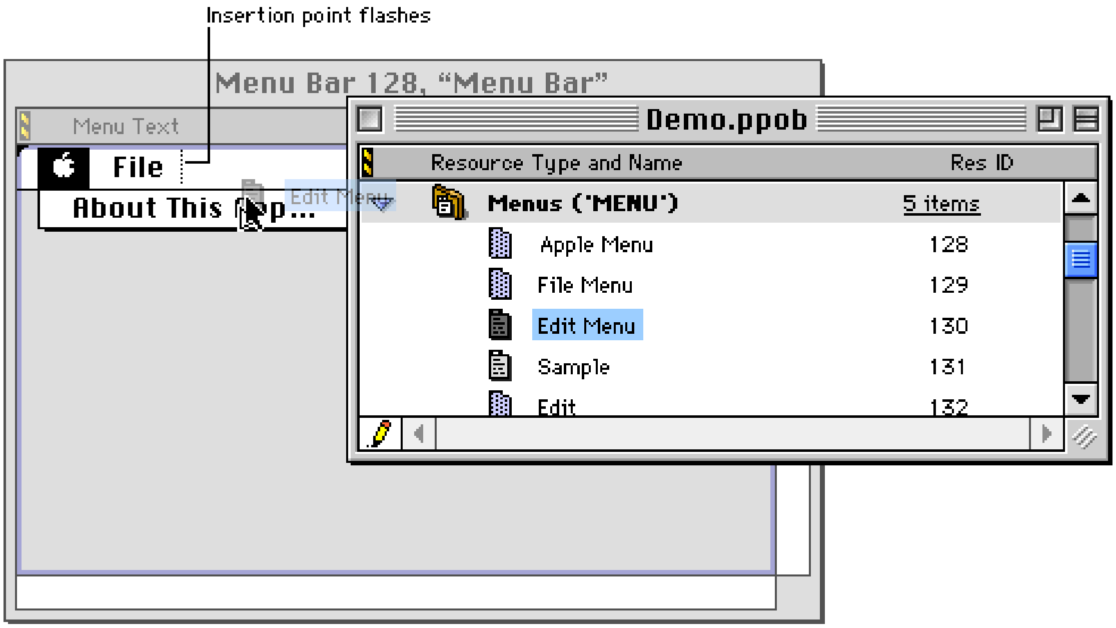
Adding an Existing Menu

If you have an existing menu resource that is not already in the menu bar, you can add it. Drag the desired menu resource from the project window and drop it into the menu bar editor.

In the project window, select the menu resource(s) you want to add to the menu bar. It must not be in the menu bar already. You can tell which menu resources are not in the menu bar, because the icon to the left of the resource is clear and not grey. In Figure 5.3, look at the difference in the icons for the **Edit** menu and the other two menu resources. The icons for menus that are already in the menu bar are grey because they are open.

Drag the selected menu resource(s) from the project window into the menu bar editor area, as shown in Figure 5.3. As you do, a small vertical line will flash in the menu bar title area indicating the insertion point for the menu you are adding. Drop the menu where you want it to appear in the menu bar.

Figure 5.3 Adding an existing menu



Resource Type and Name	Res ID
Menus ('MENU')	5 items
Apple Menu	128
File Menu	129
Edit Menu	130
Sample	131
Edit	132

Notice that you don’t have to drop the menu in the actual title area. You can drop it anywhere in the editor window. The insertion point will track your mouse movements horizontally to show you where the menu will appear in the menu bar.

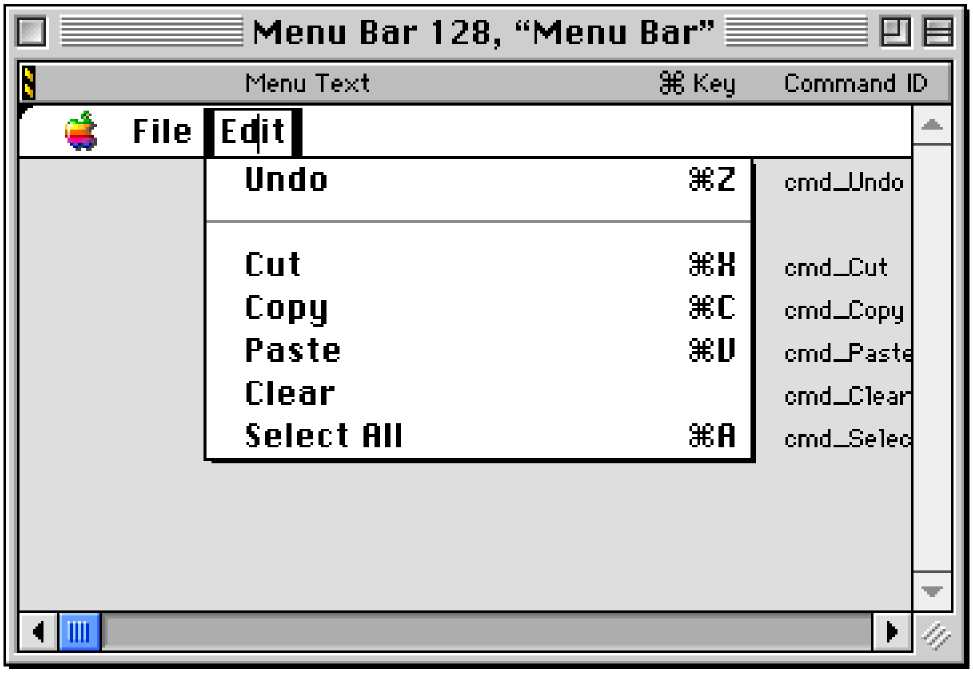
TIP If you select multiple resources, all must be absent from the menu bar or none will be added when you drop the menus.

Modifying a Menu

You may change a menu title, the menu resource ID, and the MDEF (menu definition) resource used to draw the menu.

To change the title, you can put the text into edit mode. Then you can modify the text in place, as shown in Figure 5.4. See “Text Editing” for more information.

Figure 5.4 Changing a menu title



To change the menu resource ID and the MDEF, open the Setting Menu Properties by double-clicking the menu title. You can change the menu title text in the Property Inspector Window as well. See Setting Menu Properties.

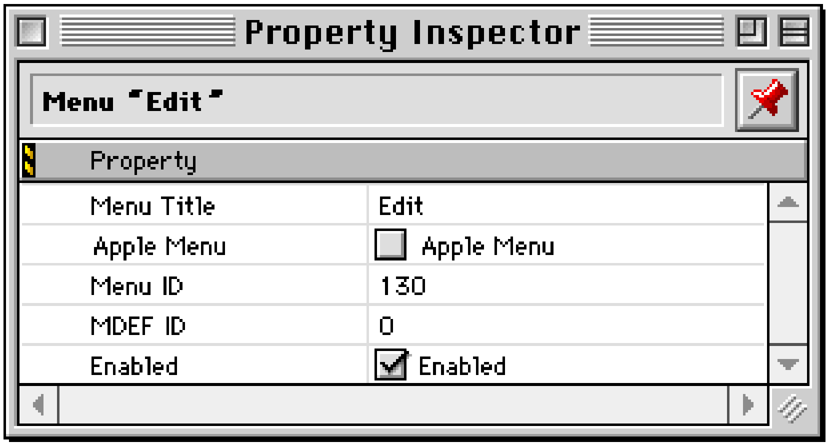
Setting Menu Properties

To modify menu properties, use the Property Inspector Window. For menus, the window displays data that applies to an entire menu. To open this window you:

- double click a menu title in the menu bar editor window
- select the menu title and press the Return key
- select the menu item and open the Property Inspector Window

When you do, the Property Inspector Window displays that menu's properties, as shown in Figure 5.5.

Figure 5.5 Menu property inspector



Menu "Edit"

Property	
Menu Title	Edit
Apple Menu	□ Apple Menu
Menu ID	130
MDEF ID	0
Enabled	☑ Enabled

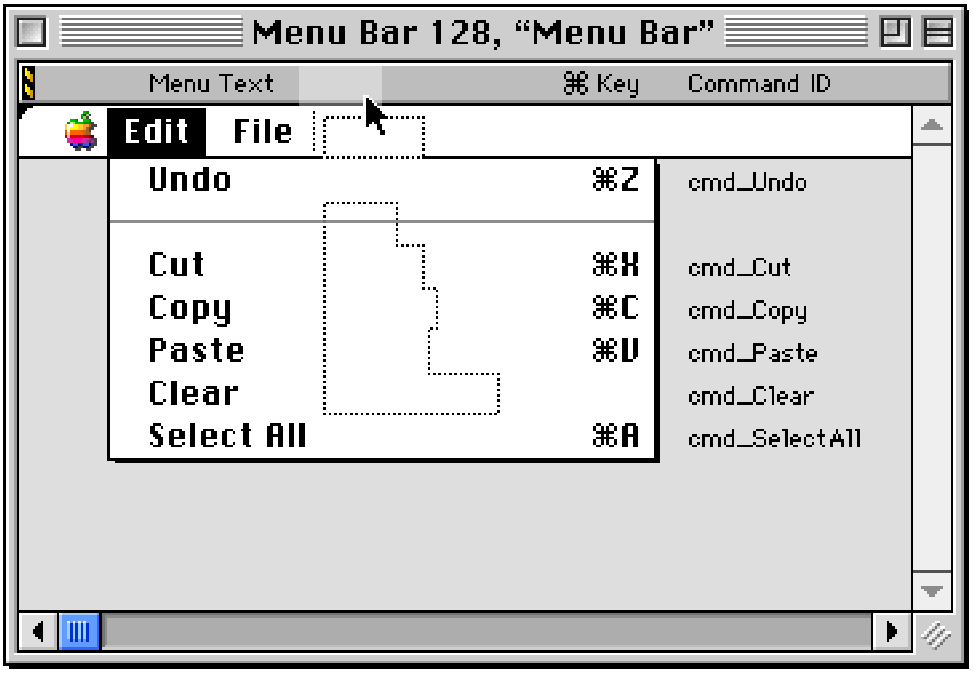
You can modify the menu title, menu ID number, and the MDEF resource ID number. Use the *Apple Menu* check box to set the menu title to be the Apple icon.

WARNING! The Mac OS Toolbox uses the Menu ID value. Constructor automatically sets the menu ID to the resource ID number. Because most programmers use these two values interchangeably, they should be the same. Don't change the Menu ID unless you absolutely need to, and you understand the consequences. Constructor will reset the Menu ID if you change the resource ID.

Changing Menu Order

To change the order of menus in the menu bar, simply drag a menu to a new location. A flashing insertion point indicates where in the menu bar the menu will appear when you drop it. Figure 5.6 illustrates the process of moving the **Edit** menu to the right of the **File** menu.

Figure 5.6 Moving a menu



Removing a Menu

To remove a menu from the menu bar, first select the menu. Then you can:

- press the Delete key
- choose **Cut** or **Clear** from the **Edit** menu
- drag the menu to the trash

TIP When you click the menu title to select it, it is likely to go into edit mode. Simply press the Return or Tab keys, or click in an empty part of the window to turn off edit mode.

Editing Menu Items

This section discusses the tasks you perform to manage and edit the items that make up a menu. For information on modifying the menu title, or information related to the menu as a whole, see “Editing a Menu Bar.”

All of the tasks discussed in this section may be performed within either the menu bar editor or the menu editor. The functionality of the menu bar and menu editors is virtually identical. The menu bar editor is simply a means of accessing a series of menus in the same window. The menu editor displays one menu.

As a practical matter, the menu bar editor is a lot more useful, because you can work with several menus simultaneously. However, it does not display submenus. Use the menu editor for submenus and other menus not in the menu bar, such as pop-up menus.

Whenever you move a menu item, Constructor takes care of managing and modifying the associated Mcmd resources automatically.

The topics discussed in this section are:

- Selecting a Menu Item—how to select menu items and control edit mode
- Making a New Menu Item—adding a new menu item
- Making a Menu Separator Item—adding a separator
- Modifying a Menu Item—changing item text, and setting other properties for an individual menu item
- Setting Menu Item Properties—using the Property Inspector Window with menu items
- Moving a Menu Item—shuffling menu items
- Adding a Submenu—creating a hierarchical menu item
- Removing a Menu Item—deleting a menu item

For information on related topics, see also:

- “Creating a Menu Resource”—how to create a menu resource
- “Menu Bar Editor Window”—how to open the menu bar editor
- “Menu Editor Window”—how to open the menu editor

Selecting a Menu Item

Working with menu items usually requires that you select one or more items. When you select an item, you may or may not want it to go into edit mode for in-place text editing.

There are four ways you can select menu items.

1. **Arrow keys**. Use the up or down arrow key to select the previous or next item in the menu. Press the shift key to extend the selection.
2. **Click text**. Click text in a menu item. If you move the cursor off the item or wait one second, the item will go into edit mode for in-place editing. You can click the item text, the Command-key character or the command number to put them into edit mode.
3. **Click the menu line**. Click the white space to the left or right of a menu item. This selects the item without going into edit mode. Press the shift key to extend the selection.
4. **Selection marquee**. Selects any intersecting item without going into edit mode.

You can also choose **Select All** from the **Edit** menu to select all the items in a menu.

TIP Use the left or right arrow key to change menus. Although you can select multiple menu *items*, you cannot select multiple menus.

You can get into and out of edit mode at will.

Getting into edit mode Click on the text and move the mouse off the text (or wait a second). Alternatively, you can also select a single item and press the Tab key to enter edit mode.

Getting out of edit mode Press the Return key. The item remains selected but edit mode is turned off.

Making a New Menu Item

To add a new menu item to a menu, choose **New Menu Item** from the **Edit** menu. The new item is added after any existing selected menu item, or at the end of the menu if there is no selected item. New items are named “untitled item,” have no command key equivalent, and a menu command number of zero. Make sure you modify the command number appropriately.

You can copy, paste, and duplicate selected menu items. A pasted item appears after any selected item, or at the end of the menu if there is no selected item. Duplicated items appear immediately after the original. If an item is in edit mode, editing operations affect the item text, not the item as a whole.

If you have two or more menu editor windows open simultaneously, you can drag a menu item from one window to another to create a copy.

Making a Menu Separator Item

You can make a separator item using any of these techniques:

- Choose **New Separator Item** in the **Edit** menu.
- Make a new menu item. When the item appears in edit mode, make the item text a single hyphen. Press Return when you have made this change, and the separator appears.
- Open the Setting Menu Item Properties for an item and turn on the *Separator Item* check box.

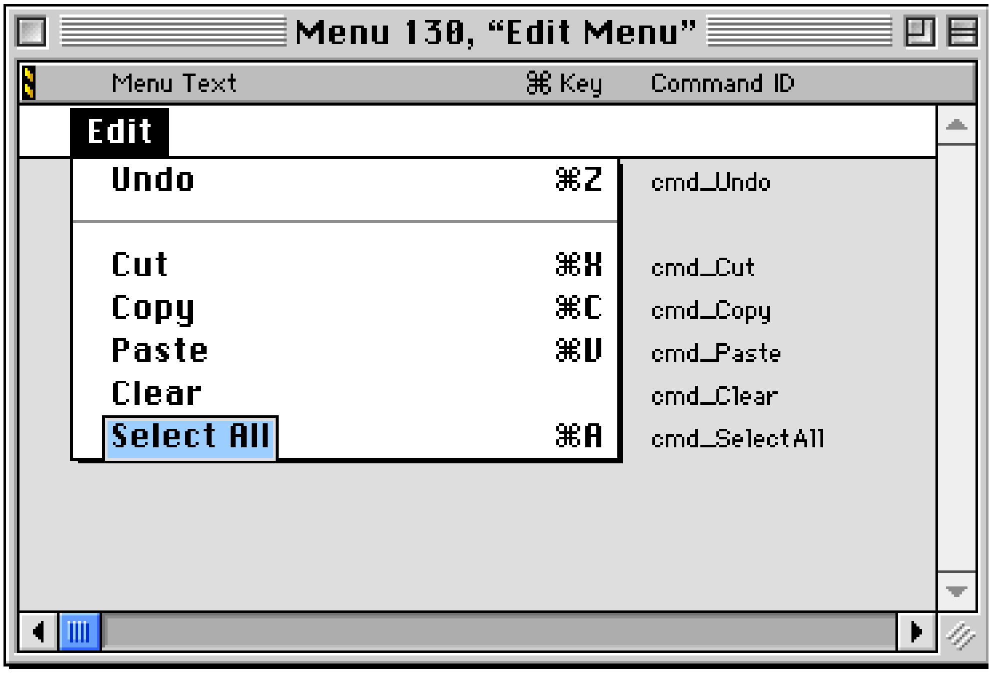
You cannot put a separator item into edit mode. To convert a separator back into a regular menu item, use the Property Inspector Window and turn off the *Separator Line* check box. You can then edit the menu text normally.

You can also copy and paste, duplicate, or copy a separator item by dragging.

Modifying a Menu Item

The editor window displays the menu item text, the command key equivalent, and the command number. These items may all be edited in place. Select the item and put it into edit mode. Then edit the item. For information on selecting an item and entering edit mode, see “Selecting a Menu Item.”

Figure 5.7 Editing a menu item in place



Menu Text	⌘ Key	Command ID
Undo	⌘Z	cmd_Undo
Cut	⌘X	cmd_Cut
Copy	⌘C	cmd_Copy
Paste	⌘V	cmd_Paste
Clear		cmd_Clear
Select All	⌘A	cmd_SelectAll

When editing a menu item in place, the Tab key advances you to the next part of the item. You can move from item text, to command key, to command number with the Tab key.

For command numbers, you can enter a numerical value, or a four-character text string. Constructor automatically recognizes non-numerical entries and displays them as text (inside single quote marks). To enter a series of numerals as text (for example, you want the command to be ‘1234’ rather than the number 1234) use single quote marks around the numerals as you enter them.

Make sure each menu item has a unique menu command number. You will encounter unpredictable results if duplicate command numbers occur.

Use the Property Inspector Window to see and modify other item-related properties. You can open the property inspector by double-clicking a menu item. In that window you can set not only the item text, command key equivalent, and command number, but a marking character, text style, an icon, and much more.

See “Setting Menu Item Properties” for a detailed discussion.

