## Help contents

Click a topic listed below to see information about that topic. Read **Essentials** if you are new to Resource Workshop or to this version of Resource Workshop.

Essentials for working well with Resource Workshop.

Tasks with step-by-step directions.

Resources that you create with Resource Workshop.

Menu commands reference.


Resource Script Language reference.

Glossary of terms.

## Essentials

These Help topics tell you the essentials for using Resource Workshop. Before attempting any tasks, take the time to read the following topics:

Getting started
Borland resource command line tools
Differences between Borland and Microsoft compilers
Undoing mistakes
Contacting Borland

## Tasks

Click a topic listed below to see information about that topic.

**Identifiers**
Working with identifiers

**Projects**
Adding a linked resource to a project
Adding an embedded resource to a project
Creating a new project
Opening an existing project
Managing your project - the Project window

**Resources**
Creating a font resource
Creating a new accelerator table
Creating a new bitmap
Creating a new cursor
Creating a new dialog box
Creating a new icon
Creating a new menu
Creating a new string table
Creating a resource type
Saving resources and resource files
Working with resources

# Glossary

**- A -**

accelerator
Accelerator editor
ACCELERATOR (resource script)
Airbrush tool
Alignment palette
Attribute pane (Accelerator editor)

**- B -**

background color
binary resource statement
bitmap
Black Frame tool
Black Rectangle tool

**- C -**

Caption window
Check Box tool
Check Box tool (BWCC)
Colors palette
Combo Box tool
constant declaration
controls
cursor

**- D -**

#define statement
DIALOG (resource script)
dialog box
Dialog editor
dialog units
directives
DLL file

**- E -**

Edit Text tool
Ellipse tool
Eraser tool
executable file

**- F -**

Filled Ellipse tool
Filled Rectangle tool
Filled Rounded Rectangle tool
font
font header
foreground color

**- G -**

grid
Group Box tool
Group Shade tool (BWCC)

**- H -**

Hand tool
Horizontal Dip tool (BWCC)
Horizontal Scroll Bar tool

**- I -**

icon
Icon tool
Icon window
identifier files
identifiers
inverted area

**- L -**

Line tool
List Box tool

**- M -**

menu
MENU (resource script)
Menu editor
multiple-line statement

**- O -**

octal values
Outline pane (Accelerator editor)
Outline pane (Menu editor)

**- P -**

Paintbrush tool
Paint Can tool
Paint editor
Pen tool
Pick Rectangle tool
project
project file
Project window
Push Button tool
Push Button tool (BWCC)

**- R -**

Radio Button tool
Radio Button tool (BWCC)
RCDATA
Rectangle tool
resource
resource editor
resource file
resource file type
resource ID
resource script
Rounded Rectangle tool

**- S -**

Scissors tool

## cursor

A bitmapped image 32x32 pixels in size that represents the current location of the mouse pointer on the screen, as well as its current function.

## bitmap

A graphical image. Windows programs use bitmaps to represent scroll bar arrows, and the Minimize and Maximize buttons, for example. You can also use bitmaps for splash screens or the Borland Windows Custom Control button icons.

## font

A collection of data used to draw individual characters on an output device.

## icon

A bitmapped image, usually 32x32 pixels in size. Windows programs typically use customized icons to represent minimized windows.

### menu

An item on the menu bar or in a menu command.

### string table

A table that holds error messages, prompts, or any other text strings your application needs to display.

## dialog box

A pop-up window that lets the user specify information (files to open, colors to display, text to search for, and so on).

## accelerator

A key combination a user presses to perform a task in an application. It is a substitute for a menu command.

## user-defined resource

A resource that you define. You first create the resource type, then add resources of this type to your project.

## RCDATA

A resource script statement you use to include any type of data directly in the resource compiler file.

### window pane

In the Paint editor, the way you look at two different views of the image you're creating or editing.

## Paint editor

The editor you use to create or edit any bitmapped resource, including:

- icons
- cursors
- bitmaps
- fonts

## Pick Rectangle tool

The Paint editor tool you use to select a rectangular area of a graphical image for copying, moving, or deleting.

## Scissors tool

The Paint editor tool you use to select any shaped area of a graphical image.

## grid

Lines that you can display on an image in the Paint editor. Each square of the grid represents a single pixel.

## Airbrush tool

The Paint editor tool you use to paint free-form patterns on an image.

## Filled Ellipse tool

The Paint editor tool you use to paint an ellipse-shaped filled-in frame on an image.

## Filled Rectangle tool

The Paint editor tool you use to paint a rectangular filled-in frame on an image.

## Filled Rounded Rectangle tool

The Paint editor tool you use to paint a filled-in frame, in the shape of a rounded rectangle, on an image.

## Paintbrush tool

The Paint editor tool you use to paint free-form patterns on an image.

## Zoom tool

The Paint editor tool you use to zoom an entire image or a part of it.

## Eraser tool

The Paint editor tool you use to erase an entire image or a part of it.

## Pen tool

The Paint editor tool you use to paint free-form lines and shapes on an image.

## Paint Can tool

The Paint editor tool you use to fill an area of your image with the currently selected color.

## Line tool

The Paint editor tool you use to paint straight lines on an image.

## Text tool

The Paint editor tool you use to add text to an image.

## Ellipse tool

The Paint editor tool you use to paint an ellipse-shaped empty frame in an image.

## Rectangle tool

The Paint editor tool you use to paint a rectangular empty frame in an image.

## Rounded Rectangle tool

The Paint editor tool you use to paint an empty frame, shaped like a rounded rectangle, in an image.

## Colors palette

The Paint editor palette you use to select the colors in your image. You can work with a Colors palette even if your image is black and white.

## foreground color

The color in the foreground of your image. It's typically one of the colors you use to create the features of your resource.

## background color

The color that appears to underlie your image. It's also the color that's left behind when you select an area and delete or move it.

## resource script

The source script for your resources. Each resource script statement specifies a resource to be included in your executable file.

### text editor

Any editor that generates an ASCII file. This editor is used to edit the resource script.

## #define statement

A C program statement you use to assign symbolic definitions to constant values.

## identifiers

The statements you use to assign symbolic definitions to constant values.

### resource ID

A unique integer value that identifies a resource.

## identifier files

Files in which you store your program's identifiers. These are .H, .PAS, or .INC files.

## constant declaration

A Pascal declaration that you use to assign symbolic definitions to constant values.

## Project window

The window you see when you open a new or existing project. If the project is new, the window is empty. For existing projects, the window displays:

- a complete list of files in the project
- types of resources contained in each file
- identifiers associated with resources

## executable or DLL file

The ultimate destination for all resources you define with Resource Workshop. It's the file that users execute to run your program. This file is either an executable file (.EXE) or a dynamic-link library (.DLL) file.

## transparent and inverted areas

Attributes assigned to a cursor or icon. At run time, a transparent area allows the desktop color behind the icon or cursor to show through. An inverted area reverses the desktop color.

## Tools palette

The Paint editor palette you use to select the Paint editor tool you want to work with.

## Hand tool

The Paint editor tool you use to move an image around. The Hand tool is not on the Tools Palette; you can change any tool into a hand by holding down the Ctrl key.

## Icon window

The window that lists all icon images in the selected resource. Multiple images represent different color formats of the same icon, such as a 2-color and a 16-color format.

## font header

The data in a font resource that describes the entire font collection, such as the typeface number, size, and character set.

## project file

A file that contains one or more resources, or refers to files containing resources, or both. Typically, this file is a resource compiler (.RC) file.

## resource file

A file that contains one or more compiled resources. The file extension is .RES. Typically, you compile all resources for an application into a single .RES file and then bind the .RES file to the executable file.

## resource file type

The type of file that contains your resources. You can create the following types of resource files:

- Bitmap (.BMP)
- Cursor (.CUR)
- Dialog (.DLG)
- Font (.FNT)
- Icon (.ICO)
- Resource script (.RC)
- Resource (.RES)

## project

A collection of one or more resources. A project is stored in a file that contains one or more resources, or refers to files contains resources, or both. Typically, this file is a resource compiler (.RC) file.

## resource

Data that define the visible portions of your Windows program. Resources provide a consistent user interface that makes it easy for users to switch from one Windows program to another.

## Dialog editor

The editor you use to create dialog boxes and dialog box controls.

## controls

The buttons, combo boxes, text, list boxes, and scroll bars on a dialog box. Controls let users interact with dialog box data.

## Tools palette

The Dialog editor palette you use to select the Dialog editor tool you want to work with.

## Caption window

The window you use to add a caption to your dialog box or to a dialog control. It's default position is above the Tools palette and the Alignment palette.

## Alignment palette

The Dialog editor palette you use to select tools that align dialog box controls.

## binary resource statement

A resource script statement that identifies a resource and the filename where it is contained.

### multiple-line statement

Multiple resource script statements that specify the contents of a resource.

## directives

Resource script statements that affect how the resource script file is compiled.

## DIALOG

A multiple-line resource script statement that specifies a dialog window.

## MENU

A multiple-line resource script statement that defines a menu resource and specifies which menu items appear on the menu.

## ACCELERATOR

A multiple-line statement resource script statement that defines keyboard shortcuts for menu items and other program control actions.

## Menu editor

The tool you use to create and customize menu commands and menu items.

## Outline pane

The Menu editor pane that shows you the outline of the menu in pseudo code.

## String editor

The tool you use to customize string tables.

## Outline pane

The Accelerator editor pane that shows you all the accelerators defined in the selected accelerator table.

## Accelerator editor

The tool you use to customize accelerator tables.

### octal values

Use 0o (a zero followed by the letter o) or just 0 (zero) as the leading characters for octal notation.

## Resource editor

A Resource Workshop editor. Editors are provided for accelerators, dialog boxes, menus, icons, bitmaps, cursors, fonts, and string tables.

## dialog units

Dialog units represent the units used to specify a dialog window or dialog box control.

Horizontal dialog units are 1/4 the width of a character in the dialog's font. Vertical dialog units are 1/8 the height of a character in the dialog's font. They are calculated from the height and width of the dialog's font.

## Attribute pane

The Accelerator editor pane where you change an accelerator key.

## Black Frame tool

The Dialog editor tool you use to put a rectangular empty frame in your dialog box. The frame is the color of the current window frame.

## Black Rectangle tool

The Dialog editor tool you use to put a rectangle in your dialog box that's the same color as the current window frame.

## Check Box tool

The Dialog editor tool you use to put a rectangular button in your dialog box with text to the left or right.

## Combo Box tool

The Dialog editor tool you use to put a box in your dialog box that's a combination of a list box and a static control or an edit text control.

## Edit Text tool

The Dialog editor tool you use to put a rectangle in your dialog box. The user can enter text from the keyboard into this box.

## Group Box Tool

The Dialog editor tool you use to put a rectangular box around a group of controls in your dialog box. It's used to visually group controls together. You can include a caption in the upper left corner of the group box.

## Horizontal Scroll Bar tool

The Dialog editor tool you use to put a horizontal rectangle in your dialog box with a direction arrow on each end.

## Icon tool

The Dialog editor tool you use to put an icon in your dialog box.

## List Box tool

The Dialog editor tool you use to put a rectangle in your dialog box. The rectangle usually includes a list of strings. It can also contain a visual representation of the list.

## Push Button tool

The Dialog editor tool you use to put a rectangular button in your dialog box. The user "presses" the button to select an action. Push buttons always contain text, so you need to specify a caption for each one.

## Radio Button tool

The Dialog editor tool you use to put a circular button in your dialog box with text to the left or right. When the button is turned on, a solid dot fills the circle. Radio buttons are used in groups to represent related but mutually exclusive options.

## Text Static tool

The Dialog editor tool you use to put text in your dialog box.

## Vertical Scroll Bar tool

The Dialog editor tool you use to put a vertical rectangle in your dialog box with a direction arrow on each end.

## Check Box tool (BWCC)

The Dialog editor tool you use to put a Borland-style check box in your dialog box. A Borland-style check box is raised and displays a check mark, rather than an "X."

## Group Shade tool (BWCC)

The Dialog editor tool you use to put a Borland-style shaded rectangular box in your dialog box. The box groups other controls visually. It can appear recessed into the dialog box or raised above its surface.

## Horizontal Dip tool (BWCC)

The Dialog editor tool you use to put a Borland-style horizontal dividing line in your dialog box. The line gives the impression of being etched into the surface of the dialog box.

## Push Button tool (BWCC)

The Dialog editor tool you use to put a Borland-style push button in your dialog box. A Borland-style is larger than most standard Windows push buttons and can contain symbols.

## Radio Button tool (BWCC)

The Dialog editor tool you use to put a Borland-style radio button in your dialog box. The radio button is diamond-shaped and appears raised from the surface of the dialog box.

## Static Text tool (BWCC)

The Dialog editor tool you use to put a fixed text string in your dialog box. You use the string principally for labeling parts of the dialog box.

## Vertical Dip tool (BWCC)

The Dialog editor tool you use to put a Borland-style vertical dividing line in your dialog box. The line gives the impression of being etched into the surface of the dialog box.

## Getting started

If you're a first time Resource Workshop user, there are a few things you should know before you start: what projects and resources are, how to use them, and how to set preferences for the way Resource Workshop works.

Go to one of these Help screens for more information:
Projects overview
Resources overview
Setting preferences
Tips for the new user

Once you have background information, you can start using Resource Workshop. When you first begin, you need to create a new project. If you've already used Resource Workshop, you should have an existing project that you can open. Then, you can add resources to your project.

**See Also**
Adding a linked resource to a project
Adding an embedded resource to a project
Creating a new project
Opening an existing project

## Tips for the new user

Here are some tips that can help you get started:

- Use the File menu to open existing projects or create new ones.

- Use the status line at the bottom of the Resource Workshop window for explanations of functions.
- The left side of the status line gives a brief explanation of currently highlighted menu commands.
- The right side of the status line displays useful information when you use any of the graphics editors.

- If you use identifiers, store them in identifier files. Avoid storing them in your resource files. Whenever you store something, Resource Workshop prompts you for the location in which to store it.

- The Edit|Undo and Edit|Redo commands step back and forth through the previous tasks you have performed. Use File|Preferences to specify the maximum number of levels you can undo.

- You don't need access to source code to work with an application's resources. Resource Workshop can decompile an existing executable file to let you make changes to its resources.

- You can work with existing executable, dynamic link library (.DLL), and run time font (.FON) files, but you can't create these file types with Resource Workshop.

**See Also**
Tips on editing resources in existing applications

## Tips on editing resources in existing applications

Keep these considerations in mind when editing resources in existing applications:

- If you add an accelerator, make sure it returns the same ID value as its corresponding menu command. If you don't, the accelerator will either execute the wrong command or do nothing.

- You can modify existing bitmaps, cursors, and icons in an application, but you can't delete them. Do not add new bitmaps. In most cases, the application will not be able to use them.

- You can reposition items in a dialog box and convert controls to their Borland custom control counterparts.

  As you edit, be sure not to change the type of control associated with each control ID value. For example, if control ID 100 is a check box, don't change it to a radio button. The application will still treat it as a check box.

  In most cases, you can remove controls that are not directly tied to the application's functionality. For example, you can usually remove a caption (a static text item that has no effect on how the application works) but you can't remove an edit control, which does affect how the application works.

  Never add new controls. The application will not be able to use them.

- With most applications, you can safely move commands within a menu. Don't, however, move commands from one menu to another. (For example, don't move the Open command from the File menu to the Edit menu.) If you do, the application might be unable to display context-sensitive Help or check or uncheck the menu commands.

  Never change the order of the menus in the menu bar. For example, if File is the first menu, don't make it the second.

- Use caution when editing existing string tables. Some programs load the strings into buffers of fixed size, and adding text to an existing string could overflow the buffer.

- Never add new strings. The application will not be able to use them.

**See Also**
Customizing existing applications for BWCC
Resources

## Setting preferences

When you first use Resource Workshop, you should set preferences for how it works.

Choose File|Preferences to bring up the Preferences dialog box where you set configuration options.

You can:

- set the number of changes you want to trace when you undo or redo actions.
- specify the name of the text editor that Resource Workshop uses.
- set the path where Resource Workshop searches for include files.
- determine how a project is saved.
- create backup files each time you save a project.
- choose the Windows version you're working with.

## Undoing mistakes

Resource Workshop lets you undo or redo any action with the Edit|Undo or the Edit|Redo commands. Edit|Undo "undoes" your most recent action. Edit|Redo reverses the effect of the most recent Undo command.

**Edit|Undo**
The Edit|Undo command inserts any characters you deleted, deletes any characters you inserted, replaces any characters you overwrote, and moves your cursor back to its prior position. It undoes operations in the Dialog editor and the Paint editor.

If you undo a block operation on a resource, the resource will appear as it was before you executed the block operation. The Undo command will not change an option setting that affects more than one window.

If you continue to press Undo, it continues to undo changes until it reaches the number specified in the Undo Levels input box in the Preferences dialog box. Depending on the amount of memory in your computer, you can undo or redo up to 99 actions. The default number is 10.

**Edit|Redo**
The Edit|Redo command is effective only immediately after you use Edit|Undo or Edit|Redo. A series of Redo commands reverses the effects of a series of Undo commands.

## Contacting Borland

Borland offers a variety of services to answer your questions about Resource Workshop.

**Technical Support**
If you use Resource Workshop with Borland's Pascal products, call 408-461-9144 from 6 a.m. to 5 p.m. Pacific Time (PT) to reach the Pascal Technical Support team.

If you use Resource Workshop with Borland's C++ products, call 408-461-9133 from 6 a.m. to 5 p.m. PT to reach the C++ Technical Support team.

When you telephone Technical Support, call from a phone near your computer and have Resource Workshop running. Keep the following information handy to help process your call:

- Product serial number and version number
- Brand name and model of any hardware in your system
- Operating system and version number
- Contents of your AUTOEXEC.BAT and CONFIG.SYS files
- Contents of your WIN.INI and SYSTEM.INI files (located in your \WINDOWS directory)
- A daytime phone number where you can be contacted
- The specific steps to reproduce the problem

**Advisor Line**
The Borland Advisor Line is a service for users who need immediate access to advice. The Advisor Line operates weekdays from 6 a.m. to 5 p.m. PT. The first minute is free; each subsequent minute is $2.00. The number is 900-555-1001.

**Other Borland resources**
TechFax is a 24-hour automated system that sends free technical information to your fax machine. You must have a touch-tone phone to use this service. You can request up to three documents per call. The TechFax number is 800-822-4269.

The Borland File Download BBS has sample files, applications, and technical information you can download with your modem (2400 baud). No special setup is required. The Borland BBS number is 408-439-9096.

**Customer Service**
Borland Customer Service is available weekdays from 7 a.m. to 5 p.m. PT to answer any non-technical questions you have about Borland products, including price information, upgrades, and order status. The Customer Service number is 408-438-5300.

**Online information services**
Subscribers to the CompuServe, GEnie, or BIX information services can receive technical support by modem. Use the commands in the following table to contact Borland while accessing an information service.

| Service | Command |
|---|---|
| CompuServe | GO BORLAND |
| BIX | JOIN BORLAND |
| GEnie | BORLAND |

Address electronic messages to Sysop or All. Don't include your serial number; messages are in public view. Include as much information as possible.

## Projects

A project is a collection of one or more resources. A project is stored in a file that contains one or more resources, or refers to files containing resources, or both. Typically, this file is a resource compiler (.RC) script file.

When you first use Resource Workshop, you need to create a new project or open an existing one. Once the project is open, it's displayed in the Project window.

You need to save the project when you exit; Resource Workshop does not save the project automatically.

**See Also**
Creating a new project
Opening an existing project
Managing your project - the Project window
Resources
Saving a project

## Creating a new project

To create a new project,

1. Choose File|New Project.

2. In the New Project dialog box, turn on the Project File Type radio button that corresponds to the type of file on which you want to base your project.

3. Click OK.

Resource Workshop displays your new project in the Project window where you add resources to it. You name your project when you save it.

**See Also**
Adding a linked resource to a project
Adding an embedded resource to a project
Managing your project - the Project window

## Opening an existing project

An existing project is usually one that you created with Resource Workshop. It can also be an .RC file you created with other resource development software.

You can also work with the resources in any application developed for Windows 3.0 or higher, even if you don't have access to the source code. If you have access only to an executable file, Resource Workshop can decompile the file to let you make changes to the resources.

To open an existing project,

1. Choose File|Open Project.

2. In the Open Project dialog box, specify the file containing the project you want to open.

3. What Resource Workshop does next depends on whether the project is a binary file or a file containing resource data.

- If the project is a binary file (an executable file, a .RES file, or a dynamic link library file), Resource Workshop decompiles the resources and shows you its progress on the left side of the status line.
- If the project consists of a main .RC file and other files containing resource data, Resource Workshop reads the project file and then compiles each resource, showing you its progress in the Compile Status dialog box.
- If Resource Workshop can't compile the project file, you see the Compiler Error dialog box. It shows you the error and where it occurred.

4. Once the project is compiled or decompiled, Resource Workshop displays the Project window with the resources listed in it.

When the project is open, you can add resources to it. Save the project when you finish.

**See Also**

Adding a linked resource to a project
Adding an embedded resource to a project
Executable and dynamic link library files
Managing your project - the Project window
Resource compiler files
Resource files
Saving a project

## Compiler Error dialog box

The Compiler Error dialog box shows the compile error and where it occurred.

**Error information**
The compiler's internal error number and text of the error are displayed at the top of the dialog box.

**File name and line number information**
The name of the project file and the line number of the error are displayed in the middle of the dialog box.

**Resource Script display box**
The resource script statement that contains the error is displayed in the Resource Script display box. Five lines of source script are displayed.

**Text Editor input box**
The Text Editor input box is where you enter the name of the text editor you want to use.

**Run Editor button**
The Run Editor button brings up the text editor.

**See Also**
Opening an existing project

## Text Editor input box

The Text Editor input box is where you specify the name of the text editor that Resource Workshop uses. The default text editor is the Windows Notepad editor, NOTEPAD.EXE.

You use the text editor to edit a resource's source script.

**See Also**
Using the text editor

## Run Editor button

The Run Editor button brings up the text editor specified in the Text Editor input box.

You can fix the error in the resource's source script. Save the file, exit the editor, and open the project again.

## Managing your project - the Project window

The Project window lists the resources in your project. It acts as an effective file management tool, making it easy to look at an overall view of a project. Even if your project contains a large number of resources, you can quickly scan the project by scrolling through the Project window.

Once you open a new or existing project, Resource Workshop displays the Project window. For a new project, the window is empty. You have to put resources into it by creating them or adding them as files.

For an existing project, you can see the:

- complete list of files in the project
- types of resources contained in each file
- identifiers associated with resources

**Embedded and linked resources**

The resources in your project file can be embedded in the file or linked to it.

- An embedded resource is stored in resource script form in the project file. It exists only as part of the project in which it's stored, and it can't be used in other projects.
- A linked resource is a separate file that is referenced in the project file. Linked resources can be used in other projects.

Use the View menu to determine how the Project window displays information. You can group resources by file or type; display the identifiers in the project; list the names of resources; display items in resources; and display all possible types of resources.

**Status line**

The status line at the bottom of the Resource Workshop screen is divided into a left and right side. It displays information about commands and tools.

The left side of the status line displays information about the currently highlighted menu command. For example, if you choose the Resource|New command, the status line displays "Create new resource."

The right side of the status line is active only when you're in the Dialog editor or the Paint editor. It displays details about the editor and tool you're working with.

**See Also**

Adding a linked resource to a project
Adding an embedded resource to a project

## Saving a project

It's a good idea to save your work often. Resource Workshop provides you with a variety of save commands so you can choose exactly what you want to save and how to save it.

Choose the File menu and select one of these options:

- File|Save Project saves everything in your current project. This is the option you usually choose. When you save a new project that hasn't been named yet, Resource Workshop displays the Save File As dialog box, where you can specify a name and directory.

- Resource Workshop always saves the project file and any files it references. Resource Workshop can also save to a .RES file or bind the resources to an executable or dynamic link library file, if you turned one or both of the Multi-save check boxes on in the Preferences dialog box.

- If your project is based on an .RC file, Resource Workshop compiles the project as part of the save process. This compiled version is stored in a file with an .RWS extension. The next time you open this project, Resource Workshop can save time by using the .RWS file instead of the .RC file.

- File|Save File As lets you rename the current project when you save it. Resource Workshop displays the Save File As dialog box, where you can enter the new file name.

**See Also**
Executable and dynamic link library files
Resource compiler files

## Preprocessor Warning dialog box

The resource you're about to edit contains comments, preprocessor directives, and/or multi-field macros. If you edit the resource and save changes, the comments, preprocessor directives, and/or multi-field macros will not be saved.

Choose OK to edit the resource or Cancel to cancel the edit. Choose View Only to load the resource into the appropriate editor in display-only mode.

## Resources

Resources are data that define the visible portions of your Windows program. Resources provide a consistent user interface that makes it easy for users to switch from one Windows program to another.

In general, a Windows application's resources are separate from the program code, letting you make significant changes to the interface without even opening the file that contains your program code. This also lets different applications share the same set of resources, so that you don't have to reinvent all your favorite resources. Instead, you can use them over and over.

Here are the types of resources you can create with Resource Workshop:
Accelerators
Bitmaps
Cursors
Dialog boxes
Fonts
Icons
Menus
String tables
User-defined resources
VERSIONINFO

**See Also**
Identifier and identifier files
Resource file types
Tips on editing resources
Working with resources

## Resource file types

A file you create and edit with Resource Workshop can be in either binary or text format. Resource Workshop generates standard Windows file formats, which means you can use Resource Workshop files with programs that generate binary code from resource script files.

Some Resource Workshop .RC files will be incompatible with the Microsoft Resource Compiler. See the Help topic called Differences between Borland and Microsoft compilers for more information.

Here are the types of resource files you can create with Resource Workshop:
Bitmap (.BMP)
Cursor (.CUR)
Device driver files
Dialog (.DLG)
Executable and dynamic link library files
Font (.FNT)
Icon (.ICO)
Resource compiler (.RC)
Resource (.RES)

## Resource compiler files

A resource compiler (.RC) file is a resource script file that contains one or more resources. The file can contain the resources in script form and references to other files that actually contain the resources. The resources in the referenced files can be in either binary format or script form, depending on the resource type.

In general, you should base all your Resource Workshop projects on at least one .RC file.

## Resource files

A resource (.RES) file contains one or more compiled resources.

Typically, when creating a Windows program, you compile all resources for an application into a single .RES file, and then you bind the .RES file to the executable file as part of the linking process.

However, if you don't want to produce a .RES file, you don't have to. Resource Workshop will compile resource files and bind them directly to an executable file.

If you want to change the resources in a compiled binary file, Resource Workshop will decompile the file and let you make the changes, then save the resources back to the original file.

**See Also**
Working with a binary file

## Executable and dynamic link library files

An executable (.EXE) or dynamic link library (.DLL) file is the ultimate destination for all resources you define with Resource Workshop. It's the file that users execute to run your program. In general, you compile the resource compiler file (.RC file) into the resource file (.RES file), then use your compiler to bind the .RES file to the executable or DLL file.

You can also use Resource Workshop to bind the resources directly to the executable or DLL file and bypass the Microsoft Resource Compiler altogether.

If you want to change the resources in a compiled binary file (an executable file, a DLL file, or a .RES file), Resource Workshop will decompile the file and let you make the changes, then save the resources back to the original file.

**See Also**
Resource compiler files
Resource files
Working with a binary file

## Device driver files

A Windows device driver (.DRV) file is a special form of a dynamic link library (.DLL) file.You can edit the resources in one of these files just as you can in any .DLL file.

## Dialog files

A dialog (.DLG) file is a resource script file that contains a description of one or more dialog boxes. There is no requirement that these resources be dialogs; they can be any resource found in an .RC file.

**See Also**
Resource compiler files

## Icon files

An icon (.ICO) file contains an icon in binary format. The file can include more than one icon image.

## Bitmap files

A bitmap (.BMP) file contains a bitmap in binary format.

## Cursor files

A cursor (.CUR) file contains a cursor in binary format.

## Font files

Font files take two forms, binary and run time.

- A binary font (.FNT) file contains the definition of a customized font in binary format. You use the Resource Workshop Paint editor to design a font and store it in an .FNT file.
- A run time font (.FON) file is a resource-only dynamic link library that contains a font directory and one or more fonts. You must create .FON files outside of Resource Workshop using program development tools. However, once you create a .FON file, you can use Resource Workshop to modify the file.

## Identifiers and identifier files

Windows requires that every resource be associated with a unique name or a unique integer (called a resource ID). By default, Resource Workshop assigns a name to each new resource.

The default name isn't very descriptive, and referring to a resource by name alone decreases the application's efficiency at run time. To overcome these shortcomings, you can rename the resource and assign it an identifier (a C #define or a Pascal constant).

**Identifiers**

An identifier consists of two parts: a text literal (the identifier name) and a value (typically an integer). Identifiers must be unique within a resource type. Only the first 31 characters are significant; Resource Workshop ignores any characters past the 31st character.

Assigning an integer value to an identifier speeds up calls to the resource at run time, but you won't be able to use the short integer value directly as a parameter. You must either typecast the integer into a long pointer to **char** or use a macro to do the typecasting for you.

- If you write your program in C or C++, use the MAKEINTRESOURCE macro.
- If you write your program in Pascal, use the MakeIntResource type (a pointer to char).

If you're working with a .RES file, an executable file, or a DLL file, Resource Workshop decompiles all resource IDs in the files into integer values. You can't add identifiers to this type of file, but you can save the file as an .RC file and assign identifiers to its resources.

**Identifier files**

When you create a new project, the first thing you should do is specify a file in which to store your identifiers.

- If you write your program in C, store identifiers in one or more header (.H) files that use #defines to assign values to identifier names.
- If you write your program in Pascal, store identifiers in one or more units (.PAS) or include (.INC) files that use constants to assign values to identifier names. If you use a Pascal unit or include file, you can assign only numeric values to the constants in the file.

**Note:** If you store constants in an include file, you can store only constant declarations, comments, and compiler directives.

**See Also**

Working with identifiers

## Working with identifiers

Windows requires that every resource have a resource ID associated with it. You can use Resource Workshop to assign actual integers as resource IDs, but it's more likely that you'll want to use an identifier to represent the resource ID.

When you work with identifiers, you

- create an identifier file
- add identifiers
- edit identifiers
- delete identifiers
- list identifiers

**See Also**
Identifiers and identifier files

## Creating an identifier file

After you open a new project and give it a name, add the identifier by taking these steps:

1. Choose File|Add to Project. You see the Add File to Project dialog box.
2. Click the down-arrow at the end of the File Type list box.If you write your program in C, choose
   ```
   H c header
   ```
   If you write your program in Pascal, choose one of the following
   ```
   PAS Pascal constant unit
   INC Pascal constant include
   ```
3. In the File Name input box, enter a name for the identifier file.
4. Click OK. Resource Workshop creates the identifier file.

**See Also**
Identifiers and identifier files

## Adding identifiers

You can add an identifier to your identifier file before you create the resource it will be associated with. To add an identifier,

1. Choose Resource|Identifiers to display the Identifiers dialog box.
2. Click the New button. Resource Workshop displays the New Identifier dialog box.
3. In the File drop-down list, enter the name of the file in which the identifier is to be stored.
4. Type the resource name in the Name input box.
5. Type the ID value in the Value input box.
6. Click the OK button.

**Note:** The new resource name now appears in the Identifiers list box, and its Value is given as (unused).

**See Also**
Identifiers and identifier files

## Editing identifiers

You can change an identifier value by doing the following:

1. Choose Resource|Identifiers to display the Identifiers dialog box.
2. Select the identifier whose value you want to change.
3. Click the Change button. Resource Workshop displays the Change Identifier Value dialog box.
4. Type a new value in the New Value input box and click OK.

The new identifier value will be written to your .H, .PAS, or .INC file the next time you choose File|Save Project.

**See Also**
Identifiers and identifier files

## Deleting identifiers

If an identifier is not used in your project, you should delete it from the .H, .PAS, or .INC file. There are three reasons you might have an unused identifier:

- You assign an identifier to a resource and then delete the resource.
- You add an identifier to the project and then never use it.
- You rename a resource that already has an integer identifier value.

To delete an identifier,

1. Choose Resource|Identifiers to display the Identifiers dialog box.

2. Select the identifier you want to delete. You can delete an identifier that is still in use.
   - If the selected identifier is not associated with a resource (either because the resource was deleted or the identifier was never used), the Usage box says (unused).
   - If, however, the identifier is still associated with a resource, the Usage box automatically highlights the type and name of the associated resource.

3. Click the Delete button.
   - If the identifier is unused, it is deleted immediately. No warning dialog box is displayed.
   - If the identifier is still in use, Resource Workshop displays a warning dialog box that says "define [or Constant] is used. Delete anyway?" To delete the identifier, click the Yes button. If you don't want to delete the identifier, click the No button.

The next time you choose File|Save Project, Resource Workshop updates the identifier file, removing the deleted identifier.

**See Also**
Identifiers and identifier files

## Listing identifiers

To list the identifiers in your project, do the following:

1. Choose Resource|Identifiers to display the Identifiers dialog box.

2. Use the View radio buttons to choose the identifiers you want to see. If you choose Single File, select the file whose identifiers you want to see from the file name list box next to the View radio buttons.

3. Scroll the Identifiers list box to the identifiers you want to see. When you highlight an identifier in the list box, its name and integer value appear in the Name and Value boxes above the list box.

**See Also**
Identifiers and identifier files

## Working with binary files

Resource Workshop allows you to open executable (.EXE), dynamic link library (.DLL), and resource (.RES) files as projects so you can customize their user interfaces.

When you load one of these files, Resource Workshop decompiles the resources in the file and shows them to you as though they were part of a regular .RC file. When you finish, Resource Workshop compiles the resources again into binary code and stores them in the original file.

Because the resources you work with in this type of project are never stored as resource scripts, you can't assign any identifiers to the resource IDs. However, you can save the project as an .RC file. Then the resources can be saved as resource scripts, and you can assign identifiers to them.

Once you save the project as an .RC file, Resource Workshop won't automatically save the resources back to the original file. You have to enter the original file name in the Preferences dialog box in order to save the resources there as well.

The whole process works as follows:

1. Choose File|Open Project and select the executable, .RES, or DLL file from the Open Project dialog box.

2. Choose File|Save File As. In the Save File As dialog box, select RC Resource Script from the File Type list box. Enter the name of the new .RC file.

3. Press OK to save the file. Resource Workshop automatically places you in the .RC file.

4. Resource Workshop asks you if you want the reference in the project file to refer to this external file from now on.

- Click Yes to replace the resource script or current file reference in the Project file with a reference to the new file. All future changes to the resource will be saved to the external resource file and not in the project file or in any previous resource file.
- Click No to create the resource file without making any changes to the Project file.

**See Also**
Executable and dynamic link library files
Identifiers and identifier files
Resource files

## Working with resources

When you display resources in the Project window, you see them listed by name. To load a resource, you can do either of the following:

- Double-click the resource name in the Project window. Resource Workshop automatically starts the appropriate editor, if one is available. If an editor is not available (for example, for a user-defined resource), Resource Workshop starts the internal text editor.
- Select the resource name in the Project window and choose either Resource|Edit or Resource Edit as Text. Resource|Edit loads an editor, if one is available. Resource|Edit as Text always starts the internal text editor.

Once a resource is loaded, you can

- add a linked resource to a project
- add an embedded resource to a project
- delete a resource
- rename a resource
- save a resource
- specify resource memory options

**See Also**

Using a resource editor
Using a text editor

## Using a resource editor

You can edit any resource in a project by double-clicking the resource. You can also edit a resource by clicking the resource name in the Project window and choosing Resource|Edit.

Here are the available editors:

- For an accelerator resource, Resource Workshop loads the Accelerator editor.
- For a dialog box resource, Resource Workshop loads the Dialog editor.
- For a menu resource, Resource Workshop loads the Menu editor.
- For an icon, bitmap, cursor, or font resource, Resource Workshop loads the Paint editor.
- For a string table resource, Resource Workshop loads the String editor.