Setting Menu Item Properties

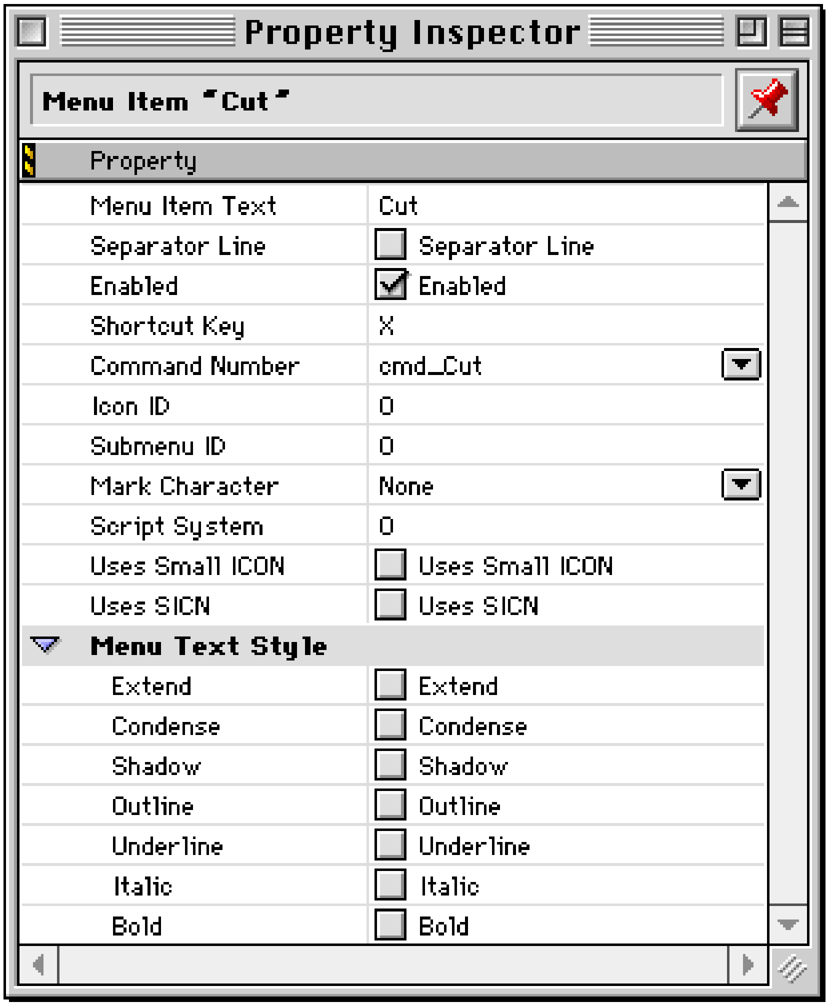
To modify menu item properties, use the Property Inspector Window. The menu item property inspector displays data that applies to a particular menu item. To open this window you:

- double click a menu item in either the menu bar editor or the menu editor
- select the menu item and press the Return key
- select the menu item and open the Property Inspector Window

When you do, the Property Inspector Window displays that item's properties, as shown in Figure 5.8. You can modify many attributes of a menu item. Following the illustration, Table 5.1 list each item in the window and the menu attribute it specifies.

TIP For menu command numbers, the pop-up menu lists all standard PowerPlant command numbers by symbolic name.

Figure 5.8 Menu item property inspector



Menu Item "Cut"

Property	
Menu Item Text	Cut
Separator Line	□ Separator Line
Enabled	☑ Enabled
Shortcut Key	X
Command Number	cmd_Cut
Icon ID	0
Submenu ID	0
Mark Character	None
Script System	0
Uses Small ICON	□ Uses Small ICON
Uses SICN	□ Uses SICN
Menu Text Style	
Extend	□ Extend
Condense	□ Condense
Shadow	□ Shadow
Outline	□ Outline
Underline	□ Underline
Italic	□ Italic
Bold	□ Bold

Table 5.1 Menu item properties

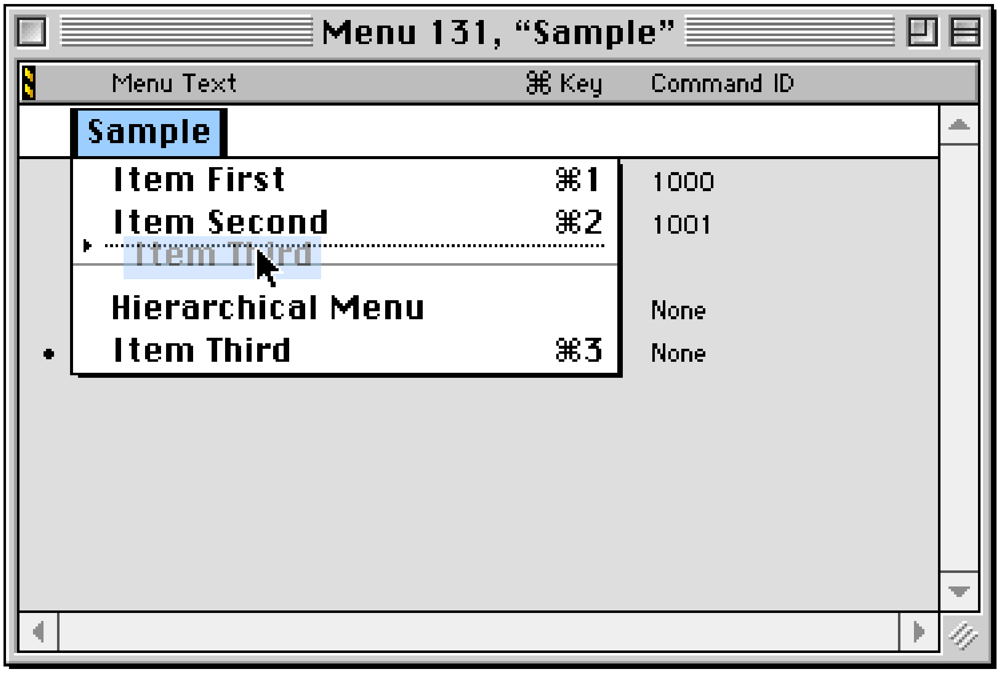
Item	Specifies
Menu Item Text	menu title
Separator Line	whether this item is a separator bar
Enabled	menu item is enabled by default
Shortcut Key	command-key equivalent
Command Number	menu command number for this item, a 32-bit value

Item	Specifies
Icon ID	resource ID for an icon to appear to the left of the menu item
Submenu ID	menu ID for a submenu if this is a hierarchical menu item, an 8-bit value
Mark Character	character used to mark this item
Script System	number of desired script system; 0 uses Roman, Chicago 12 on US software
Uses small ICON	use regular icon and shrink it
Use SICN	use a small icon
Menu Text Style	options for appearance of menu item text

Moving a Menu Item

To reposition a menu item in a menu, simply drag it to a new location in the menu. Figure 5.9 illustrates the process.

Figure 5.9 Moving a menu item



Menu Text	⌘ Key	Command ID
Item First	⌘1	1000
Item Second	⌘2	1001
Hierarchical Menu		None
Item Third	⌘3	None

There is a small arrow and a flashing line that indicate the insertion point in the menu.

You can change the order of menu items using cut and paste. A pasted item is placed in the list after the currently selected item, or at the end of the menu if there is no selected item.

You can also drag a menu item from one menu to another in the menu bar editor. Drag the item to the menu bar. The menu bar editor dynamically switches menus as you drag the item across the menu bar. When you reach the proper destination menu, drag down the menu to drop the menu item in the correct location.

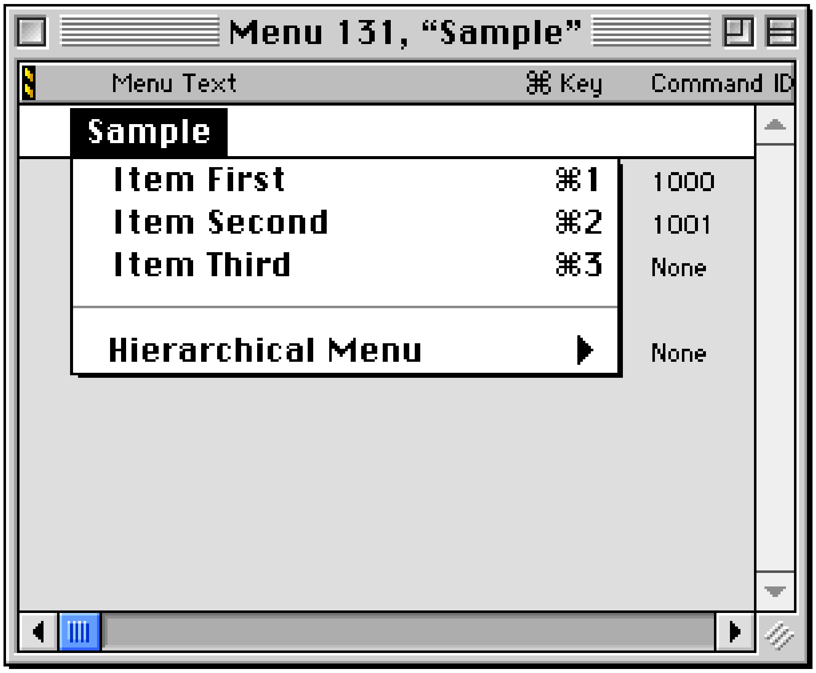
Adding a Submenu

To make a menu item a hierarchical menu item, you must attach a submenu to it. To do so, open the Setting Menu Item Properties. Then enter a non-zero value in the Submenu ID field.

The Submenu ID must be a value from 1 to 255 for an application submenu. Of course, you must also create a menu with the correct ID, or the hierarchical menu item will not work at runtime.

When you set the Submenu ID field, the standard hierarchical triangle mark appears to the right of the menu item in the menu editor window, as shown in the last menu item in Figure 5.10.

Figure 5.10 A hierarchical menu



Menu 131, "Sample"

Menu Text	⌘ Key	Command ID
Item First	⌘1	1000
Item Second	⌘2	1001
Item Third	⌘3	None
Hierarchical Menu ▶		None

Removing a Menu Item

To remove one or more menu items from a menu, first select the item or items. Then you can:

- press the Delete key
- choose **Cut** or **Clear** from the **Edit** menu
- drag the items to the trash

The items disappear from the menu.

6

Editing Bitmaps

This chapter discusses Constructor’s bitmap editor. You use the bitmap editor to create and modify bitmaps inside a variety of resources.

Editing Bitmaps Overview

Many different kinds of resources store information in bitmaps. The ability to interactively edit bitmap data in a graphical user interface greatly increases productivity and efficiency. You can see what you’re doing while you do it.

Constructor has a simple, powerful, and intuitive bitmap editor. Constructor uses a single bitmap editor to modify bitmaps in a variety of resource types, including:

- icon suites—‘icl8’, ‘icl4’, ‘ICN#’, ‘ics8’, ‘ics4’, ‘ics#’, ‘icm8’, ‘icm4’, and ‘icm#’ representing large icons, small icons, and icon masks in 8-bit color, 4-bit color, and black and white
- ‘ICON’—black and white icon
- ‘cicn’—color icon
- ‘PICT’—bitmaps, not QuickDraw drawing commands
- ‘PAT ’—black and white pattern
- ‘ppat’—color pattern
- ‘CURS’—black and white cursor
- ‘crsr’—color cursor

Each type of resource has a separate section in the Constructor project window. However, when you open up any of these resources Constructor opens the same bitmap editor.

This chapter discusses how to manage and modify the *contents* of a bitmap resources. For information on how to create, copy, open, modify, and delete a bitmap resource as a whole, see “Managing Resources Overview.”

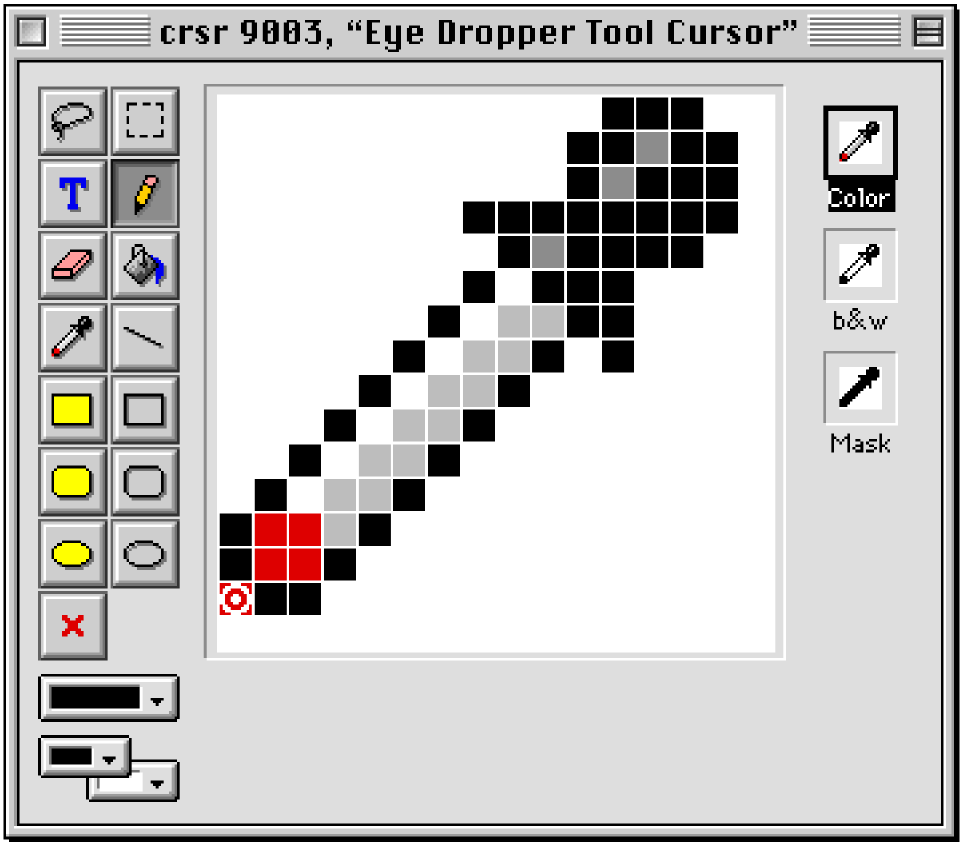
The topics in this chapter are:

- The Bitmap Editor—the various parts of the editor window
- Bitmap Editor Tools—each individual editor tool and how to use that tool
- Editing a Bitmap—tips and ideas on how to use the bitmap editor

The Bitmap Editor

Figure 6.1 illustrates the bitmap editor with all its features.

Figure 6.1 The Constructor bitmap editor



The same bitmap editor works for all resources containing bitmaps. However, certain features of the editor are only applicable to some resources.

The Constructor bitmap editor has the following components:

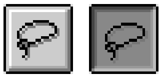
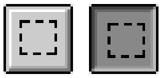
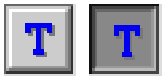
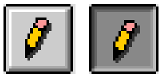
- Tool Palette—tools to control drawing operations
- Editing View—where you draw, usually in fatbits
- Sample Views—actual-size views of your work

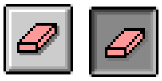
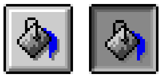
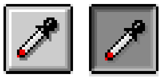
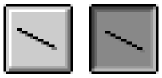
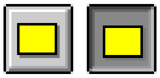
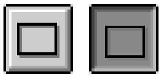
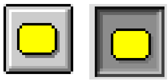
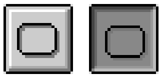
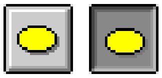
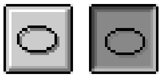
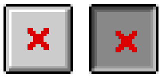
Tool Palette

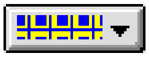
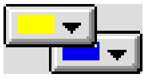
The tool palette is a set of push buttons representing various operations. Figure 6.1 shows the bitmap editor tools on the left of the editor window.

To use a particular tool, click the tool to select it. When you do, the button representing the tool changes appearance and looks pushed. Table 6.1 lists each tool, its icons (both pushed and unpushed), and the tool’s purpose. For detailed information on each tool, see that tool’s individual topic in “Bitmap Editor Tools.”

Table 6.1 The bitmap editor tools

Tool	Icons	Purpose
Lasso Tool		select an irregular area
Marquee Tool		select a rectangular area
Text Tool		add text to a bitmap
Pencil Tool		draw pixel by pixel, usually with the foreground color

Tool	Icons	Purpose
Eraser Tool		erase pixels with the background color
Paint Bucket Tool		fill an area of contiguous pixels of one color with current pattern
Dropper Tool		set foreground or background color (affects pattern color)
Line Tool		draw a line with the foreground color
Filled Rect Tool		draw a rectangle filled with the current pattern
Empty Rect Tool		draw a rectangle outlined with the foreground color
Filled Rounded Rect Tool		draw a rounded rectangle filled with the current pattern
Empty Rounded Rect Tool		draw a rounded rectangle outlined with the foreground color
Filled Oval Tool		draw an oval filled with the current pattern
Empty Oval Tool		draw an oval outlined with the foreground color
Hot Spot Tool		mark the hot spot for a cursor (shown in cursor resources only)

Tool	Icons	Purpose
Pattern Tool		set the fill pattern, based on foreground and background colors
Color Tool		set the foreground or background color

Editing View

The Editing View is the central part of the Constructor bitmap editor window, as shown in Figure 6.1. You modify the contents of the Editing View to modify the underlying bitmap. This works regardless of the kind of resource in which that bitmap appears.

The contents of the Editing View are usually shown in fat bits. This is a magnified image of the bitmap where a block of pixels in the Editing View represents a single pixel in the actual view. Magnification of bitmaps is automatic and depends upon the relationship of the size of the underlying bitmap to the dimensions of the Editing View.

To edit the contents of a bitmap, you first select a tool in the Tool Palette. Then you move the cursor to the Editing View. When the cursor is inside the Editing View, the cursor changes to reflect the currently selected tool. You can then click and drag to modify the pixels in the Editing View. For details on each individual tool and how to use it, see “Bitmap Editor Tools.”

As you make changes, the Sample Views update to reflect the new bitmap.

The Editing View supports the usual mechanisms for transferring data between windows and between applications. You can copy, cut, and paste bitmaps into the Editing View. You may also drop a bitmap into the Editing View.

You may drag from another Constructor window, or from an application that supports drag and drop (such as the Finder desktop or the Scrapbook).

When you drop an item on the Editing View, it remains at its original scale. It is not resized to fit. If the original bitmap is larger than the bitmap in the Editing View, the original bitmap will be cropped.

Sample Views

The Sample Views are on the right side of the bitmap editor window, as shown in Figure 6.1.

The actual Sample Views displayed in the editor window vary depending upon the nature of the resource. Figure 6.1 shows the ‘crsr’ Sample Views. Figure 6.2 shows the Sample Views for an icon suite. Other resources have different Sample Views. Larger PICT images have no sample view, because the Editing View shows the bitmap at actual size.