## Using a text editor

The text version of a resource is also called source code or script.

You can edit the text version of any resource. To edit the text version of a resource,

1. Click the resource in the Project window.

2. Choose Resource|Edit As Text. Resource Workshop opens its internal text editor and displays the script for the resource.

The default text editor, NOTEPAD.EXE:

- uses the Del, Home, End, PgUp, PgDn, and BkSp keys as you would expect.
- is always in insert mode.
- can't use tabs, but you shouldn't spend any time formatting anyway, because Resource Workshop is likely to rearrange the text for you when it compiles.

When you finish, choose Control|Close. Click Yes in response to the prompt "Resource has changed. Compile?" (If you click No, you lose your changes). You can also use the Compile menu to compile your resource before you exit the text editor.

Resource Workshop compiles your changes and saves them. If there's a syntax error, you are put back in the text editor so you can correct the error.

## Adding an embedded resource to a project

To embed a resource in your project (to create a new resource in script form), open the project you want to work with and then

1. Choose Resource|New to display the New Resource dialog box.

2. Double-click the type of resource you want to create.

The next steps depend on the type of resource you're creating.

- If you're creating an accelerator, menu, dialog box, string table, or an RCDATA or VERSIONINFO resource, Resource Workshop puts an entry for the resource in the Project window and opens the appropriate resource editor.

- If you're creating a bitmapped resource (an icon, cursor, bitmap, or font), Resource Workshop opens the Storage Format dialog box. To embed the bitmapped resource, click the Source button.

- If the type of resource you want to create isn't listed, click New Type to create a user-defined resource type for your resource.

**See Also**
Adding a linked resource to a project
Creating a user-defined resource type

## Adding a linked resource to a project

You can link a resource stored in an external file to the current project by using the File|Add to Project command. You can use this command to add an existing resource file or to create a new file for a new resource.

**Existing resource files**
Here's how you link a resource in a separate file to your project. With the project open,

1. Choose File|Add to Project. Resource Workshop displays the Add File to Project dialog box.

2. Either type the name of the file containing the resource in the File Name input box, or double-click the file name if it's listed in the Files list box.

   If the file isn't in the current directory or if it is of a different type from the current type, you can select it in either of these ways:

- Type the file's full path and name into the File Name input box.
- Change directories by using the Directories list box. Then enter your search criteria in the File Name input box and press Enter, or select a type from the File Type list. When the file name you want appears in the Files list box, double-click to select the file.

3. In the RCINCLUDE drop-down box, you see the current project file listed, which is most likely where you will put the reference to the new file. If your project contains more than one .RC file and you want to put the reference elsewhere, scroll down the list to find the name of the file in which you want to place the reference.

4. Press Enter or click OK to add the file to the project. Resource Workshop puts an entry that points to this file in the Project window.

   If you choose View|By File, you see the file name and the resource name. Any changes you make in the project to this resource are reflected in the original resource file.

**New resource files**
Adding a new file to a project works the same as adding an existing file, except that you enter the name of a nonexistent file. When you click OK to add the file to the project, Resource Workshop tells you that the file doesn't exist and asks if you want to create it. If you do, click OK. Resource Workshop creates a file of the appropriate type (based on the file extension or, if the extension isn't standard, the file type) and inserts a reference to the file in the Project window.

**See Also**
Adding an embedded resource to a project

## Renaming a resource

It's easy to rename a resource in your project:

1. Click the resource in the Project window to select it.

2. Choose the Resource|Rename command. Resource Workshop displays the Rename Resource dialog box, where you rename your resource.

3. Enter the new resource name in the New Name input box.

4. Click OK. Resource Workshop asks you if you want to create a new identifier by that name.

- If you click Yes, Resource Workshop displays the New Identifier dialog box, where you enter an new integer value for the renamed resource. The next time you save your project, the identifier name and value will be saved to your .H, .PAS, or .INC file.
- If you click No, Resource Workshop renames the resource without creating an identifier.

## Deleting a resource

To delete a resource from a project, select it in the Project window, then do one of the following:

- Press the Del key or choose Edit|Delete to completely delete it.
- Choose Edit|Cut to cut the resource into the Windows clipboard so you can paste it elsewhere.

## Specifying resource memory options

Resource Workshop makes it easy to specify how each resource in your project should be managed in memory. It's better to leave the memory settings at their defaults unless you're an experienced Windows programmer - you might not be able to foresee the implications of changing the way a resource is handled in memory.

To specify memory options,

1. Select the name of the resource in the Project window. You can click the name with the mouse or use the arrow keys to select it.

2. Choose Resource|Memory Options to open the Resource Memory Options dialog box.

3. Check one or more of the Memory Options check boxes.

## Saving resources and resource files

It's a good idea to save your work often. Resource Workshop provides you with a variety of save commands so you can choose exactly what you want to save and how to save it.

- Choose File|Save File As to rename the current resource file when you save it. Resource Workshop displays the Save File As dialog box, where you can enter the new file name.

- Choose Resource|Save Resource As to put a resource in a separate file for use with other projects. Resource Workshop displays the Save Resource As dialog box, where you can enter the new file name and choose the appropriate file type.

- Choose File|Save Project to save everything in the current project. If you're saving a new project for the first time, Resource Workshop displays the Save File As dialog box so you can specify a name and directory.

  Resource Workshop saves the project file and any files it references. If you have selected .RES or .EXE from the Preferences dialog box, Resource Workshop also compiles and saves to a .RES file or binds the resources to an executable file or DLL. If your project is based on an .RC file, Resource Workshop also compiles the project as part of the save process.

# Menus

Menus are lists of commands the user chooses from. Most Windows applications have a menu bar across the top of the screen that contains the names of the application's menus. Each menu contains a set of commands.

**Terminology**

These terms describe the elements of a menu resource: pop-up command, pop-up menu, menu item, and menu separator.

- Pop-up commands cause menus to be displayed. Pop-up commands can appear in the menu bar and can also appear inside pop-up menus, where they cause another menu (called a "cascading menu") to be displayed.
- Pop-up menus are the rectangular boxes containing lists of commands from which a user can choose. They come in two forms:
- Drop-down menus are displayed from the menu bar or from within a menu. They are tied to a pop-up command and are always displayed from that command's name.
- Floating menus can appear anywhere in the application window.
- Menu items are the commands that appear in the menus, such as Open, Save, or Print.
- Menu separators are the lines that divide the menu items into logical groups. Separators don't do anything other than make the menu easier to read and use.

**Working with menus**

The Menu editor makes it easy to create and edit menus. Working with menus involves four basic steps:

1. Start the Menu editor.
2. Create or edit the menu
3. Test the menu.
4. Save the menu.

**See Also**

Adding accelerator text to menus
Creating a new menu
Editing an existing menu
Saving the menu
Testing the menu
Using the Menu editor

## Using the Menu editor

You use the Menu editor to create and edit menus. It provides three different views of the menu you're editing:

- the Outline pane shows the menu's pseudo code
- the Attribute pane is where you customize the currently highlighted line in the menu
- the Test Menu pane is where you test your menu

Use the Tab key to move from pane to pane. Change pane positions with the View command.

## Outline pane

The Outline pane shows you the new menu's pop-up commands, menu items, and separators in pseudo code. The top line in the pane is the name of the menu, and the other lines are statements defining pop-up menus and menu items.

The pseudo code doesn't show all parameters for each statement. It just shows the structure of your menu. To see the complete code with all parameters for each statement, edit the menu's resource script.

The actual editing of the menu takes place in the Attribute pane. Use F6 to move between the Attribute pane and the Outline pane.

**See Also**
Editing a menu resource as text

## Attribute pane

The Attribute pane is where you edit pop-up commands and menu items, assign ID values, and set attributes for your menus and menu items. The statement you're editing is highlighted in the Outline pane.

Use the mouse, Tab, or Shift+Tab keys to position on the field you want.

Here are the components of the Attribute pane:

**Item Text input box**
The Item Text input box is where you specify the name of a menu item or pop-up menu.

**Item ID input box**
The Item ID input box is where you choose a unique ID for the menu item.

**Item Type radio buttons**
The Item Type radio buttons choose the type of menu you're adding.

**Break Before radio buttons**
The Break Before radio buttons control the format of menu commands.

**Initial State radio buttons**
The Initial State radio buttons choose the menu item's initial state.

**Checked check box**
The Checked check box turns the initial state of the command to On.

## Test Menu pane

The Test Menu pane displays your menu and lets you test it. The Menu editor automatically updates the test menu as you change it in the Attribute pane.

**See Also**
Testing a menu

## Creating a new menu

To create a new menu,

1. Open the project you want to add the menu to.
2. Choose Resource|New to create a new resource for that project. The New Resource dialog box appears.
3. Double-click MENU in the Resource Type list box.
4. Click OK to place the menu in the current project file or choose another project file.

Resource Workshop displays the Menu editor with a default menu you can begin customizing.

**See Also**
Customizing a menu
Opening an existing project
Using the Menu editor

## Editing an existing menu

To edit an existing menu, open the project in which the menu is stored and do one of the following:

- Double-click the menu resource name in the Project window.
- Highlight the menu resource name and choose Resource|Edit.

Resource Workshop displays the Menu editor with the menu you've chosen loaded in.

**See Also**

Customizing a menu
Editing a menu resource as text
Using the Menu editor

## Customizing a menu

Once a menu is loaded into the Menu editor, you're ready to add new menu commands, pop-up menus, or separators, or to move, copy, or delete any part of the menu.

You can use the three standard pop-up menus (Edit, File, and Help) in your menus. You can also customize your menus by adding accelerators to them.

**See Also**

Adding a new statement
Adding a pop-up command
Adding accelerator text to menus
Adding menu items and separators
Creating a floating menu
Deleting a menu statement
Moving and copying menu statements

## Adding a new statement

You customize your menu by adding statements for menu items, pop-up menus, and menu separators.

Position the cursor in the Outline pane on the line preceding where the statement is to go. To insert a statement at the beginning of the outline, highlight the top line (MENU_1 or the name of the menu resource).

When you've decided where the new statement goes, and you've highlighted the appropriate line, add a new statement with the Menu menu. You can add a new pop-up menu, a menu item, or a single separator. You can also add a predefined File, Edit, or Help menu.

**See Also**
Customizing a menu

## Moving and copying menu statements

You can use Cut, Copy, and Paste on the Edit menu to move and copy the statements in the Outline pane.

To move a statement, highlight the statement and choose Edit|Cut. The highlighted statement is removed from the outline and placed in the paste buffer. Note that you can't cut the last (or only) statement from the outline. There must always be at least one menu item, pop-up command, or menu separator in the outline.

To copy a statement, highlight it and choose Edit|Copy. The highlighted statement remains in the outline, but a copy is placed in the paste buffer.

To insert the cut or copied statement into your menu, highlight the statement immediately before the point at which you want the statement to appear, and choose Edit|Paste.

**Undoing errors**
The Menu editor lets you undo and redo changes. Choose Edit|Undo or Edit|Redo or use the accelerators Alt+Backspace (Undo) and Shift+Alt+Backspace (Redo).

The text of the Undo command varies to indicate which change it is going to undo. For example, if you mistakenly change a menu item into a separator, the command becomes Undo Change Item.

**See Also**
Customizing a menu

## Adding menu items and separators

To add a new menu item or separator to a menu, decide where you want the new statement to appear and highlight the previous line in the Outline pane.

The Menu editor inserts the new menu item or separator below the highlighted line.

- To insert a new menu item, press Ins, or choose Menu|New Menu Item.
- To insert a new menu separator, press Ctrl+S or choose Menu|New Separator.

You can now edit the text string, menu ID, and accelerators, if any.

**Editing a menu item**
A newly added menu item has the generic designation "Item." To make it useful, you need to edit it with the Attribute pane. You can change the text that displays in the menu, change the menu's numeric ID, and put a check mark next to the menu if it's a toggle.

**See Also**
Adding accelerator text to menus
Customizing a menu

## Adding a pop-up command

A pop-up command produces a menu of command choices or another, cascading menu. In most Windows applications, all items in the menu bar across the top of the window are pop-up commands.

You can insert a pop-up command in the top level of your menu to make it appear on the menu bar, and you can embed a pop-up command inside a menu.

To add a pop-up command,

1. In the Outline pane, highlight the line immediately before where you want the new pop-up command to appear.

   If you're adding a pop-up command that will appear first in the menu bar (the position typically occupied by the File menu), highlight the first line of the outline (MENU_1 or the name of the menu resource).

2. Press Ctrl+P or choose Menu|New Popup.

3. Use the Attribute pane to further define the pop-up command.

**See Also**
Adding accelerator text to menus
Customizing a menu

## Deleting a menu statement

Highlight the statement you want to delete, then press Del or choose Edit|Delete to delete it, or choose Edit|Cut to delete the statement and copy it to the Clipboard.

Note the following about deleting menu statements:

- If you delete a POPUP statement, you delete the pop-up command it defines and all the items contained in the pop-up menu.
- You can't delete _End Popup_ statements.
- You can't delete the last (or only) statement from the outline. There must always be at least one menu item, pop-up command, or menu separator in the outline.

**See Also**
Customizing a menu

## Creating a floating menu

A floating menu is one that can be displayed anywhere in the application's window space. It is not tied to the menu bar.

Each floating menu must be saved as a separate menu resource within the project file.

To create a floating menu,

1. Choose Resource|New to create a new resource for that project. The New Resource dialog box appears.

2. In the Resource Type list box double-click MENU.

3. Choose View|View as Pop-up to see the floating menu as it would appear on the screen at run time.

   When you view the menu in the Test Menu pane, it will still appear tied to the menu bar, but as long as your code uses the **TrackPopupMenu** function correctly, the menu will float at run time.

4. Select the first line of the outline (MENU_1 or the name of the menu resource).

5. Press the INS key to add at least one menu item at the top of the outline.

6. Select the string POPUP "Pop-up" in the outline and press the DEL key to delete the POPUP statement, its menu item, and the _End Popup_ statement.

7. Add any additional menu items you want.

8. Edit the menu items in the Attribute pane.

9. Save your project.

**See Also**
Customizing a menu

## Adding accelerator text to menus

To let users know what accelerators they can use, you can add accelerator text to your menus.

For example, if your application includes a Shift+Del accelerator that duplicates the Cut command on the Edit menu, you can add the text "Shift+Del" next to that command.

- Use the tab character (\t) to separate the menu title from the accelerator text. For example, Cut\tShift+Del means that the accelerator Shift+Delete is assigned to the menu command Cut.
- Use the right-align character (\a) to right-align accelerator text.

**Note:** Pop-up commands don't have accelerator keys.

When you add accelerator text to menus, you use these input boxes:

- The Item Text and the Item ID input boxes in the Menu editor Attribute pane.
- The Command input box in the Accelerator editor Attribute pane.

**See Also**

Customizing a menu

## Testing a menu

The Menu editor gives you immediate testing capability. The test menu is updated as you make changes to your menu, so you can test changes whenever you want.

The Menu editor also has a built-in debugging tool that you can use to test for duplicate menu item IDs. Choose Menu|Check Duplicates to search for duplicate values. If the Menu editor finds duplicates, it displays a dialog box with the message "Duplicate command value found."

When you close this message box, the Menu editor highlights the statement with the duplicate value. Determine whether the menu ID is an identifier or a numeric value by looking at the Item ID input box.

- If the menu ID is a number, enter a new number that doesn't conflict with the other menu IDs.
- If the menu ID is an identifier, choose Resource|Identifiers to display the Identifiers dialog box where you can change the identifier.

**See Also**
Test Menu pane

## Saving a menu

It's a good idea to save your changes as you go along, rather than waiting for Resource Workshop to prompt you when you close the project.

There are two primary options you can use to save changes you've made to a menu. You can save the entire project or save the menu in a resource script file.

**See Also**
Save the entire project
Save the menu in a resource script file

## Saving a menu in a resource script file

You can save the menu resource as a separate file, either an .RC or .DLG file. This allows you to create menu resources that you can use in more than one project.

To save the resource as a file, choose Resource|Save Resource As. This command brings up the Save Resource As dialog box, where you enter the file name for the menu resource.

To add the file to the project as an external resource, choose File|Add to Project.

## Editing a menu resource as text

The Menu editor lists the pseudo code for your menu in the Outline pane, but this window doesn't list complete source code statements with all parameters.

To work with the resource script for a menu, select the menu name from the Project window by clicking it. Then choose Resource|Edit As Text. Resource Workshop brings up the source script for the resource in its internal text editor.

Don't spend any time inserting comments in your resource script or formatting the text, because the Resource Workshop incremental compiler does its own formatting and discards all comments.

**See Also**
Resource script language
Using a text editor

## Item Text input box

The Item Text input box is where you enter the menu name or command that appears in the menu bar or menu.

If you want the user to choose the command from the menu by typing one of the letters in the command, put an ampersand (&) immediately before that letter. Windows will display the command with that letter underlined.

If you plan to link an accelerator to this command, you can add accelerator text to the menu.

- Use the tab character (\t) to separate the menu title from the accelerator text. For example, Cut\tShift+Del means that the accelerator Shift+Delete is assigned to the menu command Cut.
- Use the right-align character (\a) to right-align accelerator text.

Note that pop-up commands don't have accelerator keys.

**See Also**

Adding accelerator text to menus

## Item ID input box

Resource Workshop automatically assigns a unique resource ID to each new menu item. You can accept this value, or you can enter a unique number or name into the Item ID input box. You don't need to enter this value for pop-up commands.

If you add accelerator text to the menu command, you need to enter the value in this input box in the Command input box in the Accelerator editor Attribute pane.

**See Also**
Adding accelerator text to menus

## Item Type radio buttons

The Item Type radio button selects the menu type you're creating. You can choose between an actual menu item and a menu separator.

**Pop-up**
Turn the Pop-up radio button on if you're creating a pop-up menu.

**Menu Item**
Turn the Menu Item radio button on if you're creating a menu item.

**Separator**
Turn the Separator radio button on if you're creating a menu separator.

## Break Before radio buttons

The Break Before radio buttons control the format of menu commands in the menu bar and in pop-up menus.

**No Break**
Turn the No Break radio button on if you don't want a break before this menu item.

**Menu Bar Break**
Turn the Menu Bar Break radio button on if you want to start a new line in the menu bar, or a new column in a pop-up menu.

**Menu Break**
Turn the Menu Break radio button on to start a new line in the menu bar. In a pop-up menu, this option starts a new column and draws a vertical line to separate the columns.

**Help Break**
Turn the Help Break radio button on to move the menu item to the far right of the menu bar. Use this option only with top-level menu items in the menu bar.

## Initial State radio buttons

The Initial State radio buttons select the menu item's initial state.

**Enabled**
Turn the Enabled radio button on to enable the menu item or pop-up, so that it will work when the user chooses it.

**Disabled**
Turn the Disabled radio button on to disable the menu item. The user won't be able to distinguish between Enabled and Disabled items; they'll look the same on the menu. Use the **EnableMenuItem** function to change the state of the menu item.

**Grayed**
Turn the Grayed radio button on to disable the menu item or pop-up and gray the displayed text. The shading lets the user know the menu item is not currently available. Use the **EnableMenuItem** function to change the state of the menu item.

## Checked check box

If the menu item will toggle on and off, turn the Checked check box on. This places a checkmark next to the menu item. A checkmark means that the command is on.

Since menu bar commands and pop-up menus don't toggle, this option only applies to menu items on pop-up menus.

Use the **CheckMenuItem** function to change the state of the menu item.

## Dialog boxes

Dialog boxes give the user a way to interact with your application. A dialog box is usually a pop-up window that lets the user specify information (files to open, colors to display, text to search for, and so on).

When a dialog box is displayed in an application, it's shown as a window. The dialog box usually contains a number of controls, such as buttons, text boxes, and scroll bars. Controls usually let the user specify information, but can also be used to display static text and graphics in a dialog box.

From the programmer's perspective, the dialog box is the parent window and each control is a child window acting as an input device. To create a dialog box, you fill an empty dialog box with the controls you want.

Resource Workshop's Dialog editor makes it easy to create and edit dialog boxes. Working with dialog boxes involves four steps:

1. Start the Dialog editor.
2. Create a new dialog box or edit an existing one.
3. Test the dialog box.
4. Save the dialog box.

**Customizing the Dialog editor**
Before you work on a dialog box, you can customize the Dialog editor by choosing Options| Preferences. You see the Preferences dialog box, where you can set configuration options.

**See Also**
Creating a new dialog box
Editing an existing dialog box
Saving a dialog box
Testing a dialog box
Using the Dialog editor

## Using the Dialog editor

You use the Dialog editor to create and edit dialog boxes. The Dialog editor displays:

- The Control, Align, and Option menus you use to manipulate the dialog box and its controls
- An empty dialog box on the left side of the screen
- The Tools palette that contains control and mode tools
- The Alignment palette you use to align controls
- The Caption window where you add a new caption to your dialog box or control

## Tools palette

The Tools palette is displayed on the right side of the Dialog editor screen. It contains icons for:

- mode tools that put the Dialog editor in various operational modes (first column)
- the standard Windows controls (second and third columns)
- the Borland custom controls (fourth column)
- any custom controls you've loaded that Resource Workshop recognizes

**See Also**

Borland custom control tools
Control tools
Mode tools
Working with controls

## Control tools

The standard Windows control tools are displayed in the second and third columns of the Tools palette. Control tools make it easy to work with controls.

To put a control in your dialog box, just select a control tool with your mouse and drag it to your dialog box. Release the mouse button to place the control.

Here are the standard Windows control tools on the Tools palette:

 Black Frame

 Black Rectangle

 Check Box

 Combo Box

 Edit Text

 Group Box

 Horizontal Scroll Bar

 Icon

 List Box

Push Button

 Radio Button

 Text Static

 Vertical Scroll Bar

The Custom Control tool in the lower right corner of the Tools palette brings up the New Custom Control dialog box, where you add a new control to your dialog box. Custom controls are active only if you've used Options|Install Control Library to install a custom control library.

**See Also**
Controls
Working with controls

## Black Rectangle tool

The Black Rectangle tool puts a rectangle in your dialog box that's the same color as the current window frame.

Click in the dialog box to place the rectangle. Double-click in the rectangle to bring up the Static Style dialog box, where you customize the rectangle.

**See Also**
Static controls

## Check Box tool

The Check Box tool puts a rectangular button in your dialog box with text to the left or right.

When a check box is turned on, an X appears on the button. When a check box is turned off, the X disappears. Check boxes are often used to represent Boolean (on/off) states for individual options.

Click in the dialog box to place the check box. Double-click in the check box to bring up the Button Style dialog box, where you customize the check box.

**See Also**
Button controls

## Combo Box tool

The Combo Box tool puts a box in your dialog box that's a combination of a list box and a static control or edit text control.

Click in the dialog box to place the combo box. Double-click in the combo box to bring up the Combo Box Style dialog box, where you customize the combo box.

**See Also**
Combo Box controls
Edit text controls
List Box controls
Static controls

## Custom Control tool

The Custom Control tool brings up the New Custom Control dialog box, where you can put a control in your dialog box that is different from the standard Windows or Borland types and is recognized by Resource Workshop.

**See Also**
Borland custom controls
Custom controls

## Edit Text tool

The Edit Text tool puts a rectangle in your dialog box. The user can enter text from the keyboard into this box.

Place the edit text in your dialog box by moving the cursor. Double-click to bring up the Edit Text Style dialog box, where you customize the text.

**See Also**
Edit Text controls

## Group Box tool

The Group Box tool puts a rectangular box around a group of controls in your dialog box. It's used to visually group controls together. You can include a caption in the upper left corner of the group box.

Click in the dialog box to place the group box. Double-click in the group box to bring up the Button Style dialog box, where you customize the group box.

**See Also**
Button controls

## Icon tool

The Icon tool puts an icon in your dialog box. Click in the dialog box to place the icon. Double-click in the icon to bring up the Static Style dialog box, where you customize the icon.

**See Also**
Static controls

## Horizontal Scroll Bar tool

The Horizontal Scroll Bar tool puts a horizontal rectangle in your dialog box with a direction arrow on each end.

Click in the dialog box to place the scroll bar. Double-click in the scroll bar to bring up the Scroll Bar Style dialog box, where you customize the scroll bar.

**See Also**
Scroll bar controls

## List Box tool

The List Box tool puts a rectangle in your dialog box. The rectangle usually includes a list of strings. It can also contain a visual representation of the list.

Usually, a user can browse through the list box, then select one or more items.

Click in the dialog box to place the list box. Double-click in the list box to bring up the List Box Style dialog box, where you customize the list box.

**See Also**
List Box controls

## Push Button tool

The Push Button tool puts a rectangular button in your dialog box. The user "presses" the button to select an action. Push buttons always contain text, so you need to specify a caption for each one.

Click in the dialog box to place the button. Double-click in the button to bring up the Button Style dialog box, where you customize the button.

**See Also**
Button controls

## Radio Button tool

The Radio Button tool puts a circular button in your dialog box with text on its left or right side. When the button is turned on, a solid dot fills the circle. Radio buttons are used in groups to represent related but mutually exclusive options.

Click in the dialog box to place the button. Double-click in the button to bring up the Button Style dialog box, where you customize the button.

**See Also**
Button controls

## Text Static tool

The Text Static tool puts text in your dialog box. Click in the dialog box to place the static text. Double-click in the static text to bring up the Static Style dialog box, where you customize the static text.

**See Also**
Static controls

## Vertical Scroll Bar tool

The Vertical Scroll Bar tool puts a vertical rectangle in your dialog box with a direction arrow on each end.

Click in the dialog box to place the scroll bar. Double-click in the scroll bar to bring up the Scroll Bar Style dialog box, where you customize the scroll bar.

**See Also**
Scroll bar controls

## Black Frame tool

The Black Frame tool puts a rectangular empty frame in your dialog box. The frame is the color of the current window frame.

Click in the dialog box to place the frame. Double-click in the group box to bring up the Static Style dialog box, where you customize the frame.

**See Also**
Static controls

## Mode tools

Mode tools are displayed in the left column of the Tools palette. Mode tools put the Dialog editor in various modes.

Here's a list of the Mode tools on the Tools palette:


Duplicate


Selector


Set Groups

Set Order


Tab Set

Test

Undo

## Selector tool

You use the Selector tool to select one or more controls in your dialog box. To select a single control, simply click the Selector inside the control. To select more than one control, click and drag your mouse around the controls you want to select.

When you use the Selector tool, your cursor is the shape of an arrow. Choose the Selector tool to exit another mode and return your cursor to the arrow shape.

**See Also**
Selecting multiple controls

## Tab Set tool

Use the Tab Set tool (the horizontal arrows icon) to specify which controls are tab stops, so that users can use Tab to move to the control.

**See Also**
Specifying which controls are tab stops

## Set Groups tool

Use the Set Groups tool (the G icon) to define a group of controls.

**See Also**
Grouping related controls

## Set Order tool

Use the Set Order tool (the 1,2 icon) to reorder controls.

**See Also**
Reordering controls


## Test tool

Use the Test tool to test the controls in your dialog box.

**See Also**
Testing controls

## Duplicate tool

Use the Duplicate tool (the box and arrow icon) to add multiple copies of a control to your dialog box.

For multiple selected controls, the Duplicate tool has the same effect as Align|Array. When a single control is selected, it has the same effect as Edit|Duplicate.

**See Also**
Adding multiple copies of controls


## Undo tool

Use the Undo tool to undo any editing you do in the Dialog editor. The Undo tool works on commands that affect groups of controls as wells as commands that change single controls.

## Alignment palette

The Alignment palette makes it easy to align controls.

First select the controls you want to align and place them in your dialog box. Then align the controls with a tool from the Alignment palette.

Here are the tools you can choose from:


Bottoms


Horizontal Center in Dialog


Horizontal Centers


Left Sides


Right Sides


Tops


Vertical Center in Dialog


Vertical Centers

You can also use the Align Controls dialog box to align controls.

**See Also**
Selecting multiple controls

## Left Sides tool

The Left Sides tool aligns controls so their left sides are on the left side of the of the selection frame.

**See Also**
Selecting multiple controls

## Horizontal Centers tool

The Horizontal Centers aligns controls so their horizontal centers are in the center of the selection frame.

**See Also**
Selecting multiple controls

## Right Sides tool

The Right Sides tool aligns controls so their right sides are on the right side of the selection frame.

**See Also**
Selecting multiple controls

## Horizontal Center in Dialog tool

The Horizontal Center in Dialog tool moves the selection frame horizontally so it's centered in the dialog box. The relative position of the individual controls within the selection frame is unchanged.

**See Also**
Selecting multiple controls

## Tops tool

The Tops tool aligns controls so their tops are at the top of the selection frame.

**See Also**
Selecting multiple controls

## Vertical Centers tool

The Vertical Centers tool aligns controls so their vertical centers are in the center of the selection frame.

**See Also**
Selecting multiple controls

## Bottoms tool

The Bottoms tool aligns controls so their bottoms are at the bottom of the selection frame.

**See Also**
Selecting multiple controls

## Vertical Centers in Dialog tool

The Vertical Center in Dialog tool moves the selection frame vertically so it's centered in the dialog box. The relative position of the individual controls within the selection frame is unchanged.

**See Also**
Selecting multiple controls

## Caption window

The Caption window is where you add a caption to your dialog box or to a dialog control:

1. Click the title bar of the dialog box or control to select it.
2. Press F6 to move to the Caption window. Type in the caption.
3. Press Enter. The new caption appears on the dialog box title bar or in the control.

You can also use the Window Style dialog box to add a caption to the dialog box. Use one of the Style dialog boxes to add a caption to a control.

**See Also**
Editing controls

## Creating a new dialog box

To create a new dialog box,

1. Create a new project or open an existing one.
2. Choose Resource|New. Resource Workshop displays the New Resource dialog box.
3. In the Resource Type list box, double-click DIALOG.
4. Click OK to place the dialog box in the current project file or choose another project file.

You're now in the Dialog editor, where you can customize your dialog box.

**See Also**
Creating a new project
Customizing your dialog box
Using the Dialog editor
Opening an existing project

## Editing an existing dialog box

To edit a dialog box that already exists in a project file,

1. Open the project that contains the dialog box you want to edit.
2. Double-click the dialog resource name you want to edit, or select it and choose Resource|Edit.

You're now in the Dialog editor, where you can customize your dialog box.

**See Also**
Customizing your dialog box
Editing a dialog box resource as text
Opening an existing project
Using the Dialog editor

## Customizing a dialog box

Once in the Dialog editor, you define your dialog box by:

- adding a caption
- assigning a custom class
- including a menu
- choosing a window type, frame style, and dialog box style
- specifying fonts
- using Borland custom controls

To make changes to the dialog box window or set its attributes, you first need to select it by clicking its title bar or outer edge. Once it's selected, you see a selection frame around it and you can:

- resize the dialog box window by dragging the appropriate edge or corner or by using the Align|Size command.
- move the dialog box window by dragging the mouse cursor from within the window, or by using the arrow keys.
- press Enter or double-click anywhere in the dialog box window to display the Window Style dialog box, where you set dialog box attributes

You can also add, change, group, reorder, move, resize, or delete dialog controls so that your dialog box functions the way you want it to.

**See Also**

Controls
Setting dialog box position
Working with controls

## Adding a caption to a dialog box

To add a caption to a new dialog box,

1. Open the Window Style dialog box by double-clicking the title bar or outer edge of the dialog box.
2. In the Caption input box, type the caption you want to appear at the top of your dialog box.
3. Turn on the Caption radio button.
4. Click OK.

You can also enter a caption in the Caption window.

## Including a menu in a dialog box

Because it's really a window, a dialog box could include a menu. For example, some applications use a dialog box for the main window, in which case the dialog box would need a menu.

To include a menu in your dialog box,

1. Define the menu as a separate resource and add it to the project. Remember the value you enter in the Item Text or Item ID input box.

2. Open the Dialog editor for the dialog box you want to add the menu to.

3. Open the Window Style dialog box by double-clicking the title bar or outer edge of the dialog box.

4. Type the menu's resource name or numeric ID in the Menu input box.

In standard drawing modes, the Dialog editor doesn't display the menu.

## Choosing a window type, frame style and dialog box style

You choose a window type for your dialog box by turning on one of the Window Type radio buttons in the Window Style dialog box.

You choose a frame style for your dialog box by turning on one of the Frame Style radio buttons in the Window Style dialog box. The frame style of the dialog box determines the appearance of the dialog box frame and whether the dialog box displays a title bar at the top.

You choose a dialog style for your dialog box by turning on one or more of the Dialog Style check boxes in the Window Style dialog box. The dialog box style determines what the dialog box looks like and how the user can work with it.

## Specifying dialog box fonts

To choose how text is displayed in your dialog box,

1. Open the Window Style dialog box by double-clicking the title bar or outer edge of the dialog box.

2. Click the Fonts button to open the Select Font dialog box, where you select a typeface, size, and style for text in your dialog box. The characters displayed at the bottom of the Select Font dialog box show the current typeface, size, and styles you've selected.

The default font text in for dialog boxes is the 10-point System font.

## Assigning a custom class to a dialog box

If you're an experienced Windows programmer, you might want to assign a custom class to your dialog box. Then you can process dialog box messages with your own windows procedures instead of using Windows procedures.

Another reason for assigning a custom class is to make the dialog box a Borland-style dialog box.

To assign a custom class to a dialog box;

1. Open the Window Style dialog box by double-clicking the title bar or outer edge of the dialog box, or by selecting the window and pressing Enter.

2. Type the class name in the Class input box. If you're creating a Borland-style dialog box, enter "bordlg."

**See Also**
Customizing existing applications for BWCC
Using Borland custom controls

## Setting dialog box position

If your dialog box uses the WS_OVERLAPPED style, you can let Windows position it on the screen. To give control of the dialog box's position to Windows,

1. Select the dialog box frame by clicking on its edge or the title bar.

2. Choose Align|Size.

3. In the Size Dialog dialog box, click the Set by Windows radio button (one of the Horizontal radio buttons). Resource Workshop disables the X input box, indicating that Windows has control of the dialog box.

This option is generally used for dialog box frames that function as main windows.

## Controls

Controls are individual components of dialog boxes. They let users interact with dialog box data. In general, controls fall into these categories:
Button controls
Combo Box controls
Custom controls
Edit Text controls
List Box controls
Scroll Bar controls
Static controls

Once you've defined your dialog box, you can work with controls to make the dialog box do what you want it to.

**See Also**
Families of controls
Working with controls

## Families of controls

Each control you use comes from one of five family groups:

- Standard Windows controls such as push buttons, check boxes, list boxes, and radio buttons. Icons for the standard Windows controls appear in the second and third columns of the Tools palette. Standard Windows controls are always available.

- Borland Windows Custom Controls (BWCC). These controls (including radio buttons, check boxes, and push buttons) offer both visual and functional enhancements over the standard Windows controls. BWCC is always available; these tools appear in the fourth column of the Tools palette.

- Custom controls whose class is recognized by Resource Workshop. These controls are stored in a dynamic-link library (DLL) that includes the **ListClasses** function. When the DLL file is installed, the icons for these controls appear in one or more additional columns in the Tools palette, starting to the right of the BWCC tools.

- Custom controls whose class is recognized by the Windows SDK dialog editor. These controls are stored in a DLL file that includes their bitmaps but does not include the **ListClasses** function. They are not represented in the Tools palette, but their names appear in the drop-down list of the New Custom Control dialog box. When you add one of these controls to a dialog box, its bitmap appears on the screen when Resource Workshop is in WYSIWYG display mode.

- Custom controls whose class is not recognized by Resource Workshop or the Windows SDK dialog editor. Resource Workshop adds their names to the drop-down list in the New Custom Control dialog box, but they appear on the screen in WYSIWYG mode as gray rectangles.

**See Also**
Control tools
Borland custom control tools

## Button controls

Use the Control menu or one of following tools to add button controls to your dialog box:
Check Box
Group Box
Push Button
Radio Button

You use the Button Style dialog box to customize a button control. To display this dialog box, double-click the button control you want to modify.