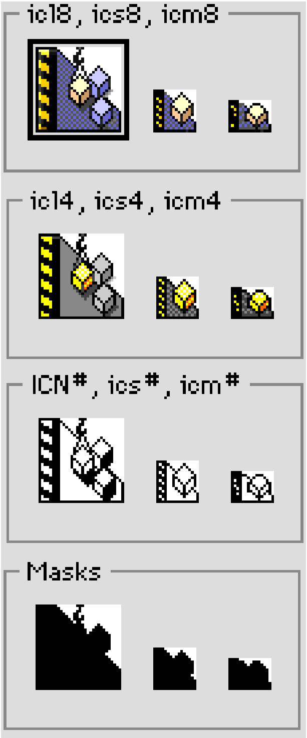
Each sample view represents either a different resource or a different portion of a resource. The Editing View displays the data for the currently selected sample view. The selected sample view has a heavy black border around it.

To switch to a different sample view, click the desired sample view. This selects the view and displays the data in the Editing View.

You can use drag and drop with any sample view. The operation can be between Sample Views in the same window, between Constructor bitmap editor windows, or between applications.

If you drop a bitmap from another source on a sample view, the bitmap is scaled to fit. If you drop a bitmap with more colors onto a sample view that displays fewer colors, the colors are converted automatically.

Figure 6.2 Icon suite Sample Views



TIP You can use drag and drop with the icon suite Sample Views to create 4-bit icons, black and white icons, and masks by simply dragging the 8-bit icon into the various Sample Views.

Bitmap Editor Tools

This section discusses the individual tools available on the tool palette. The tools are:

- Lasso Tool •Marquee Tool
- Text Tool •Pencil Tool
- Eraser Tool •Paint Bucket Tool
- Dropper Tool •Line Tool
- Filled Rect Tool •Empty Rect Tool
- Filled Rounded Rect Tool •Empty Rounded Rect Tool
- Filled Oval Tool •Empty Oval Tool

- Hot Spot Tool •Pattern Tool
- Color Tool

Lasso Tool

The icon for the lasso tool is:

Use the lasso tool to select pixels in an irregularly shaped area.

To use the tool, select it in the tool palette. Then click and drag in the Editing View. As you do, a thin line appears following your path. When you release the mouse button, the path closes and converts into an animated dashed line (also known as marching ants).

The bounds of the marching ants indicate the selected pixels. If you do not complete a closed path with the tool, Constructor closes the path with a straight line between your starting point and the point at which you released the mouse button.

The lasso tool ignores pixels in the current background color. If you have an object in a bitmap surrounded by background color, you can draw the lasso loosely around the object. When you release the mouse button, the lasso tightens around the object automatically.

TIP Double-click the lasso tool to select the outermost outline of non-background-color pixels in the Editing View.

See also: “Tool Palette,” “Marquee Tool,” “Selecting Pixels.”

Marquee Tool

The icon for the marquee tool is:

Use the marquee tool to select pixels in a rectangular area.

To use the tool, select it in the tool palette. Then click and drag to define a rectangular area. As you do, an animated selection rectangle (also known as marching ants) appears. When you release the mouse button, pixels within the selection rectangle are selected.

If you hold the Shift key down while dragging, the marquee is restricted to a square shape.

TIP Double-click the marquee tool to select the entire bitmap in the Editing View.

See also: “Tool Palette,” “Lasso Tool,” “Selecting Pixels.”

Text Tool

The icon for the text tool is: T

Use the text tool to enter text in a bitmap. Select this tool, click in a bitmap to set an insertion point, and type. The initial position of your click forms one margin for your text, either right or left depending upon your justification settings. Text will then extend to the edge of the bitmap before wrapping. To force a shorter line, use the Return key.

Use the Font and Style menus to choose text characteristics for the text you enter. Text appears in the current foreground color. You can have only one font/style/color per text entry.

Once you click the mouse to end text entry, the text becomes part of the bitmap and loses its textual nature. You cannot edit the text after this happens except by modifying the pixels in the bitmap that make up the letters.

The text editor does not support cut, copy, or paste.

TIP To increase or decrease font size by one point as you enter text, type Command-Up Arrow or Command-Down Arrow respectively.

Pencil Tool

The icon for the pencil tool is: ✏

Use the pencil tool to draw a pixel, usually in the foreground color.

To use the tool, select it in the tool palette. Then click a pixel in the Editing View. That pixel changes to the current foreground color, with one exception. If the pixel you click is already in the foreground color, the pixel changes to the background color.

If you click and drag the pencil, each pixel in your path changes to the foreground color. If the first pixel clicked is already in the foreground color, then the pencil draws in the background color.

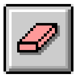
The pencil is one pixel high and wide. You cannot change the default pencil size of one pixel.

TIP Change any non-foreground color pixel to background color by double-clicking the pixel with the pencil.

You can constrain drawing to a straight line. To do so, press the Shift key *before* drawing, and then begin dragging. The pencil is constrained to move in a straight line, either horizontally or vertically. You may release the Shift key after you start dragging and the pencil remains constrained. Pressing the Shift key *after* you begin dragging does not constrain pencil movement.

See also: "Tool Palette," "Paint Bucket Tool," "Line Tool."

Eraser Tool

The icon for the eraser tool is: 

Use the eraser tool to change pixels to the background color. Erasing does not automatically turn pixels white. When you erase the pixel, you set its color to the currently selected background color.

To use the tool, select it in the tool palette. Then click a pixel in the Editing View. That pixel changes to the current background color. If you click and drag the eraser, each pixel in your path changes to the background color.

TIP Double-click the eraser tool to erase the entire Editing View. Another way to erase pixels is to select them with the lasso or marquee tool, then press the Delete key.

You can constrain erasing to a straight line. To do so, first press the Shift key *before* erasing, and then begin dragging. The eraser is constrained to move in a straight line, either horizontally or vertically. You may release the Shift key after you start dragging and the eraser remains constrained. Pressing the Shift key *after* you begin dragging does not constrain movement.

See also: “Tool Palette,” “Lasso Tool,” “Marquee Tool.”.

Paint Bucket Tool

The icon for the paint bucket tool is:

Use the paint bucket tool to fill an area with the current pattern. The pattern is based on the current foreground and background colors.

To use the tool, select it in the tool palette. Then click a pixel. That pixel and all contiguous pixels of the same color will change to the current pattern. Contiguous pixels are those horizontally or vertically adjacent to each other. Two pixels of the same color that are next to each other diagonally are not contiguous as far as the paint bucket tool is concerned.

TIP The paint bucket tool fills an area with the current pattern, *not* the current foreground color (although the pattern is based on the current foreground and background colors). To fill an area with the current foreground color, set the pattern to a solid fill before using the paint bucket tool.

See also: “Tool Palette,” “Pattern Tool,” “Color Tool.”

Dropper Tool

The icon for the dropper tool is:

Use the dropper tool to set the foreground or background color.

To use the tool, select it in the tool palette. Then click a pixel in the Editing View. (The dropper’s hot spot is indicated in red, at the tip of the dropper.)

When you click a pixel with the dropper tool, the foreground color changes to the color of the clicked pixel. If you hold the Shift key down while clicking a pixel, the background color changes to the color of the clicked pixel.

TIP You can set foreground and background color at any time without selecting the dropper tool. Option-click a pixel to set the foreground color. Shift-Option-click a pixel to set the background color.

The dropper tool does not affect the clicked pixel. It is just an easy way of "picking up" a color from the Editing View so that you can apply it elsewhere.

TIP Because the dropper sets the foreground or background color, it also affects the color of the current pattern as well.

See also: "Tool Palette," "Pattern Tool," "Color Tool."

Line Tool

The icon for the line tool is: ╲

Use the line tool to draw a line with the foreground color.

To use the tool, select it in the tool palette. Then click and drag in the Editing View. As you drag, all the pixels in a straight line between your initial click and your current position change to the foreground color. When you release the mouse button, the line is added to the bitmap. The line is one pixel wide.

You can constrain the line direction to 45° increments. To do so, press the Shift key. You may press or release the key at any time. While the Shift key is pressed, the line direction is restricted to multiples of 45°.

You cannot change the default line width of one pixel.

See also: "Tool Palette," "Pencil Tool."

Filled Rect Tool

The icon for the filled rect tool is: ▣

Use the filled rect tool to draw a rectangle that is filled with the current pattern. A filled rectangle has no outline. The current pattern is applied over the entire area of the rectangle. The pattern is based on the current foreground and background colors.

To use the tool, select it in the tool palette. Then click and drag in the Editing View. As you drag, your initial click and your current position define the corners of a rectangle. All pixels in the rectangle are changed to match the current pattern. When you release the mouse button, the rectangle is added to the bitmap.

You can constrain the rectangle to a square shape. To do so, press the Shift key. You may press or release the Shift key at any time. While the Shift key is pressed, the rectangle's sides are equal.

See also: "Tool Palette," "Empty Rect Tool," "Filled Rounded Rect Tool," "Empty Rounded Rect Tool," "Filled Oval Tool," "Empty Oval Tool," "Using Shape Tools."

Empty Rect Tool

The icon for the empty rect tool is: □

Use the empty rect tool to draw a rectangle whose bounds are drawn in the foreground color.

To use the tool, select it in the tool palette. Then click and drag in the Editing View. As you drag, your initial click and your current position define the corners of a rectangle. The bounding pixels of the rectangle are set to the foreground color. The bounds are one pixel wide. Pixels inside the rectangle's bounds remain unchanged. When you release the mouse button, the rectangle is added to the bitmap.

You can constrain the rectangle to a square shape. To do so, press the Shift key. You may press or release the key at any time. While the Shift key is pressed, the rectangle's sides are equal.

You cannot change the default pen size of one pixel.

See also: “Tool Palette,” “Filled Rect Tool,” “Filled Rounded Rect Tool,” “Empty Rounded Rect Tool,” “Filled Oval Tool,” “Empty Oval Tool,” “Using Shape Tools.”

Filled Rounded Rect Tool

The icon for the filled rounded rect tool is: ▢

Use the filled rounded rect tool to draw a rectangle with rounded corners that is filled with the current pattern. A filled rounded rectangle has no outline. The current pattern is applied over the entire area of the rounded rectangle. The pattern is based on the current foreground and background colors.

To use the tool, select it in the tool palette. Then click and drag in the Editing View. As you drag, your initial click and your current position define the corners of a rectangle. All pixels in the rounded rectangle are changed to match the current pattern. When you release the mouse button, the rounded rectangle is added to the bitmap.

You can constrain the rectangle to a square shape. To do so, press the Shift key. You may press or release the key at any time. While the Shift key is pressed, the rectangle’s sides are equal.

You cannot change the amount of roundness that Constructor uses at the corners.

See also: “Tool Palette,” “Filled Rect Tool,” “Empty Rect Tool,” “Empty Rounded Rect Tool,” “Filled Oval Tool,” “Empty Oval Tool,” “Using Shape Tools.”

Empty Rounded Rect Tool

The icon for the empty rounded rect tool is: ▢

Use the empty rounded rect tool to draw a rounded rectangle whose bounds are drawn in the foreground color.

To use the tool, select it in the tool palette. Then click and drag in the Editing View. As you drag, your initial click and your current position define the corners of a rectangle. The bounding pixels of the rounded rectangle are set to the foreground color. The bounds are one pixel wide. Pixels inside the rectangle's bounds remain unchanged. When you release the mouse button, the rounded rectangle is added to the bitmap.

You can constrain the rectangle to a square shape. To do so, press the Shift key. You may press or release the key at any time. While the Shift key is pressed, the rectangle's sides are equal.

You cannot change the default pencil size of one pixel, or the amount of roundness that Constructor uses at the corners.

See also: "Tool Palette," "Filled Rect Tool," "Empty Rect Tool," "Filled Rounded Rect Tool," "Filled Oval Tool," "Empty Oval Tool," "Using Shape Tools."

Filled Oval Tool

The icon for the filled oval tool is: ⬬

Use the filled oval tool to draw an oval filled with the current pattern. A filled oval has no outline. The current pattern is applied over the entire area of the oval. The pattern is based on the current foreground and background colors.

To use the tool, select it in the tool palette. Then click and drag in the Editing View. As you drag, your initial click and your current position define the bounds of an oval. All pixels in the oval are changed to match the current pattern. When you release the mouse button, the oval is added to the bitmap.

You can constrain the oval to a circular shape. To do so, press the Shift key. You may press or release the key at any time. While the Shift key is pressed, the oval is constrained to a circle.

See also: "Tool Palette," "Filled Rect Tool," "Empty Rect Tool," "Filled Rounded Rect Tool," "Empty Rounded Rect Tool," "Empty Oval Tool," "Using Shape Tools."

Empty Oval Tool

The icon for the empty oval tool is: ⬭

Use the empty oval tool to draw an oval whose bounds are drawn in the foreground color.

To use the tool, select it in the tool palette. Then click and drag in the Editing View. As you drag, your initial click and your current position define the corners of an oval. The bounding pixels of the oval are set to the foreground color. The bounds are one pixel wide. Pixels inside the oval's bounds remain unchanged. When you release the mouse button, the oval is added to the bitmap.

You can constrain the oval to a circular shape. To do so, press the Shift key. You may press or release the key at any time. While the Shift key is pressed, the oval is constrained to a circle.

You cannot change the default pen size of one pixel.

See also: "Tool Palette," "Filled Rect Tool," "Empty Rect Tool," "Filled Rounded Rect Tool," "Empty Rounded Rect Tool," "Filled Oval Tool," "Using Shape Tools."

Hot Spot Tool

The icon for the hot spot tool is: ☒

Use the hot spot tool to set the hot spot for a cursor. This tool is available only for resources of type 'CURS' and 'crsr'.

To use the tool, select it in the tool palette. Then click a pixel in the cursor's Editing View where you want the hot spot to be. That pixel appears in the Editing View as a target bull's-eye, as shown in Figure 6.3. You can have one and only one hot spot in a cursor. The hot spot occupies one pixel.

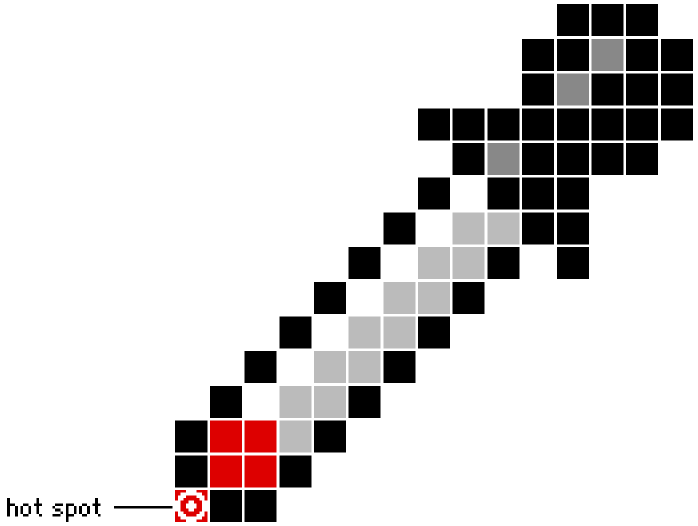
For beginners

The hot spot is the point used by the Mac OS Toolbox calls such as `GetMouse()` to determine the precise location of the cursor.

Setting the hot spot does not affect the visual appearance of the chosen pixel at runtime. It only modifies the appearance in the Editing View so you can see where the hot spot is.

See also: "Tool Palette."

Figure 6.3 The hot spot for the dropper tool



Pattern Tool

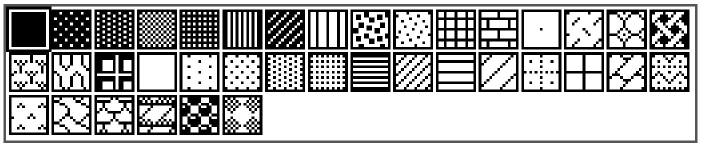
The icon for the pattern tool is: ▦

Use the pattern tool to select a pattern to use when drawing.

To use the tool, click the icon and hold the mouse button down. While the mouse button is down, a pop-up menu appears showing you available patterns. Drag the mouse to the pattern you wish to use, then release the mouse button. When you do, the central area of the pattern tool changes to reflect your selected pattern.

Figure 6.4 illustrates the pattern pop-up menu. The currently selected pattern is outlined. In this case, it is the solid fill pattern in the top left corner of the pop-up menu.

Figure 6.4 The pattern pop-up menu

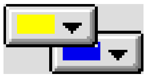


The available patterns use the current foreground and background colors. If you change the colors, you change the colors for the patterns as well.

You cannot edit the default patterns available in Constructor.

See also: “Tool Palette,” “Color Tool,” “Setting Colors and Patterns.”

Color Tool

The icon for the color tool is: 

Use the color tool to set the foreground and background colors. There are actually two icons, one in front of the other. The front icon represents the foreground color. The other icon represents the background color.

To use the tool, click the icon for the foreground or background color, and hold the mouse button down. While the mouse button is down, a pop-up menu appears displaying a color palette. Drag the mouse to the color you wish to use, then release the mouse button. When you do, the central area of the appropriate color tool changes to reflect your selected color.

The colors available in the pop-up menu depend upon two factors: the resource being edited, and the setting in the **Colors** menu.

Some resources support color, others do not. For example, if you are editing a ‘CURS’ resource, you are limited to black and white. Those are the only colors that appear in the pop-up menu, and the **Colors** menu is disabled.

If you are editing a resource that supports color, the **Colors** menu is enabled. You can then pick from several color palettes.

If you do not wish to be limited to a color palette, choose the **Use Color Picker** item in the **Colors** menu. When this item is checked and you click the color tool, the standard dialog appears allowing you to pick any color.

The available patterns use the current foreground and background colors. If you change a color, you change the color for the patterns.

TIP You can also set the foreground color to match an existing pixel by clicking a pixel with the dropper tool, or Option-clicking a pixel at any time. Set the background color by Shift-clicking a pixel with the dropper tool, or Shift-Option-clicking a pixel at any time.

See also: "Tool Palette," "Dropper Tool," "Pattern Tool," "Setting Colors and Patterns," "Choosing a Color Palette."

Editing a Bitmap

Editing a bitmap with Constructor is so simple it hardly needs explanation.

Select the tool that represents the operation you wish to perform. Then click (or click and drag) in the Editing View to modify the pixels in the bitmap. See "Bitmap Editor Tools" for details on how to use each individual tool.

You can perform operations in any order, on any pixel, at any time. You can undo or redo the most recent operation.