You customize a button control by:

- specifying the caption you want displayed with the button in the Caption input box.
- specifying the type of caption with the Caption radio buttons. If you're adding text to your button, turn on one of the Justification radio buttons.
- entering a number or an alphanumeric expression for the control in the Control ID input box.
- choosing the button's attributes with the Attributes check boxes.
- adding a scroll bar to the button with the Scroll Bar check boxes.
- choosing the type of button you want with the Button Type radio buttons.

**See Also**
Push button control ID values
Working with controls

## Push button control ID values

Windows defines a set of control ID values and names for the standard push buttons used to exit dialog boxes. To use these buttons, you need to enter the predefined value and name in the Button Style dialog box:

- Enter the ID value in the Control ID input box
- Enter the button's name in the Caption input box

The predefined values and names are:

| IDValue | ID Name | Button Type |
|---|---|---|
| 1 | IDOK | OK |
| 2 | IDCANCEL | Cancel |
| 3 | IDABORT | Abort |
| 4 | IDRETRY | Retry |
| 5 | IDIGNORE | Ignore |
| 6 | IDYES | Yes |
| 7 | IDNO | No |

**Note:** The ID name must be entered in uppercase letters.

## Combo box controls

A combo box is a combination of a list box and a static control or an edit text control.

Use the Control menu or the Combo Box tool to put a combo box control in your dialog box.

You customize a combo box control with the Combo Box Style dialog box. To display this dialog box, double-click the combo box control you want to modify.

You customize a combo box control by:

- specifying the caption you want displayed with the static control in the Caption input box. Specify the type of caption with the Caption radio buttons.
- entering a number or an alphanumeric expression for the control in the Control ID input box.
- choosing the attributes with the Attributes check boxes.
- choosing one or more of the Type radio buttons to further define the combo box.
- choosing one or more of the Owner Drawing radio buttons to indicate whether the list contained in the list box should be drawn by the list box itself or by the application.

**See Also**

Edit text controls
List box controls
Static controls
Working with controls

## Custom controls

If you want to use a control that doesn't fit into one of the predefined Windows types, you can create a custom control.

There are two types of custom controls: those that you can install and those that are application specific. Installable custom controls must be implemented using a dynamic link library (DLL). Custom controls specific to an application are implemented in the application itself. Resource Workshop draws them as either gray boxes or empty frames.

If you want to create your own custom controls, you need to design them and store them in DLLs.

**Displaying custom controls**
Before you add any custom controls to your dialog box, you may want to decide how they'll be displayed in the Dialog editor. Open the Preferences dialog box to see if you've chosen to display your dialog boxes in Draft or Normal mode.

If either of these buttons is on, turn off the Draw Custom Controls as Frames radio button so that you see more than just a frame for your custom controls.

**See Also**
Adding a custom control
Installing a custom control library
Working with controls

## Adding a custom control

Once you've installed a DLL file containing custom controls, you can add any of those controls to your dialog boxes. If your custom controls are recognized by Resource Workshop, the icons appear on the right side of the Tools palette and you can select them directly.

If your custom controls are of the types not recognized by Resource Workshop, you must follow these steps:

1. Click the Custom Control tool in the Tools palette or choose Control|Custom. You see the New Custom Control dialog box.

2. In the Class drop-down box, choose the custom control you want. The Borland custom controls are automatically displayed in this box.

3. Choose the control you want. You see a sample of the custom control in the middle of the dialog box.

4. Click OK. The mouse cursor turns into a cross hair, indicating that it's ready to place the custom control.

5. Click in the dialog box where you want to place the custom control.

**See Also**
Installing a custom control library

## Installing a custom control library

Custom controls are stored in dynamic link library (DLL) files. If you want to add custom controls to your dialog box, you need to install the appropriate DLL file(s). Then the custom controls in that DLL will be available just like any standard Windows control.

To install a DLL file containing a custom control library,

1. In the Dialog editor, choose Options|Install Control Library. You see the Install a New Control Library dialog box.

2. Specify the custom control DLL file in the File input box and choose OK.

Now the controls contained in that DLL file are available for you to add to your dialog box.

**See Also**
Adding a custom control

## Edit text controls

An edit text control lets the user enter text from the keyboard. Use the Control menu or the Edit Text tool to place edit text controls in your dialog box.

You customize an edit text control with the Edit Text Style dialog box. To display this dialog box, double-click the edit text box control you want to modify.

You customize an edit text control by:

- specifying the caption you want displayed with the edit text in the Caption input box. Specify the type of caption with the Caption radio buttons.
- justifying the text in an edit text control with the Justification radio buttons.
- entering a number or an alphanumeric expression for the control in the Control ID input box.
- choosing the edit text attributes with the Attributes check boxes.
- choosing the orientation of any scroll bars you add to the edit text control with the Scroll Bar check boxes. You can add horizontal and vertical scroll bars if the edit text control allows more text than can fit in the control border.
- choosing how the user will type text with the Line radio buttons.
- choosing how the text is displayed when it's typed with the Case radio buttons.
- choosing how the text is converted when its typed with the Text Conversion check boxes (the first set of check boxes).
- choosing how the edit text will scroll with the Automatic Scroll check boxes.

**See Also**
Working with controls

## List box controls

A list box is a rectangle containing a list of text strings. Usually, a user can browse through the list box and select one or more items. The list box sends a message to the parent window about the selected item(s).

If the list of items exceeds the length or width of the list box, you can add scroll bars to the list box.

Use the Control menu or the List Box tool to place list box controls in your dialog box.

You customize a list box control with the List Box Style dialog box. To display this dialog box, double-click the list box control you want to modify.

You customize your list box control by:

- specifying the caption you want displayed with the list box in the Caption input box.
- specifying the type of caption with the Caption radio buttons.
- entering a number or an alphanumeric expression for the control in the Control ID input box.
- choosing the list box's attributes with the Attributes check boxes.
- choosing the orientation of any scroll bars you add to the list box with the Scroll Bar check boxes. You can add scroll bars to a list box if the list of items exceeds the length or width of the list box.
- choosing whether the list in the list box should be drawn by the list box or the application with the Owner Drawing radio buttons.
- choosing one or more of the List Box check boxes to further define the list box.

**See Also**
Working with controls

## Scroll Bar controls

Windows, dialog boxes, and list boxes use scroll bars to indicate that there is more information than can currently be displayed. A scroll bar is a rectangle with direction arrows at each end. Between the arrows, a square icon (sometimes called a thumb) indicates the approximate position of the display relative to the full range of information.

Use the Control menu or the Horizontal Scroll Bar or Vertical Scroll Bar tools to add horizontal and vertical scroll bar controls to your dialog box.

You customize your scroll bars with the Scroll Bar Style dialog box. To display this dialog box, double-click the scroll bar control you want to modify.

You customize the scroll bar control by:

- specifying the caption you want displayed with the scroll bar in the Caption input box.
- specifying the type of caption with the Caption radio buttons.
- entering a number or an alphanumeric expression for the control in the Control ID input box.
- choosing the scroll bar's attributes with the Attributes check boxes.
- choosing the scroll bar's orientation with the Scroll Bar radio buttons.
- choosing the scroll bar's alignment with the Alignment radio buttons.

**See Also**
Working with controls

## Static controls

A static control displays text or art the user can't change. You can use static controls to label portions of your dialog box or to present information graphically.

Use the Control menu or one of following tools to add static controls to your dialog box:
Black Frame
Black Rectangle
Icon
Text Static

You customize static controls with the Static Style dialog box. To display this dialog box, double-click the static control you want to modify.

You customize the static control by:

- specifying the caption you want displayed with the static control in the Caption input box. Specify the type of caption with the Caption radio buttons.
- entering a number or an alphanumeric expression for the control in the Control ID input box.
- choosing the static controls' attributes with the Attributes check boxes.
- defining what's displayed by the static control with Control Type radio buttons.
- disabling character underlining with the No Character Underline check box.

**See Also**
Iconic static controls
Working with controls

## Iconic static controls

Resource Workshop lets you display icons in a dialog box. The icon must be part of the current project as an embedded or linked resource.

To place an iconic static control in your dialog box,

1. Click the Icon tool and drag it to your dialog box. Place the frame where you want the icon to appear.

2. Double-click inside the control to display the Static Style dialog box.

3. Enter the name or identifier of the icon resource in the Caption input box and choose the appropriate Caption radio button.

4. If you use an identifier - either its name or numeric value - as the Caption, turn on the Number radio button. If you don't use an identifier as the caption, turn on the Text radio button.

5. Choose OK. The icon is displayed in your dialog box.

6. If you want to edit the icon, double-click it to display the Static Style dialog box again. Then choose the Edit Icon button to start the Paint editor.

## Working with controls

Once you've defined a dialog box, you can create and manipulate its controls. In the Dialog editor, the Tools palette makes it easy to work with controls.

You manipulate controls by:

- adding controls to a dialog box
- adding multiple copies of a control
- aligning controls with a grid
- changing a control's class
- editing controls
- editing groups of controls
- grouping related controls
- moving and resizing a control
- reordering controls
- specifying which controls are tab stops

## Adding controls to a dialog box

One way to add a new control to your dialog box is to click the control you want in the Tools palette. Your cursor changes to indicate the type of control you're placing. Click the spot you want to place the control in the dialog box.

If you select a control from the Tools palette and then change your mind about placing it, choose the Selector tool. Your cursor returns to the arrow shape and you can choose another control.

You can also use the Control menu to add controls to your dialog box.

Once you add a control to your dialog box, you can specify exact coordinates for the control, as well as its width and height.

**See Also**
Specifying x- and y-coordinates for a control

## Adding multiple copies of a control

Once you place a control in your dialog box, you can place additional copies of the same control. Move to where you want to place another copy and click the right mouse button.

You can also place multiple, identical copies of a control in rows or columns with the Edit|Duplicate command or the Duplicate tool:

1. Select the control you want to duplicate.
2. Select the Edit|Duplicate command or the Duplicate tool. The Duplicate Control dialog box is displayed.
3. Enter the number of rows and columns you want and the spacing between rows and columns.
4. Click OK.

When only a single control is selected, the Duplicate tool has the same effect as the Edit|Duplicate menu command. For multiple selected controls, it has the same effect as the Align|Array command.

**See Also**
Selecting multiple controls

## Aligning controls with a grid

You can use a grid to help align the controls in your dialog box. Display a grid in your dialog box window by choosing Align|Grid. You see the Set Grid Attributes dialog box, where you specify the width and height of each cell in the grid and the grid type.

Once you specify the grid, use the mouse and the cursor keys to align the controls in the dialog box:

1. Place a selection frame around the controls you want to align.
2. Use the mouse to position the frame.
3. Place the Selector tool inside the frame.
4. Press the left mouse button.
5. Move the mouse.

If the grid is absolute, the selection frame snaps to the grid's absolute position when you release the mouse. If the grid is relative, the frame moves relative to the grid's previous position.

Once the selection frame is in place, use the arrow keys to fine-tune its position. The arrow keys move the selection frame in dialog units.

## Changing a control's class

If you're working with custom controls, you use the Generic Control Style dialog box to specify a new class, caption, control ID, and style for the control.

Display this dialog box by holding down the Ctrl key and double-clicking the control. You can also use Tab to select the control, and hold down Ctrl and press Enter.

## Editing controls

Once you've added a control to your dialog box, you can easily modify it by double-clicking it. A dialog box appears with options that can modify the look of your control. The options in this dialog box vary according to the type of control you're working with.

Here's a list of the types of controls you can edit and the dialog box you work with:

| Control Type | Dialog Box |
|---|---|
| Button | Button Style dialog box |
| Combo box | Combo Box Style dialog box |
| Custom | New Custom Control dialog box |
| Edit text | Edit Text Style dialog box |
| List box | List Box Style dialog box |
| Scroll bar | Scroll Bar Style dialog box |
| Static | Static Style dialog box |

## Grouping related controls

Use the Set Groups tool to define groups of controls. Users can move among related controls using the arrow keys.

Here's how to group controls:

1. Move the related controls so they're together.

2. Click the Set Groups tool. Any controls that are already defined as the first member of a group are surrounded by a box.

3. For each group you want to define, click the first member so it's surrounded by a box. You don't have to define the last member of a group. By clicking the first member of each group, you identify the last member of the previous group.

4. When you finish identifying the first member in each group, choose the Selector tool to return to edit mode.

You can also use the Options|Set Groups command to group controls.

## Moving and resizing a control

To move and resize a control, you first need to select it by clicking it. Then move the control by dragging it. Resize the control by moving the mouse cursor to the appropriate edge or corner (until it turns into a double arrow) and dragging the control.

If you want to use the keyboard to move and resize controls, press Tab to select the control. Then use the arrow keys to move what you've selected, and press Enter to confirm the move. Press Esc instead of Enter if you change your mind.

**See Also**
Specifying x- and y-coordinates for a control

## Reordering controls

You can specify the order in which users access the controls in your dialog box. The order is especially important when you've defined groups of related controls.

You specify the order of controls with the Set Order tool:

1. Choose the controls whose order you want to change.
2. Click the Set Order tool. Each control is ordered to show its current place in the overall order.
3. Click the controls you want to assign new order numbers to. The control is assigned the next number.
4. When you finish assigning new order numbers, choose the Selector tool to return to edit mode.

You can also use the Options|Set Order command to reorder controls.

## Specifying which controls are tab stops

When you add a control to a dialog box, it's automatically defined as a tab stop. You can change this so that only some controls are tab stops, and users can use Tab to move only to the controls you want:

To change a tab stop with the Tab Set tool.

1. Click the Tab Set tool. Each control that's currently a tab stop is surrounded by a shaded outline.

2. To set a tab stop, click any control that's not surrounded by a shaded outline. To remove a tab stop, click a control that's already a tab stop.

3. When you finish changing tab stops, choose the Selector tool to return to edit mode.

You can also turn on the Tab Stop check box in the control's Style dialog box to set tab stops. Another way to set tab stops is to use the Options|Set Tabs command.

## Specifying x- and y-coordinates for a control

Once you've added a control to your dialog box, you can specify exact x- and y-coordinates for that control. The x- and y-coordinates control positioning, as well as the size of the control.

Here's how to specify x- and y-coordinates:

1. Choose the control you want to specify coordinates for.

2. Choose Align|Size or press Alt and double-click the mouse. The Size Controls dialog box appears, where you can resize controls.

3. Turn on the Enter Value radio buttons (one of the Horizontal Size and Vertical Size radio buttons).

4. Specify the X, CX, Y, and CY values in dialog units.

## Editing groups of controls

Once you've added controls to your dialog box, you can use the Dialog editor to change groups of controls.

You can select multiple controls, align them, resize them, or place them in columns and rows.

**See Also**

Aligning multiple controls
Placing controls in columns and rows
Resizing multiple controls
Selecting multiple controls

## Selecting multiple controls

Before you align and resize groups of controls, you need to know how to select more than one control at a time.

To select multiple controls,

1. Click the Selector tool.
2. Place the cursor where you want the selection frame to start.
3. Click the left mouse button and hold it down.
4. Drag your mouse so that the selection frame surrounds all the controls you want to select.
5. Release the mouse button.

Make sure the selection frame is wide enough and high enough to surround the outer edges of all selected controls.

## Aligning multiple controls

It's easy to line up groups of controls with the Align menu. Here's how:

1. Select the controls you want to align.
2. Choose Align|Align. You see the Align Controls dialog box, where you align controls.
3. When you choose the vertical and horizontal alignment you want, click OK.

You can undo the alignment by choosing Edit|Undo or by pressing Alt+Backspace. You may have to choose Undo more than once.

You can also use the Alignment palette tools to align controls.

## Resizing multiple controls

To resize a single control, you can select it and drag the appropriate edge or corner. The Dialog editor also makes it easy to resize groups of controls. Here's how:

1. Select the controls you want to align.

2. Choose Align|Size. You see the Size Controls dialog box, where you resize controls.

3. After you choose the vertical and horizontal sizing options you want, click OK.

You can undo the sizing options by choosing Edit|Undo (or press Alt+Backspace). You may have to choose Undo more than once.

## Placing controls in columns and rows

You use the Align|Array command to arrange controls in an array of columns and rows. Here's how:

1. Select the controls you want to arrange in columns and rows.
2. If necessary, expand the selection frame so that the columns and rows you want will fit. You can drag an edge or a corner to resize the border.
3. Choose the Align|Array command. You see the Form Controls Into An Array dialog box, where you can put controls into columns and rows.
4. In the Array Layout input boxes, specify the number of rows and columns you want.
5. Turn on one of the Order radio buttons to specify how you want to order the controls in this group.
6. Click OK when you finish.

You can also use the Duplicate tool to place controls in columns and rows.

You can undo the formatting by choosing Edit|Undo (or press Alt+Backspace). You may have to choose Undo more than once.

## Testing a dialog box

Choose Options|Test Dialog or the Test tool to test your dialog box. You can press Tab and the arrow keys to see how you can move around in your dialog box, or you can type text to see how text is scrolled in an edit text control. Check to see if your controls are in the order you want them.

When you test a dialog box, the status line at the bottom of the Dialog editor says "Test."

To leave test mode and return to edit mode, do any of the following:

- Click the dialog box's OK or Cancel button.
- Choose Options|Test Dialog again.
- Press Enter.
- Click the Selector tool twice.

## Saving a dialog box

It's a good idea to save your changes as you go along, rather than waiting for Resource Workshop to prompt you when you close the project.

There are two primary options you can use to save changes you've made to an dialog box. You can save the entire project or save the dialog box in a resource script file.

**See Also**
Saving the entire project
Saving the dialog box in a resource script file

## Saving a dialog box in a resource script file

You can store your dialog box resource in a dialog (.DLG ) file. A dialog file usually contains a resource script definition for one or more dialog boxes. If you add a dialog file to a project, Resource Workshop automatically adds an **rcinclude** statement in the Project window, which references your dialog file.

To store dialog boxes in a .DLG file, you need to add a dialog file to your project. Then you can choose to store dialog boxes in this dialog file as you create them.

Here's how the process works:

1. Choose File|Add to Project. The Add File to Project dialog box appears.
2. Choose .DLG Resource Script in the Files Type drop-down box.
3. Type a new name in the File Name input box.
4. Choose OK. Click Yes when Resource Workshop asks you if you want to create the file.

You've created an empty dialog file. To create a new dialog box and add it to the DLG file,

1. Choose Resource|New. The New Resource dialog box appears.
2. Select the dialog script file you just created in the Place Resource In combo box.
3. In the Resource Type list box, double-clicked DIALOG.
4. Click OK.

You see your new dialog box in the Dialog editor, ready to customize with controls. When you finish customizing your dialog box and exit the Dialog editor, you see a DIALOG entry indented under the **rcinclude** for the .DLG file in the Project window.

**See Also**
Dialog (.DLG ) files

## Editing a dialog box resource as text

In addition to creating and modifying dialog boxes directly with the Dialog editor, you can work with the dialog box's resource script.

To work with the resource script for a dialog box, select the dialog box in the Project window by clicking it. Then choose Resource|Edit As Text. Resource Workshop brings up the source script for the resource in its internal text editor.

**See Also**
Resource script language
Using a text editor

## Window Style dialog box

Use the Window Style dialog box to specify the details of your dialog box. To open this dialog box, select the dialog box and press Enter or double-click the dialog box's title bar or outer edge.

**Caption input box**
The Caption input box is where you enter the caption of the dialog box.

**Class input box**
The Class input box is where you assign a custom class to your dialog box.

**Menu input box**
The Menu input box puts a menu in your dialog box.

**Window Type radio buttons**
The Window Type radio buttons choose a type for your dialog box.

**Frame Style radio buttons**
The Frame Style radio buttons choose your dialog box's frame style.

**Face input box**
The Face input box is where you enter the typeface for the text in your dialog box.

**Size input box**
The Size input box is where you enter the font's point size.

**Dialog Style check boxes**
The Dialog Style check boxes customize your dialog box.

**Fonts button**
Press the Fonts button to display the Select Font dialog box, where you choose the font for the text in your dialog box.

**See Also**
Customizing a dialog box

## Caption input box

The Caption input box is where you enter the caption you want displayed on the dialog box. Turn on the Caption radio button (one of the Frame Style radio buttons) to display the caption in a title bar at the top of the dialog box.

You can also enter the dialog box caption in the Caption window.

**See Also**
Adding a caption

## Class input box

The Class input box is where you assign a custom class to your dialog box. A custom class designation lets your application process dialog box messages with your own windows procedures instead of using Windows procedures.

If you enter `bordlg` in this input box, your dialog box is implemented as a Borland-style dialog box, like the ones that are used in Resource Workshop. Use the Borland custom control tools to put Borland-style controls in your dialog box.

Using the Borland custom control class improves the appearance of your dialog window by painting the background with a brush that varies according to the target display device. For screens of VGA and higher resolution, the background is a fine grid of perpendicular white lines, giving the effect of "chiseled steel." For EGA and monochrome screens, the background is white.

The Borland custom control class also optimizes the drawing of dialog boxes by calling the custom control routines directly, instead of waiting for Windows to paint the controls.

**See Also**
Assigning a custom class to a dialog box
Using Borland Windows custom controls

## Menu input box

The Menu input box is where you put a menu in your dialog box. Just type the menu's resource identifier or resource ID.

The Dialog editor won't display the menu resource exactly as you define it. Instead, it displays a pop-up called "Menu" that includes a single menu item called "Item."

## Window Type radio buttons

The Window Type radio buttons choose a type for your dialog box.

**Popup**
Turn the Popup radio button on if your dialog box is a pop-up window. Because most dialog boxes are pop-ups, this is the default.

**Overlapped**
Turn the Overlapped radio button on to define your dialog box as an overlapped pop-up window that can be partially covered by another. Define a dialog box as overlapped only when it's the main window in the application.

**Child**
Turn the Child radio button on if the dialog box is a child of the current window.

**Iconic Popup**
Turn the Iconic Popup radio button on to define your dialog box as a pop-up window that's originally displayed as an icon.

## Frame Style radio buttons

The dialog box's frame style controls the dialog box frame and whether the dialog box displays a title bar at the top.

**Dialog Frame**
Turn the Dialog Frame radio button on if you want a double border, without a title bar, around your dialog box.

**Border**
Turn the Border radio button on if you want a single, thin border, without a title bar, around your dialog box.

**Caption**
The Caption radio button creates a single, thin border around the dialog box and a title bar for the caption at the top of the dialog box. This is the default.

**No Border**
If you don't want a border or a title bar on your dialog box, turn the No Border radio button on.

## Dialog Style check boxes

The Dialog Style check boxes determine what the dialog box looks like and how the user can work with it.

**System Menu check box**
The System Menu check box inserts a system menu on the left side of the title bar. A System menu is also called a Control menu.

- The system menu appears only if you turn on the Caption radio button (one of the Frame Style radio buttons).
- If you turn on the Child radio button (one of the Window Type radio buttons), you see a close box instead of a Control menu.

**Thick Frame check box**
The Thick Frame check box places a thick frame around the dialog box.

**Vertical Scroll check box**
The Vertical Scroll check box inserts a vertical scroll bar in the dialog box frame.

**Horizontal Scroll check box**
The Horizontal Scroll check box inserts a horizontal scroll bar in the dialog box frame.

**Minimize Box check box**
The Minimize Box check box adds a Minimize button to the right side of the dialog box title bar. The minimize box appears only if you turn on the Caption radio button (one of the Frame Style radio buttons).

**Maximize Box check box**
The Maximize Box check box adds a Maximize button to the right side of the dialog box title bar. The maximize box appears only if you turn on the Caption radio button (one of the Frame Style radio buttons).

**Absolute Align check box**
The Absolute Align check box makes the dialog box coordinates relative to the display screen rather than the parent window.

**System Modal check box**
Turn on the System Modal check box if you want to make the dialog box modal. This means that the user can't switch to anything else until the dialog box is closed.

**Local Edit check box**
The Local Edit check box allocates any edit text controls to the application's local heap. Turn this check box on if your application uses the EM_SETHANDLE and EM_GETHANDLE messages.

**Modal Frame check box**
The Modal Frame check box frames the window with a combination of the dialog frame and the caption styles. This option also lets users move the dialog box.

**No Idle Messages check box**
The No Idle Messages check box suppresses sending idle (WM_ENTERIDLE) messages to the application's main window. For this option to have any effect, you must also turn on the System Modal check box.

**Clip Children check box**
The Clip Children check box protects the client area of child windows from being drawn on by the dialog box window.

**Clip Siblings check box**
The Clip Siblings check box protects the siblings of this window, and restricts drawing to this window. This option is not required for pop-up windows, but can be useful for child dialog windows.

**Visible check box**

The Visible check box makes a modeless dialog box visible before the return from **CreateDialog**. This option has no effect on modal dialog boxes (the usual kind of dialog box). By default, this option is not checked (NOT WS_VISIBLE).

## Thick Frame check box

If you want a thick frame around the dialog box when it's displayed in your application, turn on the Thick Frame check box. Use this option if you want the dialog box to be resizeable.

**Note:** Don't confuse this option with the Thick Frame radio button in the Options|Preferences dialog box. That option defines what the dialog box will look like when you select it in the Dialog editor.

## Select Font dialog box

The Select Font dialog box is where you choose the font, size, and font style for the text in your resource. Your choice is displayed at the bottom of the dialog box.

Display this dialog box with the Text|Font command or with the Fonts button in the Windows style dialog box.

**Face Name drop-down box**
The Face Name drop-down box is where you choose the typeface for the text in your resource.

**Size input box**
The Size input box is where you choose the font point size.

**Style check boxes**
The Style check boxes select the font style. The only style that you can select for dialog boxes is Bold.

**Bold check box**
The Bold check box makes the font bold.

**Italic check box**
The Italic check box italicizes the font.

**Underline check box**
The Underline check box underlines the font.

**Strikeout check box**
The Strikeout check box strikes out the font.

## Caption input box

Type the caption you want displayed with the control in the Caption input box. The type of control determines where the caption displays. For example, in a group box, the caption displays at the top left. In a push button, the caption displays inside the button.

Once you enter a caption, turn on one of the Caption radio buttons to choose how the caption displays. (These radio buttons don't apply to Borland custom controls.)

Not all controls display a caption. For example, a list box does not display the text specified in its caption.

You can also enter a caption with the Caption window.

## Caption radio buttons

The Caption radio buttons choose how the caption is displayed.

**Number**
If you don't want the caption to be surrounded by quotes, turn on the Number radio button.

**Text**
If you enter a name in the Caption input box, turn on the Text radio button. This surrounds the caption with quotes in the .RC or dialog file source code.

## Control ID input box

Enter the control's identifier in the Control ID input box. Control IDs can be a short integer or an integer expression. By convention, static controls that are not modified at run time are assigned a control ID of -1.

If you enter an alphanumeric identifier, Resource Workshop checks to see if a #define or a constant declaration has already been created for that identifier. If not, Resource Workshop asks if you want to create an identifier.

## Attributes check boxes

The Attributes check boxes select the control's attributes. Not all attributes apply to all controls. The Attributes check boxes are described in alphabetical order; scroll through the topic to find the option you want.

**Auto Horizontal check box**
The Auto Horizontal check box automatically scrolls text to the left when it exceeds the width of the control. This attribute applies only to combo box controls.

**Border check box**
The Border check box draws a border around the control. This attribute applies only to list boxes, edit text, and static controls.

**Disabled check box**
The Disabled check box disables the control by graying it. This prevents the control from responding to user input. This attribute applies to all controls.

**Group check box**
The Group check box identifies the first control in a group.

**Integral Height check box**
The Integral Height check box sizes the list box at run time so all items in the list are completely displayed (the default). If you need to precisely control the height of the list box, turn this option off. This attribute applies only to combo box controls.

**OEM Conversion check box**
The OEM Conversion check box converts text the user enters to the current OEM character set, then reconverts the text to ANSI. This option is useful in file input boxes because it ensures that any file name entered will be translatable into the OEM character set, which is used by the DOS file system. This attribute applies only to combo box controls.

**Sorted check box**
The Sorted check box automatically sorts items in a list box alphabetically. This attribute applies only to combo box controls.

**Tab Stop check box**
The Tab Stop check box lets the user press Tab to access this control.

**Vertical Scroll Always check box**
Turn the Vertical Scroll Always check box on to always place a vertical scroll bar in the list box. The scroll bar is disabled when the list box doesn't contain enough items to scroll. This attribute applies only to combo box controls. This check box is displayed only when you run Resource Workshop under Windows 3.1.

**Vertical Scroll check box**
The Vertical Scroll check box puts a vertical scroll bar in the list box. This attribute applies only to combo box controls.

**Visible check box**
The Visible check box determines whether the control is visible when the dialog box is first displayed. By default, the option is checked (WS_VISIBLE). If the option is not checked (NOT WS_VISIBLE), the control does not appear. The application can call the **ShowWindow** function at run time to make the control appear.

## Tab Stop check box

Turn the Tab Stop check box on if you want the user to be able to press Tab to access this control. When you set this attribute, all controls in the dialog box are tab stops.

You can also use the Tab Set tool to specify which controls are tab stops.

**See Also**
Specifying which controls are tab stops

## Group check box

Turn the Group check box on to indicate the first control within a group. The user can then press the arrow keys to access all controls in the group. This attribute applies to all controls.

You can also use the Set Groups tool to define a group of controls.

**See Also**
Grouping related controls

## Justification radio buttons

The Justification radio buttons justify text for radio buttons, check box buttons, and edit text. For edit text controls, these options apply to multiple-line text.

**Left**
Turn the Left radio button on to justify text to the left or place text to the left of the button.

**Right**
Turn the Right radio button on to justify text to the right or place text to the right of the button.

**Center**
Turn the Center radio button on to center text. This option applies only to edit text controls.

## Scroll Bar check boxes

The Scroll Bar check boxes place scroll bars in your controls. This attribute applies to edit text and list boxes. Buttons can include scroll bars, although they are more commonly found elsewhere in a dialog box.

**Horizontal check box**
The Horizontal check box places a horizontal scroll bar in the control.

**Vertical check box**
The Vertical check box places a vertical scroll bar in the control.

## Button Style dialog box

The Button Style dialog box is where you define button controls. To bring up this dialog box, double-click the button control you want to modify.

**Caption input box**
The Caption input box is where you enter the caption you want displayed with the control.

**Caption radio buttons**
The Caption radio buttons choose how the caption is displayed.

**Control ID input box**
The Control ID input box is where you enter the control's identifier.

**Attributes check boxes**
The Attributes check boxes select the control's attributes.

**Justification radio buttons**
The Justification radio buttons justify text.

**Scroll Bar check boxes**
The Scroll Bar check boxes put scroll bars in your buttons.

**Button Type radio buttons**
The Button Type radio buttons define the type of buttons to include in your dialog box.

## Button Type radio buttons

The Button Type radio buttons define the type of buttons to include in your dialog box.

**Push Button**
Turn on the Push Button radio button to define a button containing text. When the user clicks the button, a BN_CLICKED message is sent to the parent window.

**Default Push Button**
Turn the Default Push Button radio button to define a button containing text. It's identical to a push button, but also includes a bold border indicating that it's the default response when the user presses Enter.

**Check Box**
Turn the Check Box radio button on to define a small, rectangular button that can include text to the left or right of the button. The box is marked with an X when it's selected. It's the application's responsibility to check and uncheck the box when the user turns the box on or off.

**Auto Check Box**
Turn the Auto Check Box radio button on to define a check box that's identical to a regular check box, but Windows does the checking and unchecking instead of the application.

**3-state**
Turn the 3-state radio button on to define a button that's identical to a check box. In addition to being on or off, the button has a third state - it can be grayed to show that its state is unknown or indeterminate. It's the application's responsibility to check, uncheck, and dim the box.

**Auto 3-state**
Turn the Auto Three State radio button on to define a button that's identical to a 3-state button, but Windows does the checking, unchecking, and dimming instead of the application program.

**Radio Button**
Turn the Radio Button radio button on to define a small, circular button that has identifying text to the left or right. When it's selected, the circle is filled with a solid dot and all other mutually exclusive choices are turned off. It's the application's responsibility to fill or clear the dot when the user turns the button on or off.

Radio buttons must appear in groups. Usually, a group of radio buttons presents the user with a group of mutually exclusive options.

When a the user clicks a radio button, the button sends a BN_CLICKED message to the parent window.

**Auto Radio Button**
Turn the Auto Radio Button radio button on to define a button that's identical to a radio button, but Windows fills or clears the dot instead of the application.

**Group Box**
Turn the Group Box radio button on to define a box that groups buttons. You can also include a caption in the upper left corner of the group box.

**User Button**
Turn the User Button radio button on to customize buttons for Windows 2.0 compatibility. Try to avoid using user button controls with Windows 3.x. Instead, use owner draw buttons.

**Owner Draw**
Turn the Owner Draw radio button on to define a radio button that allows the application to paint the button. When the button needs painting, it sends a WM_DRAWITEM message to its parent.

## Scroll Bar Style dialog box

The Scroll Bar Style dialog box is where you define scroll bar controls. To bring up this dialog box, double-click the scroll bar control you want to modify or choose

**Caption input box**
The Caption input box is where you enter the caption you want displayed with the control.

**Caption radio buttons**
The Caption radio buttons choose how the caption is displayed.

**Control ID input box**
The Control ID input box is where you enter the control's identifier.

**Attributes check boxes**
The Attributes check boxes select the control's attributes.

**Scroll Bar radio buttons**
The Scroll Bar radio buttons determine the orientation of your scroll bars.

**Alignment radio buttons**
The Alignment radio buttons align your scroll bar.

## Scroll Bar radio buttons

The Scroll Bar radio buttons determine the orientation of your scroll bars. Scroll bars can be placed horizontally or vertically.

**Horizontal**
Turn the Horizontal radio button on to place the scroll bar horizontally.

**Vertical**
Turn the Vertical radio button on to place the scroll bar vertically.

## Alignment radio buttons

The Alignment radio buttons align your scroll bar. A rectangle defines the boundaries of the scroll bar. You can resize the rectangle, then use these alignment options to decide how the scroll bar is aligned within the rectangle:

**None**
Turn the None radio button on to place a scroll bar that fills the entire selection frame. If you resize the selection frame, you can distort the scroll bar. This option is the default.

**Top Left**
Turn the Top Left radio button on to place a horizontal and vertical scroll bar in the selection frame. The horizontal scroll bar is at the top of the selection frame and extends the full width of the frame. The vertical scroll bar is at the left side of the selection frame and extends the full height of the frame.

**Bottom Right**
Turn the Bottom Right radio button on to place a horizontal and vertical scroll bar in the selection frame. The horizontal scroll bar is at the bottom of the selection frame and extends the full width of the frame. The vertical scroll bar is at the right side of the selection frame and extends the full height of the frame.

## List Box Style dialog box

The List Box Style dialog box is where you define list box controls. To bring up this dialog box, double-click the list box control you want to modify.

**Caption input box**
The Caption input box is where you enter the caption you want displayed with the control.

**Caption radio buttons**
The Caption radio buttons choose how the caption is displayed.

**Control ID input box**
The Control ID input box is where you enter the control's identifier.

**Attributes check boxes**
The Attributes check boxes select the control's attributes.

**Scroll Bar check boxes**
The Scroll Bar check boxes put scroll bars in your control.

**Owner Drawing radio buttons**
The Owner Drawing radio buttons determine whether the list contained in the control should be drawn by the control or the application.

**List Box check boxes**
The List Box check boxes define the list box. You can turn one or more of these check boxes on.

## Owner Drawing radio buttons

The Owner Drawing options determine whether the list contained in the control should be drawn by the list box or the application.

**Not Owner Draw (list box controls)/No (combo box controls)**
Turn the Not Owner Draw or No radio button on if you want the list or combo box control to draw the list. This is the default.

**Fixed**
Turn the Fixed radio button on if you want the application to draw the list or combo box in response to WM_DRAWITEM messages. The application can also respond to WM_COMPAREITEM, WM_DELETEITEM, and WM_MEASUREITEM messages.

The list or combo box control sends the WM_MEASUREITEM message to the application only when the list box is initially drawn, which fixes the list box item height.

**Variable**
Turn the Variable radio button on if you want the application to draw the list or combo box in response to WM_DRAWITEM messages. The application can also respond to WM_COMPAREITEM, WM_DELETEITEM, and WM_MEASUREITEM messages.