However, some functions do have less obvious features that deserve some explanation. Topics discussed in this section include:

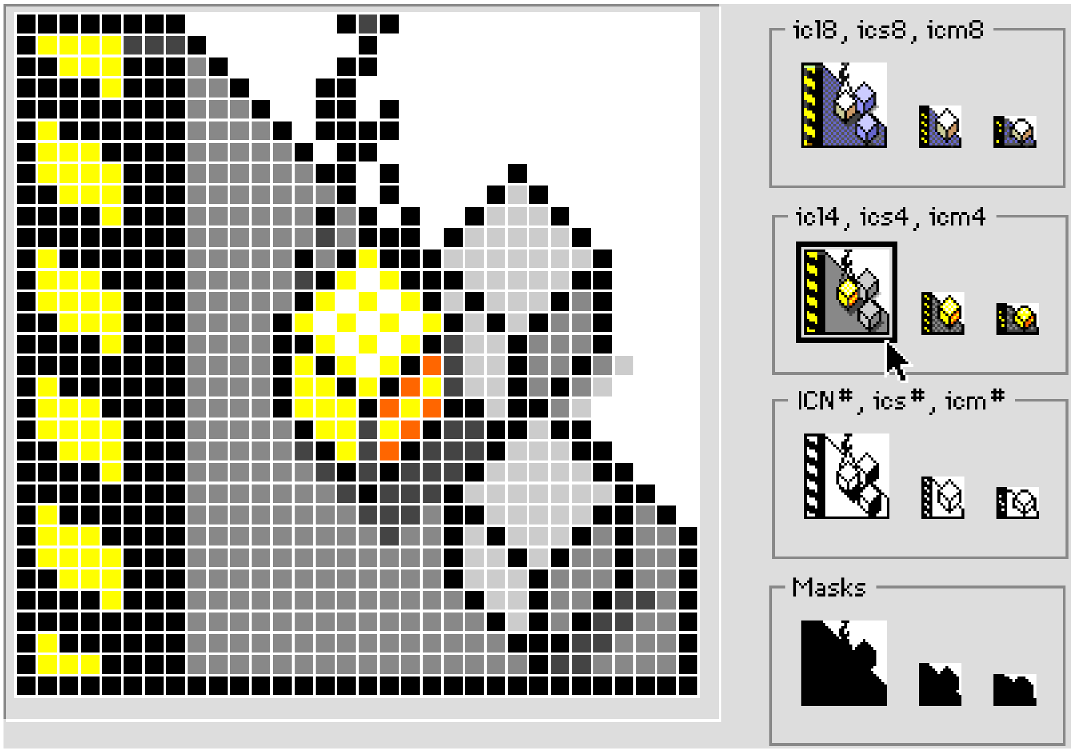
- Sample Views—choosing which sample to edit
- Nudging Pixels—moving selected pixels just a little bit
- Copying Pixels—making a copy of selected pixels
- Rotating and Flipping Pixels—manipulating selected pixels
- Using Transparency—seeing through a bitmap
- Setting Colors and Patterns—tips on setting color and pattern
- Choosing a Color Palette—controlling available colors
- Using Shape Tools—tips on using basic shapes
- Drag and Drop—using drag and drop effectively

Sample Views

The bitmap editor window usually shows several Sample Views of the bitmap. The sample view currently on display in the Editing View has a heavy black border, as shown in Figure 6.5. To view the bitmap for a particular resource, click on the sample view.

For example, to edit the ‘icl4’ resource of an icon suite, simply click the ‘icl4’ sample view, and it appears in the Editing View.

Figure 6.5 Choosing which resource to edit



Selecting Pixels

You may select pixels in a variety of ways.

Use the Lasso Tool to select irregular shapes. Double-click the Lasso Tool to select the outermost outline of non-background-color pixels in the Editing View.

Use the Marquee Tool to select rectangular shapes. Use the Shift key to contrain the selectoin marquee to a square shape. Double-click the Marquee Tool to select the entire bitmap.

You may select an entire bitmap by choosing the **Select All** command in the **Edit** menu.

Nudging Pixels

Sometimes you want to move a group of pixels just a few pixels away from its current position. It can be difficult to be this precise dragging with the mouse.

To nudge pixels, first select the pixels. Then use the cursor control keys (the arrow keys) to move the pixels in the desired direction: up, down, left, or right. Each time you type an arrow key, the selected pixels moves by one pixel in the corresponding direction.

Copying Pixels

You can copy the current selection using standard **Edit** menu commands. You can also press the Option key while dragging the selection. An Option-drag copies selected pixels. The Option key must be pressed when the drag begins. You can release the Option key at any time.

Rotating and Flipping Pixels

When a bitmap editor window is active, an **Options** menu contains commands related to manipulating bitmaps. You can use these commands to flip or rotate currently selected pixels.

The **Flip Vertical** command turns the selected pixels upside down.

The **Flip Horizontal** command swaps selected pixels left and right. This creates a mirror image of the pixels.

The **Rotate Right** command turns selected pixels 90° clockwise.

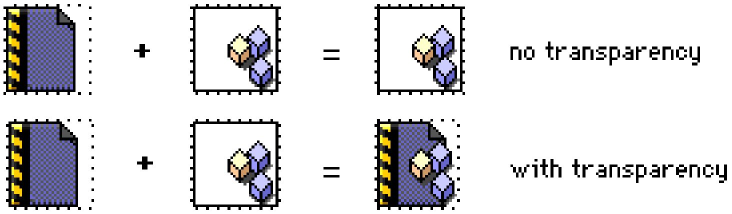
The **Rotate Left** command turns selected pixels 90° counterclockwise.

Using Transparency

The **Make Transparent** command in the **Options** menu allows you to combine bitmaps in interesting ways. In this case a picture is worth a thousand words.

Figure 6.6 demonstrates the effect of transparency. Assume the background color in the original document icon is white. You want to combine another icon-sized shape with the original document icon. Most of that new icon is occupied by white pixels.

Figure 6.6 Effect of transparency



When you put one icon on top of the other without transparency, the new icon obliterates the underlying document icon. If you choose **Make Transparent**, the currently selected pixels that match the background color do not hide any pixels that appears behind them. This allows you to combine bitmaps as shown in Figure 6.6. This works for any bitmap, not just icons.

Setting Colors and Patterns

There are three ways you can set foreground and background colors. To set the foreground color you can:

1. use the foreground Color Tool
2. select the Dropper Tool and click a pixel
3. Option-click a pixel

To set the background color you can:

1. use the background Color Tool
2. select the Dropper Tool, then Shift-click a pixel
3. Shift-Option-click a pixel

In addition, you can choose the **Black & White** command in the **Colors** menu to set the foreground to black and the background to white.

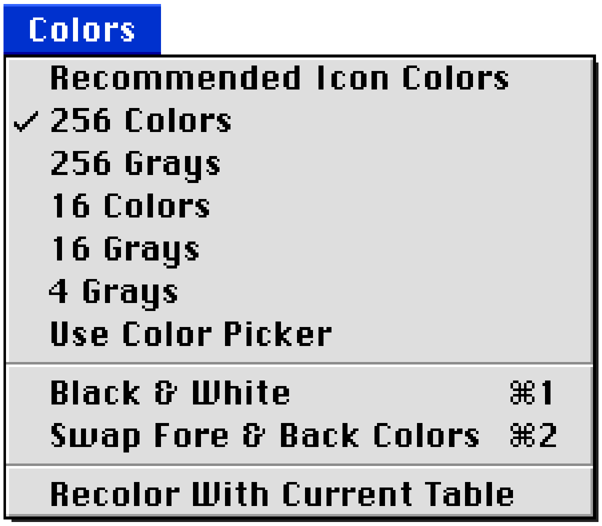
The **Swap Fore & Back Colors** command in the **Colors** menu lets you switch the foreground and background colors simply.

Any change in either foreground or background color affects the colors in the current pattern. By judiciously picking colors, patterns, and transparency, you can quickly create complex effects such as grid lines, slashes, and interesting color blends.

Choosing a Color Palette

Use the **Colors** menu to pick the color palette you wish to use. This menu is enabled when the bitmap editor is active and you are working on a bitmap that supports color. The currently selected palette is marked with a check mark. To turn off a check mark, you must select another item on the menu.

Figure 6.7 The Constructor Colors menu



Most items in the menu are self-explanatory.

Palettes are only available when appropriate. If the resource does not support certain colors or grays, the corresponding palette choice in the **Colors** menu will be disabled.

The **Recommended Icon Colors** reflect Apple Computer's suggested colors for Mac OS icon design. These colors are optimized for desktop display. The other color palettes reflect standard color and gray palettes used by the Mac OS System.

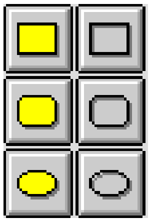
If you choose the **Use Color Picker** item, you modify how the Color Tool behaves. When this item is checked and you click the Color Tool, the standard color dialog appears. This allows you to select any color. See “Color Tool.”

The **Recolor To Current Table** changes the colors in the Editing View to match the currently-checked color palette. This is useful to update your bitmap.

WARNING! If you choose a palette with fewer colors or grays, and then recolor your bitmap, you cannot restore original colors or grays at a later time. The color information is lost.

For example, assume you have a bitmap that uses 256 colors. You choose the 256 grays palette, and then recolor the bitmap to use those grays. The bitmap changes to match the new palette. If you subsequently choose the 256 colors palette, and recolor the bitmap to use 256 colors, your original colors do *not* reappear. The color information is lost when you reduce available colors, either by switching to gray or using a palette with fewer colors or grays.

Using Shape Tools



Use these tools to draw shapes without having to manipulate the shapes pixel by pixel.

Use a filled shape tool (in the left column of the tool palette) to draw a shape that is filled with the current pattern. Use an empty shape tool (in the right column of the tool palette) to draw a shape whose bounds are drawn in the foreground color.

To use the tool, select it in the tool palette. Then click and drag in the Editing View. As you drag, your initial click and your current position define the bounds of the shape. When you release the mouse button, the shape is added to the bitmap.

You can constrain the shape with the Shift key. You may press or release the Shift key at any time while drawing the shape. While the Shift key is pressed, the sides of the shape are equal. This makes rectangle shapes square, and oval shapes circular.

TIP To create an outlined and filled shape, draw an empty shape first. Then use the Paint Bucket Tool to fill the insides of the empty shape with your chosen pattern.

See also: "Tool Palette," "Empty Rect Tool," "Filled Rounded Rect Tool," "Empty Rounded Rect Tool," "Filled Oval Tool," "Empty Oval Tool."

Drag and Drop

The bitmap editor has powerful drag and drop features, especially related to the Sample Views. Although you cannot edit the Sample Views directly, they are fully active for drag and drop.

Here are some tips for easy bitmap editing with drag and drop.

After you create an 8-bit icon in large and small sizes, drag the samples to the corresponding 4-bit, black and white, and mask samples. You now have a whole icon family! Best of all, you only had to draw each icon once. You may want to fine-tune the new icons, but most of the work has been done automatically. The same trick is useful for other kinds of resources, such as color cursors. You can even drag a large icon into a small icon sample view, although scaling generally doesn't generate particularly aesthetic results.

When editing a bitmap, you can make an instant backup to save work in progress. Simply drag the sample to the desktop, or to any folder. If you later decide you want to revert to a previous stage of development, drag your saved work from the desktop back into the sample view.

If you have a favorite bitmap that you want to use as a basis for further work, use Constructor to open the source file that contains the original bitmap resource. Then you can simply drag the original bitmap into your own Constructor project, and you're on your way.

You can drag and drop between different bitmap editor windows. For example, to create an ICON, simply drag the sample from another large icon, such as an 'icl8' and drop it in the ICON sample view.

7

Editing Text Traits

This chapter discusses how to edit a text traits resource.

Editing Text Traits Overview

Many parts of a visual interface involve text. PowerPlant lets you decide what you want that text to look like, on a pane-by-pane basis. PowerPlant handles the background housekeeping necessary to set a font, size, style, justification, color, and text drawing mode for the text in a pane. You simply set those properties in a text traits resource, and assign the resource ID to the pane.

Text traits are used in PowerPlant classes that have a textual content, including LCaption, LGroupBox, LEditField, LTextEdit, LListBox, and all the LStdControl classes (for the control's title).

This chapter discusses how to manage and modify the *contents* of a text traits resource. For information on how to create, copy, open, modify, and delete a text traits resource as a whole, see "Managing Resources Overview."

The topics in this chapter are:

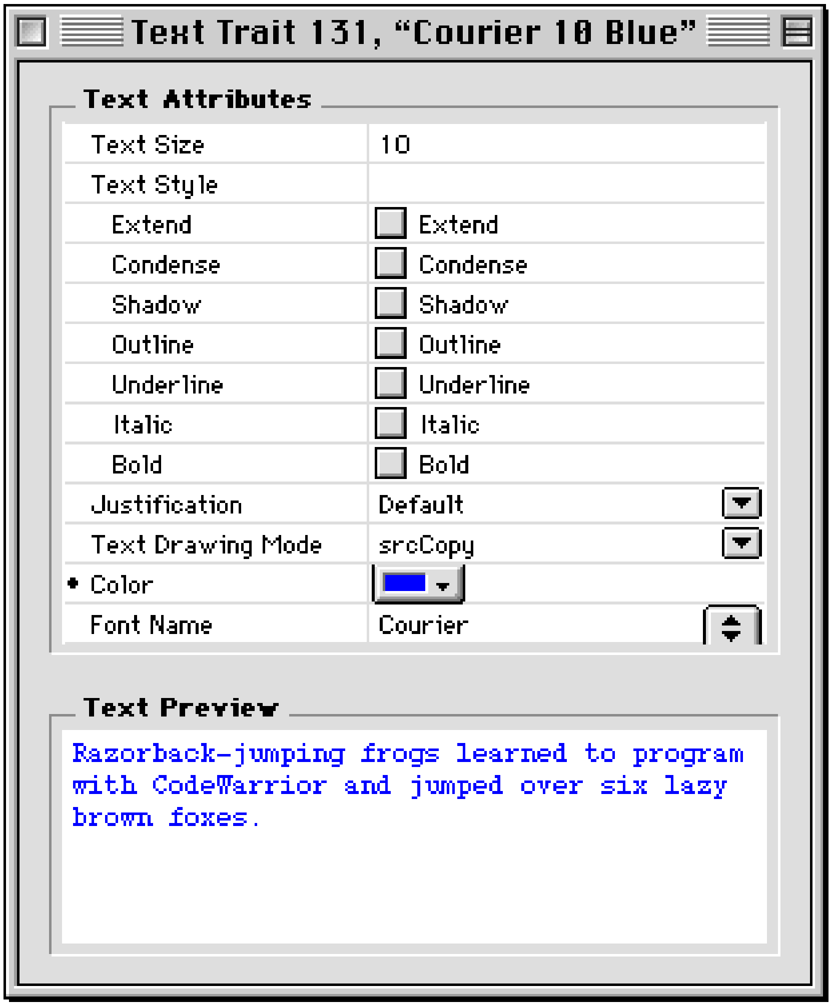
- The Text Traits Editor Window—the editor window
- Setting Text Traits Options—editing a text traits resource
- Using Text Traits Effectively—tips on managing text traits

The Text Traits Editor Window

To modify text traits, you use the text traits editor window shown in Figure 7.1. It looks very much like a standard Property Inspector Window. Constructor does not support modifying a pane's text characteristics in the layout or hierarchy windows.

The bottom part of the editor window is a preview of the appearance of the text. The background of the preview pane changes color based on the foreground color of the text. For very light colors, the background becomes black so you can see your text.

Figure 7.1 The text traits editor



Text Trait 131, “Courier 10 Blue”

Text Attributes

Text Size	10
Text Style	
Extend	□ Extend
Condense	□ Condense
Shadow	□ Shadow
Outline	□ Outline
Underline	□ Underline
Italic	□ Italic
Bold	□ Bold
Justification	Default
Text Drawing Mode	srcCopy
• Color	🟦
Font Name	Courier

Text Preview

Razorback-jumping frogs learned to program with CodeWarrior and jumped over six lazy brown foxes.

Setting Text Traits Options

You may set font, size, alignment, style, text drawing mode, and color.

Setting a font

To set a font, pick a font from the pop-up menu. The menu lists every font available on the machine running Constructor. You may also choose from *System font* and *Application font*.

The system font is the font the Mac OS System uses for menus. The Application Font is the default font used by applications in the Mac OS. In US Systems, the System font is Chicago and the application font is Geneva. Other language systems use different fonts. Users may also change the default application font.

Therefore, if you specify the system or application font, you cannot be 100% sure what font will be used on a given machine. In general you should specify common fonts such as those that are provided with the Mac OS System. If you use a non-standard font, make sure you license it and include it with your software.

Setting font size

Enter a font size (in points) in the *Size* field. A size of 0 means the pane will use the font's default size, usually 12 points.

Generally you should use common sizes such as 9, 10, or 12 points. Non-standard sizes might look bad if a user does not have a TrueType font or Adobe Type Manager (ATM) for postscript fonts.

Setting justification

Choose the desired alignment from the pop-up menu. Table 7.1 lists each option and its effect.

Table 7.1 Alignment options

Option	Text is aligned…
System default	at left side of pane in Roman languages, appropriately justified for other languages
Left flush	at left side of pane
Center	in center of pane
Right flush	at right side of pane

Setting text style

Click the check boxes for the desired combination of options.

Setting text drawing mode

Choose a text drawing mode from one of the options in the pop-up menu.

The text drawing mode controls how the text is drawn over the background. The default value is *srcCopy*, which completely replaces the background with the text. This works great for black and white, but *srcOr* works better on color backgrounds. For more information about Mac OS text drawing modes, see *Inside Macintosh: Text*, pages 3-70.

Setting color

Click the color control in the editor window to pick a color from the color palette. The color swatch in the center of the control shows you the current color. The default text color is black.

TIP Option-click the color control to open the standard color-picker dialog to set a custom color.

Using Text Traits Effectively

A text traits resource is a description of a particular appearance, not text content. You set the font, size, style, color, alignment, and text drawing mode in the text traits resource, not the text itself.

Therefore, you do not have to create a text traits resource for each and every pane. You create a text traits resource for each and every different appearance you use.

If you want to use Courier 9-point bold text in red, create a text traits resource that describes the desired appearance. Use the pane property inspector window to assign that text traits resource ID to every pane whose text uses that appearance. If you later decide that you want the text to be blue, just change the text traits resource. All the panes that refer to that resource are updated automatically.