The list or combo box control sends the WM_MEASUREITEM message to the application for each item in the list or combo box. Each item can vary in height.

**Has Strings**
Turn the Has Strings check box on if you turn on either the Fixed or Variable radio buttons and you want the combo box to store text for each list item with the LB_SETTEXT message. The combo box can retrieve list items from LB_GETTEXT. (For list boxes, the Has Strings check box is one of the List Box check boxes).

## List Box check boxes

The List Box check boxes define the list box. You can turn one or more of these check boxes on.

**Notify check box**
The Notify check box sends an input message to the parent window when the user clicks on an item in the list. This is the default.

**Sort check box**
The Sort check box sorts the items in the list alphabetically.

**Multiple Select check box**
The Multiple Select check box lets the user select more than one item at a time. The user can also toggle on and off individual items.

**Don't Redraw check box**
The Don't Redraw check box prevents the list box from being redrawn when it changes.

**Tab Stops check box**
The Tab Stops check box organizes the information in the list box in columns.

**Integral Height check box**
The Integral Height check box resizes the list box height at run time.

**Multi Column check box**
Turn the Multi Column check box on to create a list box in which the text wraps from column to column. The user scrolls the list box horizontally to display additional text. If you turn this check box on, the application must send the LB_SETCOLWIDTH message to set the width of all columns in pixels.

**Pass Keyboard Input check box**
The Pass Keyboard Input check box passes what the user types to the application.

**Extend Select check box**
The Extend Select check box lets the user select more than one item in the list.

**Has Strings**
Turn the Has Strings check box on if you turn on either the Fixed or Variable radio buttons (Owner Drawing radio buttons) and you want the list box to store text for each list item with the LB_SETTEXT message. The combo box can retrieve list items from LB_GETTEXT.

**Scroll Bar Always check box**
The Scroll Bar Always check box always puts a vertical scroll bar in the list box, but disables it when the list box doesn't contain enough items to scroll. This check box is displayed only when you run under Windows 3.1.

## Tab Stops check box

Turn the Tab Stops check box on to organize the information in the list box in columns. The default column width is 32 dialog units or 8 characters. Use Tab characters (\x09) to format the text.

(If you want to change the column width, the application should set its own tab stops using the LB_SETTABSTOPS message.)

## Integral Height check box

Turn the Integral Height check box on to cause the list box to resize its height at run time so that the client area is large enough to display items completely. This is the default.

If the Integral Height check box is turned on and the list box needs to be resized to show items completely, the list box always decreases its size. For example, if three items are completely displayed at run time but another item doesn't quite fit, the list box will decrease its size so that only the three items are displayed.

If you turn off the Integral Height check box, your application must set the height of the list box to account for any items listed in it.

## Combo Box Style dialog box

The Combo Box Style dialog box is where you define combo box controls. To bring up this dialog box, double-click the combo box control you want to modify.

**Caption input box**
The Caption input box is where you enter the caption you want displayed with the control.

**Caption radio buttons**
The Caption radio buttons choose how the caption is displayed.

**Control ID input box**
The Control ID input box is where you enter the control's identifier.

**Type radio buttons**
The Type radio buttons choose the type of combo box you're creating.

**Owner Drawing radio buttons**
The Owner Drawing options determine whether the list contained in the control should be drawn by the control or the application.

**Attributes check boxes**
The Attributes check boxes select the control's attributes.

## Type radio buttons

The Type radio buttons choose the type of combo box you're creating.

**Simple**
Turn the Simple radio button on if you want the drop-down list to always expand to display items in the list. The user can edit the items in the list. This is the default.

**Drop Down**
Turn the Drop Down radio button on if you want the combo box to consist of a single line of editable text when the dialog box is first displayed. The user can click the down arrow to expand the list, and edit all items in the list.

**Drop Down List**
Turn the Drop Down List radio button on if you want the combo box to work just like a drop down combo box, but contain a static list instead of an editable list. The user can select an item in the list, but can't change anything.

## Edit Text Style dialog box

The Edit Text Style dialog box is where you define edit text controls. To bring up this dialog box, double-click the edit text control you want to modify.

**Caption input box**
The Caption input box is where you enter the caption you want displayed with the control.

**Caption radio buttons**
The Caption radio buttons choose how the caption is displayed.

**Control ID input box**
The Control ID input box is where you enter the control's identifier.

**Attributes check boxes**
The Attributes check boxes select the control's attributes.

**Justification radio buttons**
The Justification radio buttons justify text.

**Scroll Bar check boxes**
The Scroll Bar check boxes put scroll bars in your control.

**Case radio buttons**
The Case radio buttons determine how edit text is displayed when it's typed.

**Line radio buttons**
The Line radio buttons choose how the user will type edit text.

**Text Conversion check boxes**
The Text Conversion check boxes choose how text is converted when it's typed.

**Automatic Scroll check boxes**
The Automatic Scroll check boxes choose how edit text will scroll.

**Windows 3.1 check boxes**
The Windows 3.1 check boxes assign attributes specific to 3.1. These check boxes are available only if you're running under Windows 3.1.

## Line radio buttons

The Line radio buttons choose how the user will type edit text.

**Single Line**
Turn the Single Line radio button on to restrict the text the user types to a single line. This is the default.

**Multiple Line**
Turn the Multiple Line radio button on to let the user type text on multiple lines. (You choose how multiple-line text is scrolled with the Automatic Scroll check boxes.)

## Case radio buttons

Use the Case radio buttons to choose how text is displayed when it's typed.

**Case Insensitive**
Turn the Case Insensitive radio button on to display text exactly as typed. This is the default.

**Upper Case**
Turn the Upper Case radio button on to display text in uppercase letters, regardless of how it's typed.

**Lower Case**
Turn the Lower Case radio button on to display text in lowercase letters, regardless of how it's typed.

## Text Conversion check boxes

Use the Text Conversion check boxes to choose how text is converted when it's typed.

**Password check box**
The Password check box suppresses the display of each letter as its typed and an asterisk appears in its place. This is helpful for keeping passwords secret.

**Convert OEM check box**
The Convert OEM check box converts text the user enters to the current OEM character set, then reconverts the text to ANSI. This option is useful in file input boxes because it ensures that any file name entered will be translatable into the OEM character set, which is used by the DOS file system.

**Keep Selection check box**
The Keep Selection check box highlights the selected text, even when the control doesn't have the keyboard focus. Choose Keep Selection when you want the user to see the highlighted text even when working with other controls.

## Windows 3.1 check boxes

The Windows 3.1 check boxes assign attributes specific to 3.1.

**Read Only check box**
The Read Only check box makes the edit text read-only. The user can't enter or modify text.

**Want Return check box**
The Want Return check box passes the Enter key into the edit buffer and does not perform the action for the default key. This attribute applies to multi-line text only.

## Automatic Scroll check boxes

The Automatic Scroll check boxes choose how edit text scrolls.

**Horizontal check box**
The Horizontal check box scrolls edit text horizontally. The text automatically scrolls ten characters to the right when the user types a character at the right edge of the edit text boundary. When the user presses Enter, the text scrolls back to the zero position.

**Vertical check box**
The Vertical check box scrolls edit text vertically. The text scrolls up a full page when the user presses Enter on the last line of the edit text control. For this option to have any effect, you must also turn the Multiple Line radio button on.

## Static Style dialog box

The Static Style dialog box is where you define static controls. To bring up this dialog box, double-click the static control you want to modify.

**Caption input box**
The Caption input box is where you enter the caption you want displayed with the control.

**Caption radio buttons**
The Caption radio buttons choose how the caption is displayed.

**Control ID input box**
The Control ID input box is where you enter the control's identifier.

**Attributes check boxes**
The Attributes check boxes select the control's attributes.

**No Character Underline check box**
The No Character Underline check box turns off character underlining. If your static control contains text, you can underline a character by preceding it with an ampersand (&). If you turn this check box on, underlining is disabled and ampersands are displayed as literal characters.

**Control Type radio buttons**
The Control Type radio buttons define what's displayed by the static control.

**Edit Icon button**
The Edit Icon Button starts the Paint editor, where you edit your icon. This button is enabled after you enter a caption for the icon in the Caption input box.

## Control Type radio buttons

Use the Control Type radio buttons to define what's displayed by the static control.

**Left Text**
Turn the Left Text radio button on to display text flush left within the control border. This is the default. If there are multiple lines of text, words that extend beyond the line wrap to the next line.

**Left Text-No Wrap**
Turn the Left Text-No Wrap radio button on to display text flush left within the control border. If there are multiple lines of text, this option automatically disables word wrapping. You must end lines with a carriage-return character.

**Centered Text**
Turn the Centered Text radio button on to display text centered within the control border. If there are multiple lines of text, words that extend beyond the line wrap to the next line.

**Right Text**
Turn the Right Text radio button on to display text flush right within the control border. If there are multiple lines of text, words that extend beyond the line wrap to the next line.

**Simple Text**
Turn the Simple Text radio button on to display a single line of flush-left text.

**White Rectangle**
Turn the White Rectangle radio button on to display a filled-in rectangle that uses the current window background color. The default color for the window background is white.

**Gray Rectangle**
Turn the Gray Rectangle radio button on to display a filled-in rectangle that's the color of the current screen background (desktop). The default color for the desktop is gray.

**Black Rectangle**
Turn the Black Rectangle radio button on to display a filled-in rectangle that's the same color as the current window frame color. The default color for window frames is black.

**White Frame**
Turn the White Frame radio button on to display an empty frame with a solid outline that's the color of the current window background. The default color for the window background is white.

**Gray Frame**
Turn the Gray Frame radio button on to display an empty frame with a solid outline that's the color of the current screen background (desktop). The default color for the desktop is gray.

**Black Frame**
Turn the Black Frame radio button on to display an empty frame that's the color of the current window frame color. The default color for window frames is black. When you add a frame to your dialog box, it might appear to be filled with the current background color. When you switch to Test mode, you see the frame as it's displayed at run time.

**Icon**
Turn the Icon radio button on to display an icon. Use the Edit Icon button to edit the icon.

## New Custom Control dialog box

The New Custom Control dialog box is where you add a new custom control to your dialog box. Make sure the control is already defined in a custom control library and that you've installed the library with Options|Install Control Library.

The only component of this dialog box is the Class drop-down box, where you choose the custom control you want to use. Resource Workshop displays a sample of the control in the middle of the dialog box.

The Borland custom controls are automatically displayed in this box.

**See Also**
Adding a custom control
Installing a custom control library
Working with controls

## Generic Control Style dialog box

The Generic Control Style dialog box is where you change a control's class. To display this dialog box, press Ctrl and double-click the control.

**Caption input box**
The Caption input box is where you enter the caption you want displayed with the control.

**Caption radio buttons**
The Caption radio buttons choose how the caption is displayed.

**Control ID input box**
The Control ID input box is where you enter the control's identifier.

**Class input box**
The Class input box is where you assign a custom class to your dialog box.

**Style input box**
The Style input box is where you enter the control's new style. Here are the styles you can choose from:

- Control Window Style Constants
- Dialog Window Style Constants
- Window Style Constants

**Info input box**
The Info input box is where you enter up to 255 bytes of data that can be passed to the WM_CREATE message. This data is meaningful within Resource Workshop only; if you enter data here, your dialog box won't be compatible with the Microsoft Resource Compiler.

**See Also**
Differences between Borland and Microsoft compilers

## Using Borland custom controls

The Borland Windows Custom Controls (BWCC) library contains a custom dialog class and a set of custom dialog controls (button, check boxes, group shading boxes, and so on). BWCC adds to the visual impact of your dialog boxes and their functionality.

**Using the Borland custom dialog class**

The custom dialog class, BORDLG, works on both a visual and functional level.

- It improves the appearance of your dialog window by painting the background with a brush that varies according to the target display device. For screens of VGA and higher resolution, the background is a fine grid of perpendicular white lines, giving the effect of "chiseled steel." For EGA and monochrome screens, the background is white.
- If optimizes the drawing of dialog boxes by calling the custom control drawing routines directly instead of waiting for Windows to paint the controls. This eliminates the typical sluggish drawing of dialog boxes.

To use the Borland custom dialog class,

1. Open the dialog resource you want to convert.
2. Double-click the title bar or outer edge of the dialog box to display the Window Style dialog box.
3. Enter "bordlg" as the Class and click OK.

**See Also**

Borland custom control tools
Customizing existing applications for BWCC

## Borland custom controls

These Borland custom controls are displayed in the Class drop-down box of the New Custom Control dialog box:

| Control | Description |
|---|---|
| 3-State Checkbox | A Borland-style check box that has three states - on, off, and "indeterminate," which is displayed as a checkerboard pattern. The application determines what is meant by "indeterminate." The application must call the **CheckDlgButton** function to send a BM_SETCHECK message to check the selected box. |
| Auto 3-State Checkbox | A check box that's identical to a Borland-style 3-state check box, except that BWCC and Windows combine to handle checking the selection box. |
| Auto Checkbox | A check box that's identical to a Borland-style check box, except that BWCC and Windows combine to handle checking the selection box. |
| Auto Radiobutton | A radio button that's identical to a Borland-style radio button, except that BWCC and Windows combine to handle highlighting the selected button and deselect the other buttons. |
| Bitmap | A bitmap. |
| Checkbox | A Borland-style check box. The application must call the **CheckDlgButton** function to send a BM_SETCHECK message to check the selected box. |
| Default Pushbutton | A push button that's identical to a Borland-style push button, but includes a bold border indicating that it's the default response if the user presses Enter. |
| Horizontal Bump | A convex horizontal line. |
| Horizontal Dip | A concave horizontal line. |
| Pushbutton | A Borland-style push button. When the user clicks the button, a BN_CLICKED message is sent to the parent window. |
| Radiobutton | A Borland-style radio button. The application must call the **CheckRadioButton** function to send a BM_SETCHECK message to highlight the selected button and deselect other buttons. |
| Raised Gray Group | A gray box that appears raised above the surface of the dialog box. |
| Recessed Gray Group | A gray box that appears recessed below the surface of the dialog box. |
| Static Text | A fixed text string used for labeling parts of the dialog box. |
| Vertical Bump | A convex vertical line. |
| Vertical Dip | A concave vertical line. |

**See Also**
Custom controls

## Borland custom control tools

The Borland custom control tools are displayed on the right side of the Tools palette. The tools are:

 Check Box

 Group Shade

 Horizontal Dip

 Push Button

 Radio Button

 Static Text

 Vertical Dip

Use these tools to put Borland custom controls in your dialog box.

**See Also**
Borland button and check box enhancements

## Borland button and check box enhancements

Borland push buttons, radio buttons, and check boxes have the following functional enhancements over standard Windows controls:

- An additional level of parent window notification and control over keyboard focus and tab movement. If you choose the Parent Notify option in the control's style dialog box, the control sends the appropriate message at run time:
- BBN_SETFOCUS indicates to the parent window that the push button, radio button, or check box has gained keyboard focus through an action other than a mouse click.
- BBN_SETFOCUSMOUSE indicates to the parent window that the push button, radio button, or check box has gained keyboard focus through a mouse click.
- BBN_GOTATAB indicates to the parent window that the user has pressed the Tab key while the push button, radio button, or check box has keyboard focus. The parent can intervene in the processing of the keystroke by returning a nonzero value.
- BBN_GOTABTAB indicates to the parent window that the user has pressed Shift-Tab (back-tab) while the push button, radio button, or check box has keyboard focus. The parent can intervene in the processing of the keystroke by returning a nonzero value.

- An owner-draw option that allows the parent window to draw the push button, radio button, or check box. Because your application handles drawing the control, it won't necessarily look like a Borland control, but it will have the standard behavior of that class of control.

## Push Button custom control tool

The Push Button custom control tool puts a Borland-style push button in your dialog box. A Borland-style push button can contain symbols with high visual impact, as well as an owner draw option. A Borland push button is larger than most standard Windows push buttons. Its class is BorBtn.

Once you place the Button control, double-click it to bring up the Borland Button Style dialog box. You use this dialog box to customize the control.

## Check Box custom control tool

The Check Box custom control tool puts a Borland-style check box in your dialog box. A Borland-style check box is raised and displays a check mark, rather than an "X." There is also an owner-draw option. Its class is BorCheck.

Once you place the Check Box control, double-click it to bring up the Borland Check Box Style dialog box. You use this dialog box to customize the control.

## Group Shade custom control tool

The Group Shade custom control tool puts a shaded rectangular box in your dialog box. The box groups other controls visually. It can appear recessed into the dialog box or raised above its surface. Its class is BorShade.

Once you place the Group Shade control, double-click it to bring up the Borland Shade Style dialog box. You use this dialog box to customize the control.

## Horizontal Dip custom control tool

The Horizontal Dip custom control tool puts a Borland-style horizontal dividing line in your dialog box. The line gives the impression of being etched into the surface of the dialog box. Its class is BorShade.

You can convert dips to bumps, which appear to be raised above the surface of the dialog box.

Once you've placed the Horizontal Dip control, double-click it to bring up the Borland Shade Style dialog box. You use this dialog box to customize the control.

## Radio Button custom control tool

The Radio Button custom control tool puts a Borland-style radio button in your dialog box. The radio button is diamond-shaped and appears raised from the surface of the dialog box.

When the button is clicked, a black diamond appears in its center and the button shading reverses, giving the impression that it has been pushed down. There is also an owner-draw option. Its class is BorRadio.

Once you place the Radio Button control, double-click it to bring up the Borland Radio Button Style dialog box. You use this dialog box to customize the control.

## Vertical Dip custom control tool

The Vertical Dip custom control tool puts a Borland-style vertical dividing line in your dialog box. The line gives the impression of being etched into the surface of the dialog box. Its class is BorShade.

You can convert dips to bumps, which appear to be raised above the surface of the dialog box.

Once you've placed the Vertical Dip control, double-click it to bring up the Borland Shade Style dialog box. You use this dialog box to customize the control.

## Static Text custom control tool

The Static Text custom control tool places a fixed text string in your dialog box. You use the string principally for labeling parts of the dialog box. Its class is BorStatic.

Once you place the Static Text Button control, double-click it to bring up the Borland Static Text Style dialog box. You use this dialog box to customize the control.