The text traits resource abstracts text appearance out of panes and into a separate object. This gives you the benefit of an object-oriented approach to text appearance. You create a text-appearance resource, and have panes refer to that resource. If you modify the appearance, you modify every pane that refers to the appearance resource. Simple, and elegant.

If you have several text traits resources in your project, there are two tricks that can help you manage them easily.

First, give each resource a name that describes the text settings. For example, the name “Courier 9 Bold Red” tells you what you need to know. Then you don’t have to remember what the settings are in text traits resource number 134.

Second, if you are making a series of resources that vary in only one feature, make several copies of an original, and then modify just the one feature in the copies. For example, if you want three identical text traits resources except that one is black, one red, and one blue, define the first resource completely. Make two copies, and change the color of one to red, and the color of the other to blue.

PowerPlant panes that use text refer to a single text traits resource. Therefore, all text in the pane will have the same appearance. You can switch text traits at runtime, however. See *The PowerPlant Book* chapter on utilities for information on how to set up and modify text traits at runtime.

8

Editing String Lists

This chapter discusses how to edit a string list resource with Constructor.

Editing String Lists Overview

String lists, stored in an STR# resource, are a common feature of Mac OS code. Each STR# resource contains a series of text strings that are accessed by an index number. The index number represents the position of the string in the list.

String lists are not exclusive to PowerPlant. Constructor can edit STR# resources for any type of Mac OS code.

This chapter discusses how to manage and modify the *contents* of a string list resource. For information on how to create, copy, open, modify, and delete a string list resource as a whole, see “Managing Resources Overview.”

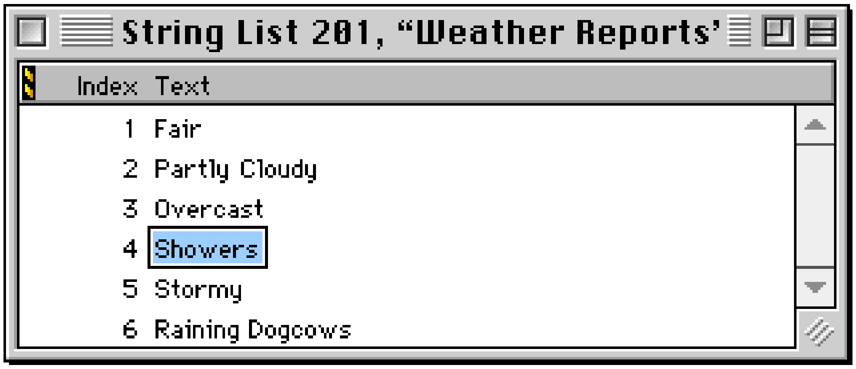
The topics in this chapter are:

- The String List Editor Window
- Editing a String List Resource

The String List Editor Window

The editor window for string lists is very simple. It has two columns: the index number for the string, and the text for the string itself.

Figure 8.1 The string list editor



Index	Text
1	Fair
2	Partly Cloudy
3	Overcast
4	Showers
5	Stormy
6	Raining Dogcows

Editing a String List Resource

The string list editor is very simple. You may add, modify, move, or delete any string in the list.

To add a new string to a string list, choose the NewString item from the Edit menu.

Edit the string text just as you would any entry in Constructor. See “Text Editing.”

To remove a string from the list, select it and:

- press the Delete key
- drag the item to the trash
- choose the Clear command from the Edit menu
- choose the Cut command from the Edit menu

9

Editing Custom Types

This chapter discusses how to edit a custom type (CTYP) resource.

Editing Custom Pane Types Overview

You will regularly create visual panes based on custom classes derived from PowerPlant. Some of these custom classes will have additional data members. The Constructor property inspector will display the data for your custom class. You can then set the default or initial properties for the custom pane in exactly the same way you would for a standard PowerPlant pane.

This chapter discusses how to manage and modify the *contents* of a CTYP resource that describes the data items in a custom pane. For information on how to create, copy, open, modify, and delete a CTYP resource as a whole, see "Managing Resources Overview."

The main topics in this chapter are:

- What a CTYP Resource Does—a straightforward explanation of the purpose of a custom type resource
- Simple Custom Panes—custom panes that have no additional data members and that don't need a CTYP resource
- Custom Type Editor Window—the CTYP editor window
- Editing a Custom Type Resource—adding and removing items from a CTYP resource
- Setting Class Properties—modifying the class properties represented by the CTYP
- Setting Data Item Properties—modifying the data items in the CTYP resource

Moving from CPPb to CTYP Resources

As of Constructor 2.4, CPPb resources are no longer supported. The new resource type (with a different internal storage format) is CTYP. The CTYP resource format will allow for new features in the future.

CPPb resources must be converted to CTYP resources using the converter application supplied with Constructor. The name of this application is Make CTYP. Drop any resource file that contains a CPPb resource onto this converter, and it will automatically generate corresponding CTYP resources. The original CPPb resource will remain in the resource file.

What a CTYP Resource Does

This section examines the purpose, benefits, and limitations of the CTYP resource architecture in Constructor.

The topics in this section are:

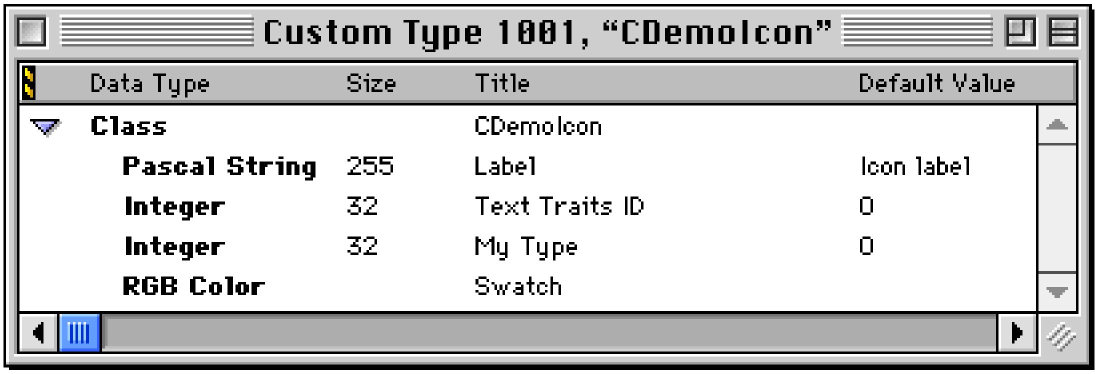
- Custom Type Properties—how a CTYP resource lets you create a custom information for the Property Inspector Window
- CTYP Libraries—how Constructor can use libraries of CTYPs
- Limitations—some things that the CTYP architecture cannot do

Custom Type Properties

When you edit a CTYP resource, you define what the Property Inspector Window will look like for your custom pane. The CTYP resource specifies the class name, class ID, and additional data types for a custom type. Based on that information, Constructor displays all the necessary fields so you can set the custom data when you open the property inspector for that pane.

Figure 9.1 illustrates a hypothetical CTYP resource for a CDemoIcon pane.

Figure 9.1 A typical custom type resource



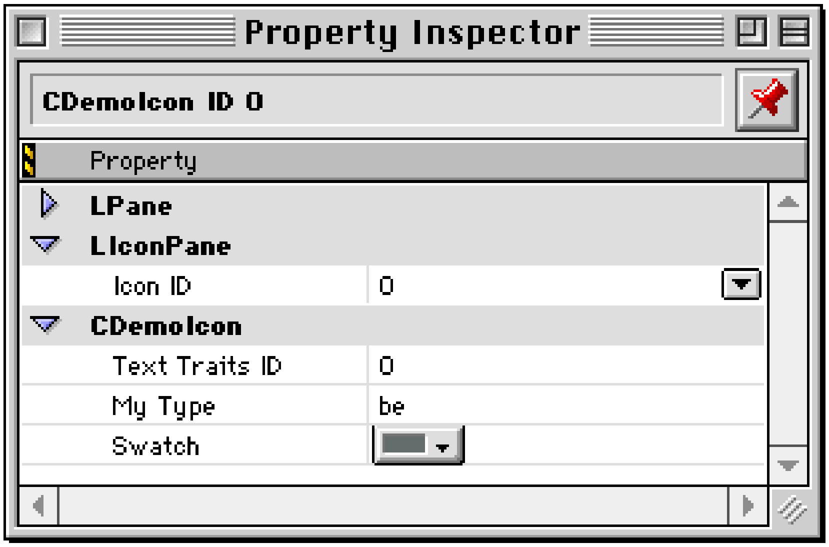
Custom Type 1001, “CDemoIcon”

Data Type	Size	Title	Default Value
Class		CDemoIcon	
Pascal String	255	Label	Icon label
Integer	32	Text Traits ID	0
Integer	32	My Type	0
RGB Color		Swatch	

This resource defines three new data items for a custom pane. Instead of just a plain icon, this icon has a label. The pane has a text traits ID to describe the appearance of the label text. And there is a third data item that specifies a special value associated with this icon.

After you create such a resource, Constructor adds the corresponding pane to the Catalog Window. You can then add the CDemoIcon pane to any layout resource. If you then open the pane property inspector, what you see is illustrated in Figure 9.2.

Figure 9.2 The property inspector for a CDemoIcon



Property Inspector

CDemoIcon ID 0

Property	
LPane	
LIconPane	
Icon ID	0
CDemoIcon	
Text Traits ID	0
My Type	be
Swatch	▨

Based on the information in the CTYP resource illustrated in Figure 9.1, Constructor has created a pane property inspector that contains the three new data items: Label, Text Traits ID, and My Type. You can now enter values for these items just like you could for a standard PowerPlant pane.

CTYP Libraries

When Constructor launches, it looks in its own directory for resource files it has created. In those files, it looks for CTYP resources. When Constructor finds a custom type resource, it adds that pane to the Catalog Window.

You can create CTYP resources and save them in one or more resource files in the same folder as Constructor. When Constructor launches, it recognizes the file, reads the data from the CTYP resources, and adds them to the Catalog Window.

Use this technique to create sets of CTYPs that are globally available to every Constructor project you open. This allows you to create libraries of custom types that are always available for easy use.

WARNING! Constructor loads the files alphabetically. If you have a custom type that uses a CTYP in another file as its parent class, and the parent CTYP loads after the child CTYP, the child CTYP will not appear in the Catalog Window because Constructor doesn’t know about the parent class. To avoid this problem, rename the files so they load in the proper order.

Limitations

There are some things that the CTYP resource doesn’t do for you.

If a custom type has additional data items, the stream constructor for that class must read these additional data items from the PPob resource and initialize the appropriate data members.

You are limited to the data types offered. There are three basic types:

- an integer (of 1-32 bits in length, signed or unsigned)

- resource ID
- a Pascal string
- an RGB color

Simple Custom Panes

A simple custom pane represents a class derived from a standard PowerPlant visual class, but with no additional data members.

If a custom class has no additional data members, use the standard PowerPlant panes to represent the custom class in Constructor. There is no need to create a CTYP resource for the custom class.

For example, let’s say you declare a class named CMyToggleButton that derives from LToggleButton. CMyToggleButton declares no additional data members, it just overrides LToggleButton behavior.

When you want to add a CMyToggleButton pane to your view, use the LToggleButton pane in Constructor. When you set the pane properties, change the class ID for the pane to whatever the class ID is for the CMyToggleButton class. Set the other properties as appropriate for your CMyToggleButton pane. You can use the standard PowerPlant pane because the properties are identical.

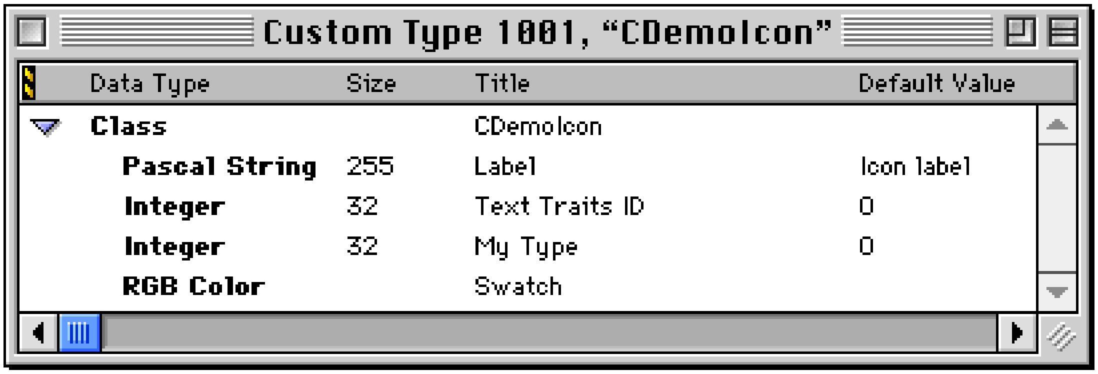
Custom Type Editor Window

When you open a CTYP resource, the editor window for that resource appears. You may open the editor window in three ways:

- double click the CTYP resource in the project window
- select the CTYP resource and press the Return key
- select the CTYP resource and choose **Edit Resource** from the **Edit** menu

Figure 9.3 shows the CTYP editor window with a typical CTYP resource.

Figure 9.3 The CTYP editor



Data Type	Size	Title	Default Value
Class		CDemoIcon	
Pascal String	255	Label	Icon label
Integer	32	Text Traits ID	0
Integer	32	My Type	0
RGB Color		Swatch	

The grey bar in the editor window contains the column heads for the data in the window. The first line of information contains the name of the class.

The editor window displays three columns of information. The data type is not editable. That is set when you create the data item. The Size, Title, and Default Value is editable in place for all data items. The class title is editable as well.

To learn how to add or remove data items from the class, see “Editing a Custom Type Resource”.

You use the standard Property Inspector Window to examine and modify the properties for the class, or the individual data items. For more on these issues, see:

- “Setting Class Properties”
- “Setting Data Item Properties”

Editing a Custom Type Resource

A custom type resource describes the data items unique to a custom class. The available data types for these items are:

- an integer (of any length, signed or unsigned)
- a resource ID
- a Pascal string
- an RGB color

This section discusses the individual tasks you must perform to edit a CTYP resource. These tasks include:

- Adding a Data Item
- Reordering Data Items
- Removing a Data Item

For information on how to create a new CTYP resource, see "Creating a Custom Type Resource."

For information on how to modify class properties, such as the class name or inheritance, see "Setting Class Properties."

For information on how to modify data item properties such as size, title, or visibility, see "Setting Data Item Properties."

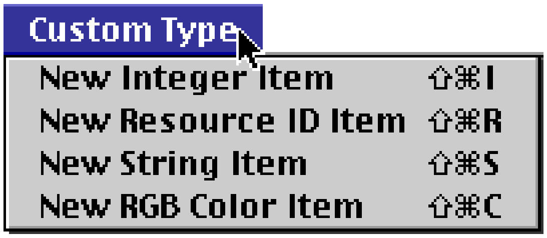
For an example of how to create a CTYP, see the code exercise for Chapter 8, "Controls and Messaging" in *The PowerPlant Book*.

Adding a Data Item

The data items in the CTYP resource correspond to data members in your custom type class. The items you add to the CTYP resource appear in the pane property inspector for the custom type.

To add an item to a CTYP resource, make sure the Custom Type Editor Window is active. When it is, the Custom Type menu appears in the menu bar. Use this menu to choose the type of data you wish to add. Figure 9.4 shows the choices available.

Figure 9.4 The Custom Type menu



The new item appears after the currently selected data item, or at the end of the data items if there is no selection.

You may also copy, paste, duplicate, and Option-drag to copy one or more items.

You may add as many items as you wish. However, the property inspector window can become unwieldy if you add many items.

Reordering Data Items

The order of items in the CTYP editor window determines the order in which items appear in the pane property inspector. To modify the order, simply drag an item to a new location.

You can also change the order of data items using cut and paste. A pasted item is placed in the list after the currently selected item.

NOTE The top entry for the class cannot be moved. It is always at the top of the list.

Removing a Data Item

To remove one or more data items from the CTYP resource, first select the item or items. Then you can:

- press the Delete key
- choose **Cut** or **Clear** from the **Edit** menu
- drag the items to the trash

The items disappear from the resource. As with most Constructor actions, you can undo this operation.

NOTE You cannot delete the class from the CTYP editor window. To delete the class, you must go to the Constructor Project Window and delete the resource. The class listing in the CTYP resource is really a reference to the resource itself.

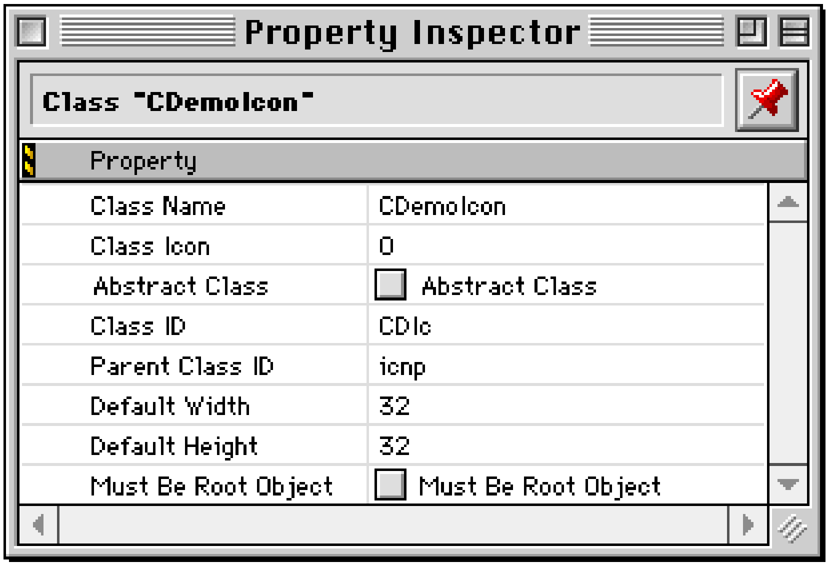
Setting Class Properties

You use the class property inspector to set class-related data. To open this window, you:

- double-click the class in the CTYP editor window
- select the class and press Return
- select the class and choose Property Inspector from the Window menu

When you do, the window in Figure 9.5 appears.

Figure 9.5 The class property inspector



Property Inspector

Class "CDemoIcon"

Property	
Class Name	CDemoIcon
Class Icon	0
Abstract Class	□ Abstract Class
Class ID	CDIc
Parent Class ID	icnp
Default Width	32
Default Height	32
Must Be Root Object	□ Must Be Root Object

Table 9.1 lists the purpose of each item in this window.

Table 9.1 Class properties

Item	Specifies
Class Name	the name of the class
Class Icon	Resource ID of a small icon to use in the Catalog Window
Abstract Class	whether this is an abstract class

Item	Specifies
Class ID	4-character string for the class ID
Parent Class ID	the class ID of the immediate base class
Default Width/Height	the default size of the pane in pixels
Must Be Root Object	must be at the top of the visual hierarchy

An abstract class cannot be added to a layout. However, the abstract class will appear in the Catalog Window (in red) in hierarchy view so you can see the inheritance chain.

A root object is one which cannot be contained within another view. Examples of PowerPlant classes that are root objects include LWindow, LDialogBox, LGrafPortView, and LPrintout.

See also:

- Setting the Class Name and ID
- Setting Class Inheritance
- Setting Custom Pane Size

Setting the Class Name and ID

The class name and ID appear in the Catalog Window. The Class ID field holds the four-character code that identifies this class. PowerPlant reserves codes that are all-lowercase for internal use. If you use all lowercase letters, Constructor will warn you. You should use at least one uppercase letter or number for your class ID.

You can edit the class name in place in the Custom Type Editor Window. Select the class name, and enter edit mode. You can do this by clicking on the class name and waiting one second. You may edit the class name in the class property inspector as well.

To modify the class ID, open the class property inspector and edit the Class ID Field. You can open the class property inspector by double-clicking the class in the Custom Type Editor Window.

Setting Class Inheritance

The class inheritance for the custom pane class determines where in the Catalog Window the custom pane appears. It also determines what data will appear in the pane property inspector, because Constructor builds the pane property inspector out of properties for each ancestor class.

You must specify the immediate base class, not a distant ancestor. To specify the base class, open the class property inspector. Edit the value in the Parent Class ID field. You can open the class property inspector by double-clicking the class in the Custom Type Editor Window.

TIP The Catalog Window lists the class ID for each class. It makes a handy reference for class IDs when setting the parent class ID.

Setting Custom Pane Size

When you drop a pane into a PPob resource, it has a default size. Your custom pane has a default size as well.

To modify the pane size, open the class property inspector and edit the Default Width and Default Height fields. You can open the class property inspector by double-clicking the class in the Custom Type Editor Window.

Setting Data Item Properties

Each data item has a unique set of properties. To edit the size, title, or default value of any item, you can use the Custom Type Editor Window and edit the value in place. You may also edit these properties in the Property Inspector Window for the item.

Each data item has additional properties. To edit these properties, you must use the property inspector. To open this window you:

- double-click an item in the Custom Type Editor Window
- select an item and press Return

- select an item and choose Property Inspector from the Window menu

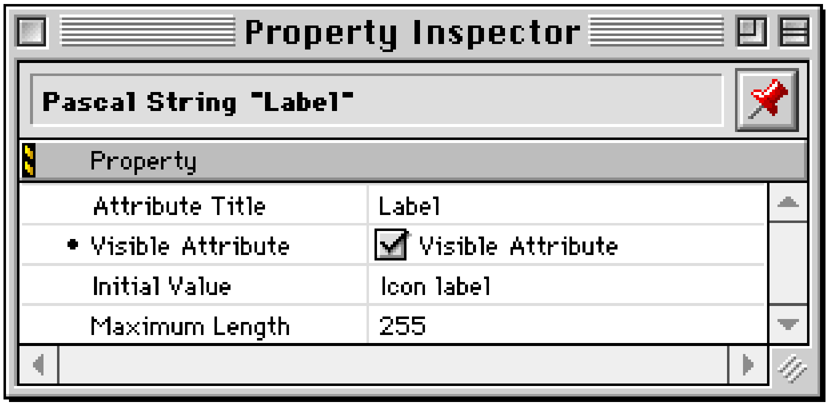
For details on the properties for each data type, see:

- Editing String Properties
- Editing Integer Properties
- Editing RGB Properties

Editing String Properties

The string data item represents a Pascal-style string—one with a length byte followed by the contents of the string. Figure 9.6 shows the string property inspector. Table 9.2 lists the purpose of each property.

Figure 9.6 The string property inspector



Pascal String "Label"

Property	
Attribute Title	Label
• Visible Attribute	☑ Visible Attribute
Initial Value	Icon label
Maximum Length	255

Table 9.2 String properties

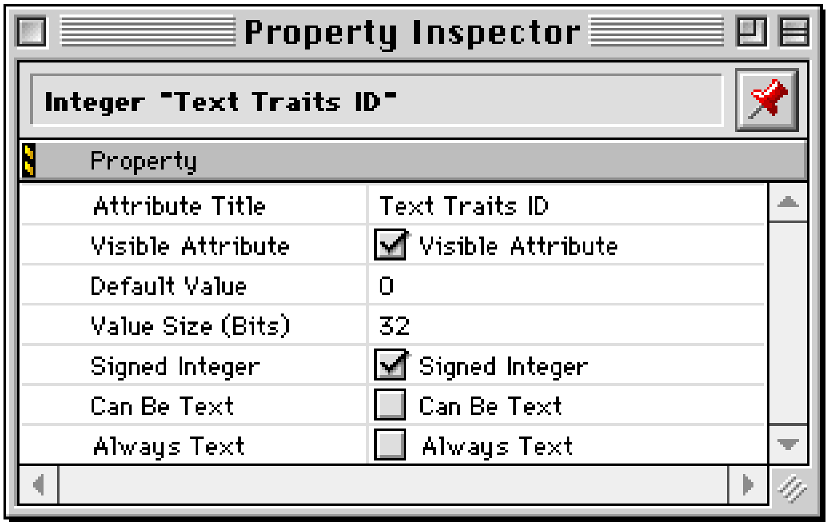
Item	Specifies
Attribute Title	the name of the item in the property inspector
Visible Attribute	whether this item will appear in the property inspector
Initial Value	the default contents of the item
Maximum Length	largest acceptable string (max. 255)

If you turn off the visible attribute, the item does not appear in the property inspector. This prevents you from modifying the default value for the string. You might want to do this to initialize a data member in the class to a constant value.

Editing Integer Properties

Figure 9.7 shows the integer property inspector. Table 9.3 lists the purpose of each property.

Figure 9.7 The integer property inspector



Integer "Text Traits ID"

Property	
Attribute Title	Text Traits ID
Visible Attribute	☑ Visible Attribute
Default Value	0
Value Size (Bits)	32
Signed Integer	☑ Signed Integer
Can Be Text	□ Can Be Text
Always Text	□ Always Text

Table 9.3 Integer properties

Item	Specifies
Attribute Title	the name of the item in the property inspector
Visible Attribute	whether this item will appear in the property inspector
Default Value	the default contents of the item
Value Size (Bits)	number of bits in this integer
Signed Integer	whether this value should be a signed or unsigned integer

Item	Specifies
Can Be Text	whether you accept text values
Always Text	whether you require that the value be treated as text

If you turn off the visible attribute, the item does not appear in the property inspector. This prevents you from modifying the default value for the string. You might want to do this to initialize a data member in the class to a constant value.

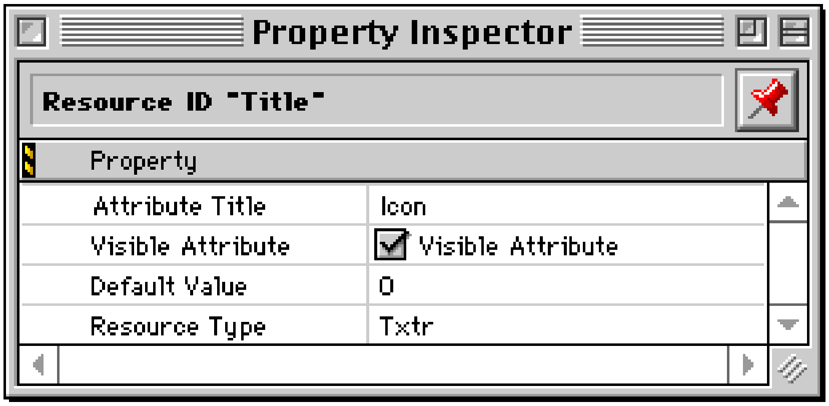
If the Value Size is one bit, Constructor treats the item as if it were a Boolean and displays a check box in the property inspector. The Value Size property must be in the range 1-32.

The text value items control how you can enter data. If these are off, only numbers are allowed in the entry. However, you may wish to have the opportunity to enter data as a string of characters rather than digits. This is particularly true of the OSType data type, which is a string of four characters stored in a 32-bit integer.

Editing Resource ID Properties

Figure 9.8 shows the resource ID property inspector. Table 9.4 lists the purpose of each property.

Figure 9.8 The resource ID property inspector



Resource ID "Title"

Property	
Attribute Title	Icon
Visible Attribute	☑ Visible Attribute
Default Value	0
Resource Type	Txtr

Table 9.4 Resource ID properties

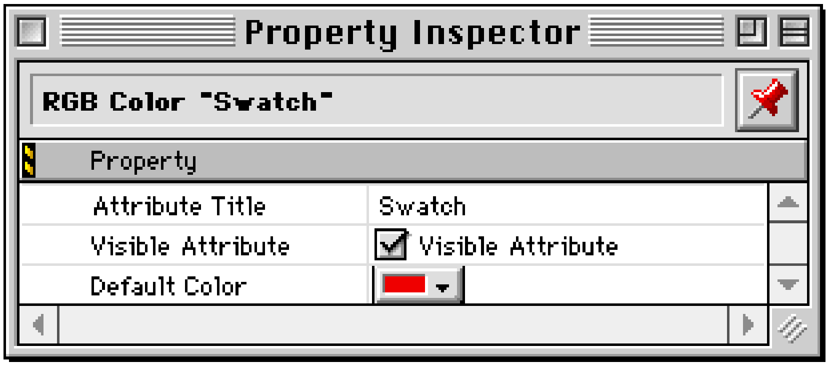
Item	Specifies
Attribute Title	the name of the item in the property inspector
Visible Attribute	whether this item will appear in the property inspector
Default Value	the default contents of the item
Resource Type	the type of the resource this property refers to

If you turn off the visible attribute, the item does not appear in the property inspector. This prevents you from modifying the default value for the string. You might want to do this to initialize a data member in the class to a constant value.

Editing RGB Properties

Figure 9.9 shows the RGB property inspector. Table 9.5 lists the purpose of each property.

Figure 9.9 The RGB property inspector



RGB Color "Swatch"

Property	
Attribute Title	Swatch
Visible Attribute	☑ Visible Attribute
Default Color	🟥 ▾

Table 9.5 RGB Color properties

Item	Specifies
Attribute Title	the name of the item in the property inspector
Visible Attribute	whether this item will appear in the property inspector
Default Color	the initial color for this item

If you turn off the visible attribute, the item does not appear in the property inspector. This prevents you from modifying the default value for the string. You might want to do this to initialize a data member in the class to a constant value.

10

Pane Properties

This chapter is a reference for the properties in each individual pane class as they appear in the Property Inspector Window.

Pane Properties Overview

Every kind of pane has stored data. Some are very simple, others are more complex. You modify and view this data in the property inspector for the individual pane.

This chapter discusses how to work with a property inspector, and how the structure of the window reflects the nature of the pane. The display of data in the Property Inspector Window is highly modular.

The rest of the chapter is a class-by-class reference to the properties for each pane in PowerPlant.

The topics in this chapter are:

- Using a Property Inspector Window—working with a property inspector
- Anatomy of a Property Inspector—the parts of a property inspector, and how each window is described in this chapter
- LPane—the base properties for all panes
- Views—properties for view classes
- LColumnView—properties for control classes
- Simple Panes—properties for pane classes
- Custom Panes—properties for custom panes

In general, the information in this chapter does not tell you the purpose for each data item in a property inspector, or how to use that item to best effect in a PowerPlant application. See *The*

PowerPlant Book for information on class data members, and how to use them in a PowerPlant application.

Using a Property Inspector Window

You open a pane property inspector in Constructor using one of three methods:

- double-click the pane in the layout or hierarchy window
- select the pane and press the Return key
- open the Property Inspector Window and select a pane

The property inspector displays various options that you can modify. The property inspector is, essentially, a modeless dialog box of sorts. You use standard techniques to modify the contents. Use the mouse to change options and enter fields, or navigate through the edit fields with the Tab and Shift-Tab keys.

Changes in a property inspector take effect in the layout and hierarchy windows quickly. Check box and radio button changes show up immediately. Changes in the edit fields appear as soon as you leave the field by pressing Tab, clicking elsewhere on the screen, or closing the property inspector.

When editing, all normal editing functions apply, including Cut, Copy, Paste and Clear. You can navigate through the text with arrow keys. You can extend the text selection by pressing the Shift key as you click the mouse.

You can undo any change that you make by choosing **Undo** from the **Edit** menu. The menu item will be either **Undo Info Change** or **Redo Info Change**.

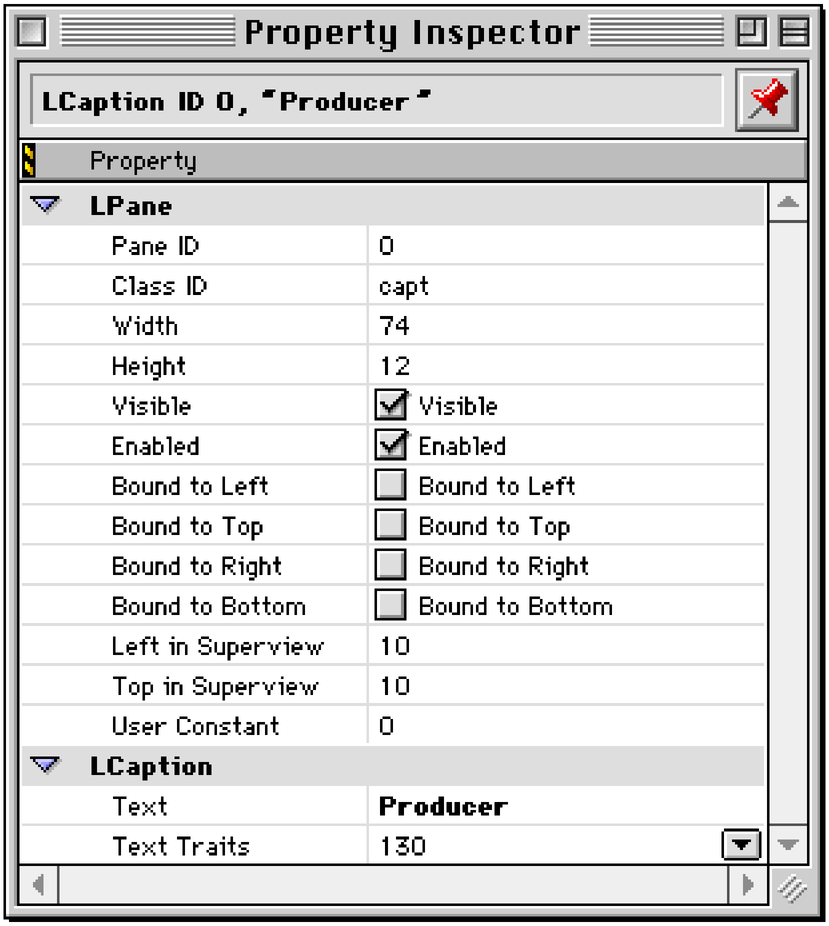
Anatomy of a Property Inspector

With a few exceptions, the property inspector for a particular type of pane reflects its place in the class hierarchy of PowerPlant panes. LPane is the base class for this entire hierarchy, and every type of pane has LPane properties.

Each type of pane usually has additional properties particular to that kind of pane. The items in the property inspector correspond to the data members (including inherited data members) for that pane class in PowerPlant.

For example, Figure 10.1 shows the property inspector for an LCaption object. Notice that the top part of the window has LPane properties, and the bottom part of the window contains the caption-specific data.

Figure 10.1 Property inspector for an LCaption



Property Inspector

LCaption ID 0, “Producer”

Property	
LPane	
Pane ID	0
Class ID	capt
Width	74
Height	12
Visible	☑ Visible
Enabled	☑ Enabled
Bound to Left	□ Bound to Left
Bound to Top	□ Bound to Top
Bound to Right	□ Bound to Right
Bound to Bottom	□ Bound to Bottom
Left in Superview	10
Top in Superview	10
User Constant	0
LCaption	
Text	**Producer**
Text Traits	130

Each property inspector is structured similarly. Properties from LPane appears at the top. Properties specific to the pane appear at the bottom.

In the topics in this chapter, information is presented consistently. The discussion of each class has these sections.

- **Description**—a brief description of the pane
- **Inheritance**—the *visual* class hierarchy for this pane, arranged with base class to the right (does not include non-visual classes)
- **Properties**—the data specific to this pane
- **Special considerations**—extra information and tips regarding this pane's properties
- **See also**—references to other information, usually to *The PowerPlant Book*

For each class the properties section has an illustration showing the data items specific to that class. Each data item is also explained in a table like this:

Table 10.1 Sample table of properties

Item	Specifies
Top/Left	the top left corner of the pane in its containing view

The actual property inspector for any pane is typically built out of the properties from each ancestor class, and the specific properties for the pane in question. To see all the properties for a pane, refer to the discussion for each ancestor class as well as the specific pane of interest.

For those cases where the property inspector of a pane is not the sum of all ancestor panes, that fact is noted in the description of the properties.

LPane

Description The base class for all visual objects in PowerPlant. A pure LPane is not typically added to a PPob resource.

Inheritance None. LPane is the base class from which all PowerPlant visual classes descend.

Properties Table 10.2 lists the property inspector for an LPane. Almost all property inspectors display this data at the top of the window.

Table 10.2 LPane properties

Item	Specifies
Pane ID	the pane's ID number, a long value
Class ID	a 4-character ID unique to each class
Width	the width of the pane
Height	the height of the pane
Visible	whether the pane is visible at runtime
Enabled	whether the pane responds to clicks at runtime
Bound to left, top, right, bottom	how the pane bounds change when its containing view changes size
Top/Left	the top left corner of the pane in its containing view
User Constant	a 32-bit value defined by the programmer

Special considerations Binding affects the pane both at runtime and in the Constructor layout editor. If binding is on for any given side of a pane, that side of the pane will remain the same distance from the edge of the enclosing view when you resize the view.

The default value for pane ID is zero. You must change the ID to a unique value if you call `FindPaneByID()` for the pane. The ID should also be unique if some other pane refers to this pane by ID, such as the ID number of the view contained in an LScroller or LActive Scroller. Otherwise, you can leave the pane ID as zero.

The class ID is used by the URegistrar and UReanimator classes to build the pane at runtime. It corresponds to the `class_ID` enum at the start of each display class's header. Constructor fills this field in for you automatically when you add a pane from the Catalog Window. If you declare a subclass, you must give it a unique class

ID. PowerPlant reserves all IDs that consist entirely of lower case letters. Make sure your class ID has at least one upper-case letter.

Views

The PowerPlant view classes represent panes that can contain other panes. Several are typically used to represent data, however, and do not generally contain other panes. All of these classes derive from LPane via LView.

These classes include:

- LView
- LWindow
- LDialogBox
- LCheckBoxGroupBox
- LGAFocusBorder
- LGrafPortView
- LImageWell
- LScroller
- LActiveScroller
- LOffscreenView
- LPicture
- LTextEdit
- LPrintout
- LPlaceHolder
- LTable
- LTableView
- LHierarchyTable
- LTextHierTable
- LOutlineTable
- LSmallIconTable
- LColumnView
- LTextColumn

LView

Description Base class for all view classes. Also, a generic, simple container for other panes, useful for implementing scrolling and grouping panes in logical containers. This may be a top-level view but can also be added to a PPob.

Inheritance LView -> LPane

Properties Table 10.3 lists the properties specific for LView.

Table 10.3 LView properties

Item	Specifies
Width, Height	define the underlying image size, in 32-bit coordinates
Horiz Scroll Position, Vert Scroll Position	the initial scroll position of the image within its frame, 0/0 means the top left corner of the image and pane frame coincide
Horiz Scroll Unit, Vert Scroll Unit	the number of pixels to scroll when a scroll bar is clicked
Reconcile Overhang	how the image behaves when the view extends below and to the right of the bottom right corner of the image

Special considerations A view may have an image larger than its bounds. In that case, the image may scroll. If the image is scrolled, *Reconcile Overhang* is on, and the view grows down and to the right beyond the bottom right of the image, the image will reposition itself to the bottom right of the view. This also happens if you use `SetImageSize()` to reduce the image and that operation leaves a gap at the bottom or right of the view.

You can make an LView a top-level view. See "Creating a Layout Resource."

See also *The PowerPlant Book* chapter on views.

LWindow

Description A top-level view for a Mac OS window. Can be used for dialog boxes, floating palettes, and document windows.

Inheritance LWindow->LView -> LPane

Properties Table 10.4 lists the properties specific for LWindow. Although LWindow descends from LView and LPane, it has a unique property inspector because of the unique and central importance of a window in the Mac OS.

Table 10.4 LWindow properties

Item	Specifies
Class ID	a 4-character ID unique to each class
Top/Left	the corner of the window on the desktop
Width/Height	the dimensions of the window
Window Proc	the type of window, document, modal, movable modal, or floating
Ref Con	a programmer-defined 32-bit value
Window Title	the window title
Auto Position	use Window Manager auto-positioning
Layer	window layer: regular, modal, or floating
Close Box	whether window has a close box
Title Bar	whether window has a title bar
Resizable	whether window is resizable
Size Box	whether window has a size box
Zoomable	whether window has a zoom box
Initially Visible	whether window is visible on creation
Enabled	whether window is enabled
Targetable	whether this window can be a target

Item	Specifies
Get Select Click	whether window should process the click that activates the window
Hide On Suspend	whether window should disappear when the application goes into the background
Delay Select	whether to respond to clicks when inactive
Erase On Update	whether to erase window on update event
Minimum Width, Minimum Height, Maximum Width, Maximum Height, Standard Width, Standard Height	specify minimum, maximum, and standard sizes for resizing and zooming
User Constant	a programmer-defined 32-bit value
Has WCTB	whether window has wctb resource (color table)
Content Color	color used in window content

Special considerations LWindow is a top-level view. You cannot add an LWindow to an existing PPob. You specify LWindow as the top-level view when you create the PPob resource. See "Creating a Layout Resource."

Some of the information you specify is stored in a WIND resource rather than the PPob. Constructor creates and maintains WIND resources for each LWindow and LDialogBox PPob.

Because windows do not have scrollbars (although views inside windows may scroll), LView scrolling properties are not included.

Any negative number in Maximum or Standard sizes will be adjusted to the appropriate size for the screen by the LWindow functions.

The *Window Kind* pop-up menu allows you to set the type of window for the Mac Toolbox. This automatically adjusts the *WDEF ID* field. Options are: Document Window, Modal dialog box, Movable modal, Modal dialog (no border), Modal dialog (shadow border), Rounded Window, Floating window, and Floating window (sidebar).

See also *The PowerPlant Book* chapter on windows. *PowerPlant Advanced Topics* chapter on drag and drop.

LDialogBox

Description A top-level view for a dialog box.

Inheritance LDialogBox->LWindow -> LView -> LPane

Properties Table 10.5 lists the properties specific for LDialogBox. The other properties are identical to LWindow.

Table 10.5 LDialogBox properties

Item	Specifies
Default Button ID	pane ID number of default button
Cancel Button ID	pane ID number of cancel button

Special considerations LDialogBox is a top-level view. You cannot add an LDialogBox to an existing PPob. You specify LDialogBox as the top-level view when you create the PPob resource. See "Creating a Layout Resource."

See also *The PowerPlant Book* chapter on dialogs.

LCheckBoxGroupBox

Description A view that enables or disables its subviews based on the setting of its own checkbox.

Inheritance LCheckBoxGroupBox->LView -> LPane

Properties Table 10.6 lists the properties in LCheckBoxGroupBox. Its other properties are inherited from LView.

Table 10.6 LCheckBoxGroupBox properties

Item	Specifies
Value Message	a 32-bit value, the control’s value message
Initial Value	the control’s initial value
Group Kind	the kind of group, Primary or Secondary
Text Traits	resource ID of a Txtr resource describing the appearance of this text
Title	the group box’s title

LGAFocusBorder

Description Draws an appropriate border when its sub pane has the focus using the PowerPlant grayscale appearance.

Inheritance LGAFocusBorder->LView -> LPane

Properties Table 10.7 lists the properties in LGAFocusBorder. Its other properties are inherited from LView.

Table 10.7 LGAFocusBorder properties

Item	Specifies
Inset Sub Pane ID	the ID of the sub pane it provides a focus border for
Commander ID	the ID of the view’s commander
Paint Border’s Face	whether or not the border should be drawn between this view’s coordinates and its sub pane.
Frame Inset Sub Pane	whether or not the sub pane is inset
Noth Inset Sub Pane Frame	whether or not to accomodate items within the subpane, such as scrollbars
Noth Face for Growbox	whether or not to accomodate a grow box

Item	Specifies
Notch Width/ Height	the number of pixels to adjust the notch for
Can be Focused?	whether or not the border view can be focused
Has the Focus	whether or not the border view has the focus

LGrafPortView

Description A top-level view that acts as the interface between PowerPlant panes and “foreign” code that knows nothing about PowerPlant. This allows you to use PowerPlant panes in other Mac OS frameworks or in other code such as HyperCard XCMDs.

Inheritance LGrafPortView->LView -> LPane

Properties Same as LView.

Special considerations LGrafPortView is a top-level view. You cannot add an LGrafPortView to an existing PPob. You specify LGrafPortView as the top-level view when you create the PPob resource. See “Creating a Layout Resource.”

LImageWell

Description Provides a view to contain a Mac OS 8 image well.

Inheritance LImageWell->LView -> LPane

Properties Table 10.8 lists its properties. Its other properties are inherited from LView.

Table 10.8 LImageWell properties

Item	Specifies
Value Message	a 32-bit value, the control’s value message
Content Res Id	the resource ID of the image to display in the image well
ContentType	the type of image to display

See also Inside *Macintosh: Mac OS 8 Toolbox Reference* and *Inside Macintosh: Mac OS 8 Human Interface Guidelines*

LScroller

Description A view that contains and scrolls another view. LScroller is *not* a scroll bar, although the LScroller creates its own scroll bars automatically.

Inheritance LScroller->LView -> LPane

Properties Table 10.9 lists the properties specific for this pane. Although LScroller inherits from LView, the LScroller property inspector does not include LView properties. An LScroller contains another view, and does not require LView properties. It is unlikely that the LScroller will itself be scrolled.

Table 10.9 LScroller properties

Item	Specifies
Left Indent	pixels from left edge of view
Right Indent	pixels from right edge of view
Top Indent	pixels from top edge of view
Bottom Indent	pixels from bottom edge of view
Scrolling View ID	the pane ID of the view contained in this scroller

Special considerations A value of -1 for left or top indent means that the corresponding scroll bar does not appear. The default values of 15 pixels for right and bottom allow room for a resize box in the bottom right corner of the view.

To have your views scroll, create an LScroller and put an LView or derived type *inside* the scroller. The LScroller coordinates the scrollbars that it creates and the scroll position of the image which is contained within it. If you have a number of panes that you want to scroll together, use an LView object to contain them, and put the view inside the LScroller.

When setting the size of an LScroller, typically you set it to extend one pixel beyond the bounds of the view it encloses, on all four sides. The location and size of the LScroller define the *outside* of the scroller, which should correspond to the outside of the view being scrolled.

See also *The PowerPlant Book* chapter on views.

LActiveScroller

Description A scrolling view with active update while the user moves the scroll thumb.

Inheritance LActiveScroller->LScroller -> LView -> LPane

Properties Same as LScroller.

Special considerations Same as LScroller.

See also *The PowerPlant Book* chapter on views.

LOffscreenView

Description A view that automatically draws its contents to an offscreen buffer before drawing the image to the screen.

Inheritance LOffscreenView->LView -> LPane

Properties Same as LView.

Special considerations Used for drawing to an offscreen bitmap.

See also *The PowerPlant Book* chapter on views.

LPicture

Description A view that contains a PICT.

Inheritance LPicture->LView -> LPane

Properties Table 10.10 lists the properties specific for this pane.

Table 10.10 LPicture properties

Item	Specifies
PICT Resource ID	resource ID of picture in this pane

Special considerations If the PICT resource is in the same file as the PPob, Constructor displays the picture in the pane.

LTextEdit

Description A PowerPlant wrapper pane for the Mac OS TextEdit. Useful for simple text processing.

Inheritance LTextEdit->LView -> LPane

Properties Table 10.11 lists the properties specific for this pane.

Table 10.11 LTextEdit properties

Item	Specifies
Text Traits	resource ID of a Txtr resource describing the appearance of this text
Initial Text Resource ID	resource ID of a TEXT resource containing default text

Special considerations LTextEdit is a simple wrapper for the Toolbox TextEdit features and suffers from the same limitations, such as a 32K limit on the amoung of text.

LTextEdit supports text in a single style only.

See also LEditField, *The PowerPlant Book* chapter on views.

LPrintout

Description Top-level view for printing.

Inheritance LPrintout->LView -> LPane

Properties Table 10.12 lists the properties specific for LPrintout. and represents all the data required for an LPrintout. Other pane and view properties are not necessary.

Table 10.12 LPrintout properties

Item	Specifies
Width/Height	dimensions of the printing area, in pixels
User Constant	a programmer-defined 32-bit value
Class ID	a 4-character ID unique to each class
Page Numbering	page numbering sequence in multi-page documents
Enabled	whether pane is enabled
Active	whether pane is active

Special considerations LPrintout is a top-level view. You cannot add an LPrintout to an existing PPob. You specify LPrintout as the top-level view when you create the PPob resource.

See also *The PowerPlant Book* chapter on printing.

LPlaceHolder

Description Holds another view for printing.

Inheritance LPlaceholder->LView -> LPane

Properties Table 10.13 lists the properties specific for LPlaceholder.

Table 10.13 LPlaceholder properties

Item	Specifies
Horizontal	how the occupant view is aligned horizontally
Vertical	how the occupant view is aligned vertically

Special considerations This view is used for printing only. It holds a regular view temporarily, and then returns the occupant to its original location.

See also *The PowerPlant Book* chapter on printing.

LTable

Description A view containing cells with data of a single length, all cells the same size, and that allows single cell selection.

Inheritance LTable->LView -> LPane

Properties Table 10.14 lists the properties specific for LTable.

Table 10.14 LTable properties

Item	Specifies
Number of Rows	number of rows in the table
Number of Columns	number of columns in the table
Row Height	height of a row, in pixels
Column Width	width of a column, in pixels
Cell Data Size	number of bytes of data in each cell

Special considerations All of the LTable items are 32-bit values.

LTable is being superseded by LTableView, which is more flexible and allows data of varying size.

LTableView

Description A very flexible view class that allows for cells containing data of varying length, cells of varying size, and multiple cell selection.

Inheritance LTableView->LView -> LPane

Properties Same as LView.

Special considerations You vary behavior of an LTableView class by attaching various helper objects.

See also Refer to LView. See *PowerPlant Advanced Topics* chapter on tables.

LHierarchyTable

Description A table with collapsible rows.

Inheritance LHierarchyTable->LTableView -> LView -> LPane

Properties Same as LView.

Special considerations None.

See also Refer to LView. See *PowerPlant Advanced Topics* chapter on tables.

LTextHierTable

Description A hierarchy table containing text data.

Inheritance LTextHierTable->LHierarchyTable -> LTableView -> LView -> LPane

Properties Same as LView.

Special considerations None.

See also Refer to LView. See *PowerPlant Advanced Topics* chapter on tables.

LOutlineTable

Description A table supporting expandable hierarchies.

Inheritance LOutlineTable -> LTableView -> LView -> LPane

Properties Table 10.15 lists the properties specific for LOutlineTable.

Table 10.15 **LOutlineTable properties**

Item	Specifies
Text Traits	Resource ID of a text traits resource
First Level Indent	pixels to indent subsidiary entries

Special considerations None.

See also Refer to LView. See *PowerPlant Advanced Topics* chapter on tables.

LSmallIconTable

Description A table containing small icons.

Inheritance LSmallIconTable->LTableView -> LView -> LPane

Properties Same as LView.

Special considerations This is really just a demonstration class of little direct utility.

See also Refer to LView. See *PowerPlant Advanced Topics* chapter on tables.

LColumnView

Description A table with one column, with data of a single length, all cells the same size.

Inheritance LColumnView->LTableView -> LView -> LPane

Properties Table 10.16 lists the properties specific for LColumnView.

Table 10.16 LColumnView properties

Item	Specifies
Column Width	width of a column, in pixels
Row Height	height of a row, in pixels
Use Single Selector	allow one cell selection only
Use Drag Select	allow dragging to select cells
Data Size	number of bytes of data in each cell
Double-Click Message	message sent in response to double-click
Selection Message	message sent when selection range changes

Special considerations In typical use, you would subclass LColumnView to display data of the appropriate type. You cannot use LColumnView directly.

You specify the number of rows at runtime.

When *Use Single Selector* is off, you can create a multi-selector to support multiple and discontiguous cell selection.

See also *PowerPlant Advanced Topics* chapter on tables.

LTextColumn

Description An LColumnView to display text.

Inheritance LTextColumn->LColumnView->LTableView -> LView -> LPane

Properties Table 10.17 lists the properties specific for LTextColumn.

Table 10.17 LTextColumn properties

Item	Specifies
Text Traits ID	resource ID of a Txtr resource describing the appearance of this text
STR# Resource ID	resource ID of an STR# resource containing initial text for cells

Special considerations PowerPlant initializes the table with the contents of the STR# resource.

See also *PowerPlant Advanced Topics* chapter on tables.

Controls

The control classes represent buttons, check boxes, scroll bars and pop-up menus. All of these classes derive from LPane via LControl.

The control classes are:

- LControl
- LStdControl
- LStdButton
- LStdCheckBox
- LStdRadioButton
- LStdPopupMenu
- LButton

- LCicnButton
- LToggleButton
- LTextButton

LControl

The property inspectors for several derived control classes include the properties for LControl.

Description LControl is the base class for all other control classes.

Inheritance LControl->LPane

Properties Table 10.18 lists the properties specific for LControl.

Table 10.18 LControl properties

Item	Specifies
Value Message	a 32-bit value, the control's value message
Initial Value	the control's initial value
Minimum	the control's minimum value
Maximum	the control's maximum value

Special considerations Because LControl does not have a visual representation, it will not be visible in the layout window unless you display pane edges. See "Displaying pane edges."

Constructor automatically makes a RidL resource (list of all the control IDs) for all views that contain controls. You can use this to link listeners (such as Dialog Boxes) to control broadcasters.

See also *The PowerPlant Book* chapter on controls.

LStdControl

Description A generic, standard Mac OS control. LStdControl represents the base class for other standard Mac OS controls, and is used directly for scroll bars.

Inheritance LStdControl->LControl -> LPane

Properties Table 10.19 lists the properties specific for LStdControl.

Table 10.19 LStdControl properties

Item	Specifies
Control Kind	the Mac OS control kind
Text Traits	resource ID of a Txtr resource describing the appearance of this text
Title	the text in the control's name or label
Toolbox Ref Con	a programmer-specified 32-bit value

Special considerations Some controls, such as scroll bars, have no title. Scroll bars in PowerPlant are LStdControl objects. The default *Control Kind* value of 16 represents a scroll bar in the Mac OS Control Manager.

The LScroller class makes its own scroll bars using LStdControl.

You may want to use the LStdControl class when you are working with a custom CDEF resource. Set the *Control Kind* field to CDEF ID plus the variation.

See also *The PowerPlant Book* chapter on controls.

LStdButton

Description A standard Mac OS push button.

Inheritance LStdButton->LStdControl -> LControl -> LPane

Properties Table 10.20 lists the properties specific for LStdButton. The LStdButton property inspector is a combination of these properties and LPane.

Table 10.20 LStdButton properties

Item	Specifies
Value Message	a 32-bit value, the control's value message
Control Kind	the Mac OS control kind

Item	Specifies
Text Traits	resource ID of a Txtr resource describing the appearance of this text
Title	text in the button
Toolbox Ref Con	a programmer-specified 32-bit value

Special considerations The *Control Kind* value for a standard button is 0.

See also *The PowerPlant Book* chapter on controls.

LStdCheckBox

Description A standard Mac OS check box.

Inheritance LStdCheckBox->LStdControl -> LControl -> LPane

Properties Table 10.21 lists the properties specific for LStdCheckBox. The LStdCheckBox property inspector is a combination of these properties and LPane.

Table 10.21 LStdCheckBox properties

Item	Specifies
Value Message	a 32-bit value, the control’s value message
Initially checked	initial value of check box
Control Kind	the Mac OS control kind
Text Traits	resource ID of a Txtr resource describing the appearance of this text
Title	text in the check box
Toolbox Ref Con	a programmer-specified 32-bit value

Special considerations The *Control Kind* value for a check box should be 1.

See also *The PowerPlant Book* chapter on controls.

LStdRadioButton

Description A standard Mac OS radio button.

Inheritance LStdRadioButton->LStdControl -> LControl -> LPane

Properties Table 10.22 lists the properties specific for LStdRadioButton. The LStdRadioButton property inspector is a combination of these properties and LPane.

Table 10.22 LStdRadioButton properties

Item	Specifies
Value Message	a 32-bit value, the control’s value message
Initial Value	initial value of check box
Control Kind	the Mac OS control kind
Text Traits	resource ID of a Txtr resource describing the appearance of this text
Title	text in the radio button
Toolbox Ref Con	a programmer-specified 32-bit value

Special considerations The *Control Kind* value for a radio button should be 2.

See also *The PowerPlant Book* chapter on controls. For information about radio groups, see “Working With Radio Groups.”

LStdPopupMenu

Description A standard Mac OS pop-up menu.

Inheritance LStdPopupMenu->LStdControl -> LControl -> LPane

Properties Table 10.23 lists the properties specific for LStdPopupMenu. The LStdPopupMenu property inspector is a combination of these properties and LPane.

Table 10.23 LStdPopupMenu properties

Item	Specifies
Value Message	a 32-bit value, the control’s value message
Title Alignment	alignment options for menu title text
Menu Resource ID	resource ID for pop-up menu
Title Width	width of menu title in pixels
Resource List	that the pop-up menu is a list of resource items, such as fonts or sounds
Fixed Width	specifies that the menu should be drawn exactly the same width as the specified pane
Text Traits	resource ID of a Txtr resource describing the appearance of the menu items
Title	menu title text
Resource List Type	4-character resource type if pop-up menu is a resource list
Initial Menu Item	number of default menu item

Special considerations The *Title Style* and *Title Placement* options have no effect on the pop-up menu items.

The *Title Width* specifies the width of the menu title as part of the entire *Width* field (an LPane data item). The difference between *Width* and *Title Width* is the space available for menu items. You may need to experiment with different widths to display the menu properly at runtime.

You must create a MENU resource that contains the text of the menu items for the pop-up menu.

See also *The PowerPlant Book* chapter on controls.

LButton

Description A graphical button containing a PICT or icon.

Inheritance LButton->LControl -> LPane

Properties Table 10.24 lists the properties specific for LButton.

Table 10.24 LButton properties

Item	Specifies
Graphics Type	resource type of image in the button
Normal Graphic ID	resource ID of the picture or icon of the button in the unpushed state
Pushed Graphic ID	resource ID of the picture or icon of the button in the pushed state

Special considerations Choose a resource type from the pop-up menu. Resource types available are ICN#, ICON, and PICT.

See also *The PowerPlant Book* chapter on controls.

LCicnButton

Description A graphical button containing a cicn resource.

Inheritance LCicnButton->LControl -> LPane

Properties Table 10.25 lists the properties specific for LCicnButton.

Table 10.25 LCicnButton properties

Item	Specifies
Normal Graphic ID	resource ID of the cicn resource showing the button in the unpushed state
Pushed Graphic ID	resource ID of the cicn resource showing the button in the pushed state

Special considerations None.

See also *The PowerPlant Book* chapter on controls.

LToggleButton

Description A button with on, off, and transition states.

Inheritance LToggleButton->LControl -> LPane

Properties Table 10.26 lists the properties specific for LToggleButton.

Table 10.26 LToggleButton properties

Item	Specifies
Graphics Type	resource type of image in the button
On Graphic ID	resource ID of the picture or icon of the button in the on state
On-Clicked Graphic	resource ID of the picture or icon of the button clicked while in the on state
Off Graphic ID	resource ID of the picture or icon of the button in the off state
Off-Clicked Graphic	resource ID of the picture or icon of the button clicked while in the off state
Transition Graphic	resource ID of the picture or icon of the button while changing state

Special considerations Choose a resource type from the pop-up menu. Resource types available are ICN#, ICON, and PICT.

See also *The PowerPlant Book* chapter on controls.

LTextButton

Description A button with textual rather than graphical content. Typically used as a column head to control sort order.

Inheritance LTextButton->LControl -> LPane

Properties Table 10.27 lists the properties specific for LTextButton. The LTextButton property inspector is a combination of these properties and LPane.

Table 10.27 LTextButton properties

Item	Specifies
Value Message	a 32-bit value, the control's value message
Initial Value	initial value of the text button
Button Title	text in the button
Text Traits	resource ID of a Txtr resource describing the appearance of this text
Selected Text Style	style options when button is in the on state

Special considerations LTextButtons are mutually exclusive buttons and may be added to radio groups. See "Working With Radio Groups."

See also *The PowerPlant Book* chapter on controls.

Simple Panes

The simple pane classes represent a variety of visual objects. Most derive directly from LPane. All of them derive from LPane without inheriting from either LView or LControl.

The simple pane classes are:

- LIconPane
- LCaption
- LGroupBox
- LEditField
- LListBox

LIconPane

Description A pane containing an icon.

Inheritance LIconPane->LPane

Properties Table 10.28 lists the properties specific for LIconPane.

Table 10.28 LIconPane properties

Item	Specifies
Icon ID	resource ID of icon in this pane

Special considerations The LIconPane is for display only. It does not respond to clicks. To use an icon in a button, see LButton, LCicnButton, or LToggleButton.

See also *The PowerPlant Book* chapter on panes.

LCaption

Description A pane containing static text.

Inheritance LCaption->LPane

Properties Table 10.28 lists the properties specific for LCaption.

Table 10.29 LCaption properties

Item	Specifies
Text	static text in the pane, limited to 255 characters
Text Traits	resource ID of a Txtr resource describing the appearance of this text

Special considerations None.

See also *The PowerPlant Book* chapter on panes.

LGroupBox

Description A titled frame that groups a set of elements visually.

Inheritance LGroupBox->LCaption -> LPane

Properties Same as LCaption.

Special considerations None.

See also *The PowerPlant Book* chapter on panes.

LEditField

Description An editable text field, typically in a dialog box.

Inheritance LEditField->LPane

Properties Table 10.30 lists the properties specific for LEditField.

Table 10.30 LEditField properties

Item	Specifies
Initial Text	text in the pane
Text Traits	resource ID of a Txtr resource describing the appearance of this text
Max. Characters	number of characters allowed in the field, must be 255 or less
Has Box	whether the field has a visible frame
Has Word Wrap	whether the field uses word wrap
Auto Scroll	whether to scroll the text when the user types beyond the end of the field
Text Buffer	whether the field uses text buffering
Outline Highlight	whether the field shows an outline highlight when the window is inactive
Inline Input	whether the field uses inline input
Uses Text Services	whether the field uses text services
Key Filter	the key filter used for this field

Special considerations To ensure that the field big enough to fit the data, set the text traits and then type in the longest expected data. You can see the result in the layout window.

The available key filters are: None, Integer, Alpha-numeric, and Printing Character. See *The PowerPlant Book* chapter on utilities for information on key filters.

The *Text Buffering*, *Inline Input*, and *Text Services* features are used for two-byte character sets and ideographic languages.

See also *The PowerPlant Book* chapter on panes. See LTextEdit for scrolling text fields.

For information on topics such as inline input and text services, see *Inside Macintosh: Text*.

LListBox

Description A pane containing a single-column Mac OS List Manager list.

Inheritance LListBox->LPane

Properties Table 10.31 lists the properties specific for LListBox.

Table 10.31 LListBox properties

Item	Specifies
Horizontal Scrollbar	whether the pane has a horizontal scrollbar
Vertical Scrollbar	whether the pane has a vertical scrollbar
Has Grow Box	whether the pane has a grow box
Has Focus Box	whether the pane uses a focus box around the current cell in the list
Double-Click Message	32-bit value message sent when pane is double-clicked
Text Traits	resource ID of a Txtr resource describing the appearance of this text
LDEF ID	resource ID of LDEF for the list
Initial Items	initial contents of the list box

Special considerations LListBox is a wrapper for the Mac OS List Manager. LTableView classes are more flexible.

PowerPlant allows multiple columns. However, you must add them programmatically with the Mac Toolbox. These lists come with their own scrollbars. You should not implement them in Constructor or PowerPlant.

See also *The PowerPlant Book* chapter on panes.

Custom Panes

You regularly declare and define custom classes derived from the PowerPlant visual hierarchy. A custom class may have modified or additional behaviors (member functions) and/or additional data members.

If a custom class has no additional data members, you can use the standard Constructor panes to represent the class. When you set the properties for that pane, simply modify the Class ID field to reflect the Class ID you declared for your custom class. See "Simple Custom Panes."

If a custom class has additional data members, you can add them to the property inspector easily. Essentially, you create a CTYP resource. When that resource is available to Constructor, that pane appears in the Catalog Window. See "What a CTYP Resource Does."

You can then use the pane just like any other pane.

11

Constructor Menu Reference

This chapter lists each menu item in Constructor and describes its purpose.

Constructor Menu Reference Overview

The Constructor menus include:

- Apple Menu
- File Menu
- Edit Menu
- Window Menu
- Arrange Menu
- Layout Menu
- Options Menu
- Colors Menu
- Font Menu
- Style Menu

The **Apple**, **File**, **Edit**, and **Window** menus are visible at all times.

The **Arrange** and **Layout** menus appear when a PPob editor is active. The **Options** and **Colors** menus appear when a bitmap editor is active.

Apple Menu

The **Apple** menu contains one Constructor-related item.

About Constructor…

Choose this item to see the way-cool About Box, and the credits for the fine people who put Constructor together for you.

File Menu

Use the **File** menu commands for managing Constructor project files. The Constructor **File** menu has all the usual file-related commands you are accustomed to seeing in a Mac OS application.

New Project File

Creates a new Constructor project file.

See “Constructor Project Files.”

Open Project File…

Uses the standard dialog to open an existing Constructor project file.

See “Constructor Project Files.”

Close

Closes the active window. If you close the project window and there are unsaved changes, you will be given the chance to save changes.

This item is enabled when there is a window open.

See “Constructor Project Files.”

Save

Saves an existing project file. If no file exists, this command displays the standard dialog so you can specify the folder where you want to store the file. By convention, Constructor project file names end with .ppob.

This item is enabled when there is a project open that has been changed without being saved.

See “Constructor Project Files.”

Save As…

Save a file to a new name. This command displays a standard dialog so you can specify the folder in which you want to store the file. By convention, Constructor project files end with .ppob.

This item is enabled when there is a project open.

For additional information concerning the use of and need for flattened resource files on Mac OS X, refer to “A Word on Flattened Resources.”

See “Constructor Project Files.”

Page Setup…

Displays the standard dialog for choosing page setup options such as paper size and page orientation.

Print…

Prints the active Constructor window.

This item is enabled when there is a document open.

Quit…

Quits Constructor. You will be asked to confirm this operation if any open files have been changed without saving.

Edit Menu

In addition to the standard **Edit** menu commands for copy and paste, the Constructor **Edit** menu contains commands that allow you to create and modify resources, panes, menus, and so forth.

The menu item text usually includes a word such as “resource,” “pane,” “menu,” or other term to clearly reflect the nature of the item you are acting upon. In some cases, there may be two or more different kinds of items listed. For example, when editing a menu resource, the **Edit** menu has a few commands to create new items related to menus, including **New Menu**, **New Menu Item**, and **New Separator Item**.

Menu items are enabled and disabled as appropriate.

Undo/Redo/Can’t Undo Action

The actual text varies depending upon the most recent action. Undo/Redo toggle so that you can repeatedly switch between undoing and redoing a single action.

Constructor supports single-level undo. Once you perform a new action, you lose the ability to undo or redo the previous action.

Cut Item

Removes the selected items from the active window and puts them on the clipboard.

Copy Item

Makes a copy of the selected items in the active window and puts them on the clipboard.

Paste Item

Makes a copy of items in the clipboard and puts them in the active window.

Clear Item

Deletes selected items from the active window.

Select All

Selects all items in the active window.

Duplicate Item

Makes a duplicate of the selected items in the active window.

New Item

Creates a new item. The menu text reflects the exact nature of the item. It might be a resource, a menu, a menu item, and so forth.

This item is disabled for panes. To create a new pane, drag the correct pane from the Catalog Window.

Edit Item

This command opens a resource editor. This could be the view editor, the menu editor, the text traits editor, and so forth. See "Opening a Resource for Editing."

When an editor is already open and an item selected in the editor (such as a menu item), this command opens the property inspector window for the item.

Window Menu

The **Window** menu contains items that display or activate various Constructor windows. This menu is always available, but certain items may be disabled when appropriate.

Property Inspector

Displays or activates the Property Inspector Window to display attributes for a resource, menu, menu item, or pane. To close the Property Inspector Window, click in the window's close box.

See "Property Inspector Window."

Catalog

Displays or activates the Catalog Window. To close the Catalog Window, click in the window's close box.

See "Catalog Window."

Alignment Palette

Display or activate the alignment palette. To hide the palette, click in the palette's close box.

See “Aligning panes,” “Distributing panes.”

Zoom Window

Zoom the window, has the same effect as clicking the window’s zoom box.

Arrange Menu

The **Arrange** menu appears when a PPob editor is active. This menu provides tools for aligning and distributing panes in the layout.

Arrange Objects…

Open the Arrange Objects dialog.

See “Aligning panes,” “Distributing panes.”

Align Left Edges

Aligns selected panes along the left of the leftmost selected pane.

See “Aligning panes.”

Align Horizontal Centers

Aligns selected panes halfway between the left of the leftmost selected pane and the right of the rightmost selected pane.

See “Aligning panes.”

Align Right Edges

Aligns selected panes along the right of the rightmost selected pane.

See “Aligning panes.”

Spread Horizontally

Distributes selected panes evenly from the left of the leftmost selected pane to the right of the rightmost selected pane.

See “Distributing panes.”

Spread Horizontally in Container

Distributes selected panes evenly from the left edge to the right edge of the container that holds all the panes.

See “Distributing panes.”

Align Top Edges

Aligns selected panes along the top of the topmost selected pane.

See “Aligning panes.”

Align Vertical Centers

Aligns selected panes halfway between the top of the topmost selected pane and the bottom of the bottommost selected pane.

See “Aligning panes.”

Align Bottom Edges

Aligns selected panes along the bottom of the bottommost selected pane.

See “Aligning panes.”

Spread Vertically

Distributes selected panes evenly from the top of the topmost selected pane to the bottom of the bottommost selected pane.

See “Distributing panes.”

Spread Vertically in Container

Distributes selected panes from the top edge to the bottom edge of the container that holds all the panes.

See “Distributing panes.”

Make Radio Group

Creates a radio group.

See “Working With Radio Groups.”

Make Tab Group

Creates a tab group.

See “Working With Tab Groups.”

Layout Menu

The **Layout** menu is visible when a PPob editor is active. Items in this menu control the layout grid, and pane visibility. Most items in the menu are enabled at all times.

Show/Hide Grid

Shows or hides a dotted rectangular grid, like graph paper, to help you align and arrange your panes. You can specify the size of the grid and whether to “snap” the panes to the grid.

See “Using a grid.”

Snap/Don’t Snap To Grid

Toggles the snap-to-grid setting. When this setting is on, dragging and dropping a pane causes it to snap to the nearest grid line.

See “Using a grid.”

Edit Grid

Displays the Grid dialog that allows you to specify how large to make the grid rectangles (in pixels).

See “Using a grid.”

Show/Hide Pane IDs

Show or hide Pane IDs in the upper right corner of the pane.

See “Displaying pane ID.”

Show/Hide Pane Edges

Show or hide the edges of panes which are otherwise invisible, such as LView and LControl. Showing the edge also shows the size of unboxed panes such as LCaption and LStdRadioButton, making it easier to adjust them and avoid overlaps.

See “Displaying pane edges.”

Show/Hide Invisible Panes

Show or hide panes that have the *Visible* check box in the property inspector deselected.

See “Displaying invisible panes.”

Show Object Hierarchy

If the Hierarchy window is not open, this item opens the window. If the window is open but inactive, this item activates the window. To close the Hierarchy window, click the window’s close box.

This item is disabled if the Hierarchy window is open and active.

See “Hierarchy Window.”

Options Menu

The **Options** menu is visible when a bitmap editor is active. Most items in the menu are enabled when pixels are selected in the editor.

Flip Vertical

Turn the selected pixels upside down (rotate 180°). See “Rotating and Flipping Pixels.”

Flip Horizontal

Swap selected pixels left and right. This creates a mirror image of the pixels. See “Rotating and Flipping Pixels.”

Rotate Right

Turn the selected pixels 90° clockwise. See “Rotating and Flipping Pixels.”

Rotate Left

Turn the selected pixels 90° counterclockwise. See “Rotating and Flipping Pixels.”

Make Transparent

Render background-color within selected pixels transparent so that any bitmap beneath shows through. See “Using Transparency.”

Colors Menu

The **Colors** menu appears when a bitmap editor is active. This menu is enabled when the resource being edited supports more than one color palette. Items in the menu are enabled when appropriate.

Most items in this menu represent color palettes for use when editing bitmaps. The currently active palette has a check mark.

See “Choosing a Color Palette.”

Recommended Icon Colors

Use the standard 34 colors for icon design. See “Choosing a Color Palette.”

256 Colors

Use the standard 256 colors palette. See “Choosing a Color Palette.”

256 Grays

Use the standard 256 grays palette. See “Choosing a Color Palette.”

16 Colors

Use the standard 16 colors palette. See “Choosing a Color Palette.”

16 Grays

Use the standard 16 grays palette. See “Choosing a Color Palette.”

4 Grays

Use the standard 4 grays palette. See “Choosing a Color Palette.”

Use Color Picker

When this item is checked, clicking the Color Tool displays the standard color picker dialog instead of a color palette. See “Color Tool.” See “Choosing a Color Palette.”

Black & White

Set the foreground color to black and the background color to white. See “Setting Colors and Patterns.”

Swap Fore & Back Colors

Set the foreground color to the current background color. Set the background color to the current foreground color. See “Setting Colors and Patterns.”

Recolor To Current Table

Draw the current bitmap using the current color palette. See “Choosing a Color Palette.”

Font Menu

This menu appears when the bitmap editor is active. The font menu lists all available fonts on your computer. Use the items in menu when entering text in a bitmap.

Style Menu

This menu appears when the bitmap editor is active. The Style menu lists contains commands to control typographic style, justification, and font size. Use the items in this menu when entering text in a bitmap.

Index