## Borland Button Style dialog box

The Borland Button Style dialog box is where you define a Borland style button or bitmap. To bring up the dialog box, place the Push Button control in your dialog box and double-click it.

When you first place a Borland push button in your dialog box, its text is Button and it takes the next available control ID. To change the button to one of the standard Borland buttons, change the control ID to one of the preset values in the following table:

| ID Name | ID Value | Type and Image |
|---|---|---|
| IDOK | 1 | OK - green check mark |
| IDCANCEL | 2 | Cancel - red X |
| IDABORT | 3 | Abort - panic button |
| IDRETRY | 4 | Retry - slot machine |
| IDIGNORE | 5 | Ignore - 55 mph speed-limit sign |
| IDYES | 6 | Yes - green check mark |
| IDNO | 7 | No - red circle and slash |
| IDHELP | 998 | Help - blue question mark |

**Caption input box**
The Caption input box is where you enter the caption you want displayed with the control.

**Control ID input box**
The Control ID input box is where you enter the control's identifier number.

**Attributes check boxes**
The Attributes check boxes select the control's attributes.

**Button Type radio buttons**
The Button Type radio buttons define the button type.

## Attributes check boxes

The Attributes check boxes select the control's attributes. Not all attributes apply to all controls. The Attributes check boxes are described in alphabetical order; scroll through the list to find the option you want.

**Border check box**
Turn the Border check box on to draw a border around the control. The border is a dark line. For static text, the border is a standard Windows border that uses the current color for the window frame.

**Caption Above check box**
Turn on the Caption Above check box to display the control's caption at the top of the control. This attribute only applies to the Group Shade custom control.

**Control Color check box**
The Group Shade check box sends the equivalent of a WN_CTLCOLOR message to its parent to set the color of the Group Shade's background and caption text. This is only available for the Group Shade custom control tool.

**Disabled check box**
Turn the Disabled check box on to disable the control. The disabled control is gray. This prevents the control from responding to user input.

**Group check box**
Turn the Group check box on to indicate the first control within a group. The user can then press the arrow keys to access all controls in the group. You can also use the Set Groups tool to define a group of controls.

**No Underline check box**
The No Underline check box forces an ampersand (&) to appear as a literal character, instead of underlining the next character in the option. This is available only for the Static Text, Group Shade, Horizontal Dip, and Vertical Dip custom control tools.

**Owner Draw check box**
The Owner Draw check box allows the parent window to draw the push button, radio button, or check box.

**Parent Notify check box**
The Parent Notify check box sends messages to the parent window.

**Tab Stop check box**
Turn the Tab Stop check box on if you want the user to press Tab to access this control. You can also use the Tab Set tool to specify which controls are tab stops.

**Visible check box**
The Visible check box determines whether the control is visible when the dialog box is first displayed. By default, the option is checked (WS_VISIBLE). If the option is not checked (NOT WS_VISIBLE), the control does not appear. The application can call the **ShowWindow** function at run time to make the control appear.

**See Also**
Grouping related controls
Specifying which controls are tab stops

## Parent Notify check box

Turn the Parent Notify check box on to cause the push button, radio button, or check box to generate additional notification messages at run time.

| Message | Description |
|---|---|
| BBN_SETFOCUS | Indicates to the parent window that the button or check box has gained keyboard focus through an action other than a mouse click. |
| BBN_SETFOCUSMOUSE | Indicates to the parent window that the button or check box has gained keyboard focus through a mouse click. |
| BBN_GOTATAB | Indicates to the parent window that the user has pressed the Tab key while the button or check box has keyboard focus. The parent window can then intervene in the processing of the keystroke by returning a nonzero value. |
| BBN_GOTABTAB | Indicates to the parent window that the user has pressed Shift+Tab while the button or check box has keyboard focus. The parent can intervene in the processing of the keystroke by returning a nonzero value. |

## Button Type radio buttons

The Button Type radio buttons define the type of push button.

**Pushbutton**
Turn the Pushbutton radio button on to place a Borland-style push button in your dialog box. When the user clicks the button, a BN_CLICKED message is sent to the parent window. This is the default setting.

**Defpushbutton**
Turn the Defpushbutton radio button on to create a button identical to a push button that also includes a bold border. The border indicates that it's the default response when the user presses Enter.

**Bitmap**
Turn the Bitmap radio button on to insert a bitmap image (based on its control ID) into the button. To read in a bitmap:

1. Use the Push Button tool to add the generic BWCC button to your dialog box. Note its control ID.

2. Switch to the Paint editor and create a bitmap image.

3. In the Paint editor, choose Resource|Rename to display the Rename Resource dialog box, then do either of the following:

- In the New Name text box, enter an integer value that equals the control ID of the button plus the appropriate offset from the following table.
- Rename the bitmap and assign it an identifier whose value equals the control ID of the button plus the appropriate offset from the following table.

| Button State | Offset for VGA | Offset for EGA |
|---|---|---|
| standard | 1000 | 2000 |
| pressed | 3000 | 4000 |
| keyboard focus | 5000 | 6000 |

4. Close the Paint editor.

5. Return to the Dialog editor. If the bitmap doesn't immediately appear in the BWCC button, resize it.

## Borland Check Box Style dialog box

The Borland Check Box Style dialog box is where you define a Borland-style check box. To bring up the dialog box, place the Check Box control in your dialog box and double-click it.

**Caption input box**
The Caption input box is where you enter the caption you want displayed with the control.

**Control ID input box**
The Control ID input box is where you enter the control's identifier number.

**Attributes check boxes**
The Attributes check boxes select the control's attributes.

**Button Type radio buttons**
The Button Type radio buttons define the check box type.

## Button Type radio buttons

The Button Type radio buttons define the type of check box.

**Checkbox**
Turn the Checkbox radio button on to place a Borland-style check box in your dialog box. You can put text to the left or right of the box. The application must call the **CheckDlgButton** function to send a BM_SETCHECK message to check the selected box.

**Auto Checkbox**
Turn the Auto Checkbox radio button on to create a check box that's identical to a Borland-style check box, except that BWCC and Windows combine to handle checking the selection box. This is the default option.

**3-state**
Turn on the 3-state radio button to create a Borland-style check box that has three states - on, off, and "indeterminate," which is displayed as a checkerboard pattern. The application determines what is meant by "indeterminate." The application must call the **CheckDlgButton** function to send a BM_SETCHECK message to check the selected box.

**Auto 3-state**
Turn the Auto 3-state radio button on to create a check box that's identical to a Borland-style 3-state check box, except that BWCC and Windows combine to handle checking the selection box. This is the default option.

## Borland Radio Button Style dialog box

The Borland Radio Button Style dialog box is where you define a Borland-style radio button. To bring up the dialog box, place the Radio Button control in your dialog box and double-click it.

**Caption input box**
The Caption input box is where you enter the caption you want displayed with the control.

**Control ID input box**
The Control ID input box is where you enter the control's identifier number.

**Attributes check boxes**
The Attributes check boxes select the control's attributes.

**Button Type radio buttons**
The Button Type radio buttons define the radio button type.

## Button Type radio buttons

The Button Type radio buttons define the type of radio button.

**Radiobutton**
Choose Radiobutton to place a Borland-style radio button in your dialog box. The application must call the **CheckRadioButton** function to send a BM_SETCHECK message to highlight the selected button and deselect other buttons.

**Auto Radiobutton**
Turn on Auto Radiobutton to create a radio button that's identical to a Borland-style radio button, except that BWCC and Windows combine to highlight the selected button and deselect the other buttons.

## Borland Shade Style dialog box

The Borland Shade Style dialog box is where you define a Borland-style group box or horizontal or vertical line. To bring up the dialog box, place the Group Shade, Horizontal Dip, or Vertical Dip control in your dialog box and double-click it.

**Caption input box**
The Caption input box is where you enter the caption you want displayed with the control.

**Control ID input box**
The Control ID input box is where you enter the control's identifier number.

**Shade Type radio buttons**
The Shade Type radio buttons define the type of shade control.

**Attributes check boxes**
The Attributes check boxes select the control's attributes.

**Alignment for Caption radio buttons**
The Alignment for Caption radio buttons choose the alignment of the Group Shade's caption.

**Left**
Turn on Left to left justify the caption in the Group Shade.

**Centered**
Turn on Centered to center the caption in the Group Shade.

**Right**
Turn on Right to right justify the caption in the Group Shade.

**See Also**
Grouping related controls

## Shade Type radio buttons

The Shade Type radio buttons define the type of shade control. A dip is intended to act as a separator in the dialog box background or in a Raised Shade box. A bump is intended as a separator in a Group Shade box.

**Group Shade**
The Group Shade radio button creates a gray box that appears recessed below the surface of the dialog box. The controls you place in the box are members of the same group.

**Raised Shade**
The Raised Shade radio button creates a gray box that appears raised above the surface of the dialog box. The controls you place in the box are members of the same group.

**Horizontal Dip**
The Horizontal Dip radio button creates a concave horizontal line. This is the default horizontal line.

**Vertical Dip**
The Vertical Dip radio button creates a concave vertical line. This is the default vertical line.

**Horizontal Bump**
The Horizontal Bump radio button creates a convex horizontal line.

**Vertical Bump**
The Vertical Bump radio button creates a convex vertical line.

## Borland Static Text Style dialog box

The Borland Static Text Style dialog box is where you define Borland-style text. To bring up the dialog box, place the Static Text custom control in your dialog box and double-click it.

**Caption input box**
The Caption input box is where you enter the caption you want displayed with the control.

**Control ID input box**
The Control ID input box is where you enter the control's identifier number.

**Attributes check boxes**
The Attributes check boxes select the control's attributes.

**Control Type radio buttons**
The Control Type radio buttons define the type of static text.

## Control Type radio buttons

The Control Type radio buttons define the static text. All BWCC static text, including Simple Text, use the standard BWCC gray background.

**Left Text**
Turn on the Left Text radio button to display text flush left within the control border. This is the default. If there are multiple lines of text, words that extend beyond the line wrap to the next line.

**Left Text-No Wrap**
Turn on the Left Text-No Wrap radio button to display text flush left within the control border. If there are multiple lines of text, this option automatically disables word wrapping. You must end lines with a carriage-return character.

**Centered Text**
Turn on the Centered Text radio button to display text centered within the control border. If there are multiple lines of text, words that extend beyond the line wrap to the next line.

**Right Text**
Turn on the Right Text radio button to display text flush right within the control border. If there are multiple lines of text, words that extend beyond the line wrap to the next line.

**Simple Text**
Turn on the Simple Text radio button to display a single line of flush-left text.

## Customizing existing applications for BWCC

Resource Workshop allows you to customize existing Windows applications with Borland-style custom controls. There are two steps to this process:

1. Modify your WIN.INI file to load the Borland Windows Custom Control (BWCC) library each time you start Windows.

2. Edit the application in Resource Workshop to change user interface features such as dialog boxes, menus, icons, and so on.

**Loading BWCC to enable Borland custom controls**
The BWCC library, which provides support for Borland-style custom controls, must be loaded before an application can use BWCC's features.

Edit the WIN.INI file (located in the Windows main directory) so that Windows loads the file LOADBWCC.EXE into memory at start up. (The installation program should have put LOADBWCC.EXE in the language compiler directory and added this directory to your PATH.)

Add LOADBWCC.EXE to the beginning of the list of files that appear after the "LOAD=" statement. LOADBWCC.EXE must appear first in the statement to ensure that BWCC is loaded into memory before any modified applications are executed.

## Accelerators

An accelerator is a key combination a user presses to perform a task in an application. It substitutes for a menu command and, like a menu command, creates a WM_COMMAND or WM_SYSCOMMAND message that tells the application what to do next. Usually, you create accelerators to duplicate commands on pop-up menus.

You store accelerator definitions in an accelerator table (the accelerator resource). Each entry in the table is an accelerator that defines the key combination a user must press and the command it produces. You can create multiple accelerator tables for different parts of a menu.

Resource Workshop provides an Accelerator editor that makes it easy to create and edit accelerators for your application. Working with accelerators involves four basic steps:

1. Start the Accelerator editor.

2. Start the Menu editor.

   Having the Menu editor open at the same time as the Accelerator editor displays the menu associated with the accelerators you're working on and makes it easier for you to enter identifiers correctly.

3. Create or edit an accelerator table and test it for duplicate keys.

4. Save the accelerator table.

**See Also**
Creating a new accelerator table
Editing an existing accelerator table
Saving an accelerator table
Testing an accelerator
Using the Accelerator editor
Using the Menu editor

## Using the Accelerator editor

Use the Accelerator editor to customize accelerator tables. The screen is divided into two panes that provide two different views of the accelerator table:

- The Outline pane shows the accelerators defined in the table.
- The Attribute pane is where you customize the selected accelerator.

Use the mouse or F6 to move from pane to pane.

**See Also**
Customizing an accelerator table

## Outline pane

The Outline pane, on the right of the Accelerator editor screen, displays the accelerators defined in the table.

The top line in the pane is the name of the accelerator table. The lines below it are accelerator entries, which have two parts:

- The first part identifies the key that is used as an accelerator. It is either an ASCII key or a virtual key.
- The second part is the item ID of the command to which the accelerator is tied. This ID is either an integer or the name of an identifier.

You can move around in this pane with the mouse or the arrow keys.

**See Also**
ASCII key
Virtual key

## Attribute pane

The Attribute pane is on the left side of the Accelerator editor screen. You use the Attribute pane to change the accelerator currently highlighted in the Outline pane.

Here are the components of the pane:

**Command input box**
The Command input box is where you enter the item ID for the command the accelerator is to execute.

**Key input box**
The Key input box is where you enter the accelerator key.

**Key Type radio buttons**
The Key Type radio buttons select the type of accelerator.

**Modifier check boxes**
The Modifier check boxes define key combinations.

## ASCII key

An ASCII key is one that can be displayed. It appears in the Key input box and follows standard conventions for representing ASCII characters.

All ASCII keys must be surrounded by quotation marks. A caret (^) indicates that the key is combined with the Ctrl key. Turn on the Alt check box (one of the Modifier check boxes) if the accelerator is combined with the Alt key.

Typically, you don't use single ASCII characters as accelerator keys; instead you combine them with Alt or Ctrl, such as Alt+R or Ctrl+L.

**See Also**
Virtual key

## Virtual key

A virtual key is typically a function key, an arrow key, or an editing key such as Home or End. Windows has predefined characters for virtual keys. These identifiers all start with VK_ and are defined in WINDOWS.H.

You don't need to look up virtual key identifiers if you use Key Value mode to insert the key.

Virtual keys have no provision for Ctrl, Shift, and Alt combinations. If you use these keys, you need to turn on the appropriate Modifier check box in the Accelerator editor dialog box.

**See Also**
ASCII key

## Creating a new accelerator table

You can create a new accelerator table in a new project or an existing one. To start the Accelerator editor and create a new accelerator table,

1. Create a new project or open an existing one.
2. Choose Resource|New to display the New Resource dialog box.
3. In the Resource Type list box, double-click ACCELERATOR.
4. Click OK to place the accelerator table in the current project file or choose another project file.

Resource Workshop displays the Accelerator editor with an accelerator table template you can customize.

**See Also**
Creating a new project
Customizing an accelerator table
Opening an existing project
Using the Accelerator editor

## Editing an existing accelerator table

To edit an existing accelerator table,

1. Open the project in which the accelerator resource is stored.

2. Double-click the accelerator resource name you want to edit, or highlight it and choose Resource| Edit.

Resource Workshop displays the Accelerator editor with the selected accelerator, ready to customize. You can also edit the accelerator table's resource script.

**See Also**
Customizing an accelerator table
Editing an accelerator table as text
Opening an existing project
Using the Accelerator editor

## Customizing an accelerator table

Once you've loaded an accelerator table into the Accelerator editor, you're ready to customize it.

**Adding an accelerator key**
To add a new accelerator to the accelerator table, press Ins or choose Accelerator|New Item.

The new key appears in the Outline pane below the currently selected line. It has a default value of 0 (zero) (displayed in the Key input box) and a unique integer value (displayed in the Command input box.) Use the Attribute pane to change the accelerator.

**Selecting an accelerator key**
To select an accelerator key, you can:

- Press Ctrl+Up Arrow or Ctrl+Down Arrow to highlight the accelerator in the Outline pane and automatically switch editing focus to the Attribute pane.
- Click the mouse on the accelerator in the Outline pane and then press F6 to switch editing focus to the Attribute pane.
- If you're already in the Outline pane, use the arrow keys to select the accelerator and then press F6 to switch editing focus to the Attribute pane.

**Editing an accelerator key**
To edit an accelerator, use the mouse or press Ctrl+Up Arrow or Ctrl+Down Arrow to highlight the accelerator in the Outline pane. The Accelerator editor displays the accelerator's current values in the Attribute pane. Enter the new key combination in the Key input box.

**Copying accelerators**
To move or copy an accelerator, select it in the Outline pane and then choose Edit|Cut or Edit|Copy. In the Outline pane, select the accelerator you want the moved or copied accelerator to follow. Choose Edit|Paste.

**Deleting accelerators**
Select the accelerator you want to delete in the Outline pane. Press Del or choose Edit|Cut or Edit|Delete.

**Undoing and redoing changes**
The Accelerator editor lets you undo and redo changes. Choose Edit|Undo or Edit|Redo, or use the accelerators Alt+Backspace (Undo) and Shift+Alt+Backspace (Redo).

**See Also**
Edit menu
Attribute pane

## Testing an accelerator

To avoid using the same key combination in more than one accelerator, you can debug an accelerator table by searching for duplicate key combinations:

1. With the accelerator table open, choose Accelerator|Check Dup Keys.

2. If two accelerators use the same key combination, the Accelerator editor displays a message and highlights the second accelerator. Make changes and continue debugging your accelerator table until you see the message "No duplicate key values found."

## Saving an accelerator table

It's a good idea to save your changes as you go along, rather than waiting for Resource Workshop to prompt you when you close the project.

There are two primary options you can use to save changes you've made to an accelerator table: You can save the entire project or save the accelerator table in a resource script file.

**See Also**
Saving the entire project
Saving the accelerator table in a resource script file

## Saving an accelerator table in a resource script file

Choose Resource|Save Resource As to save the accelerator table in its own file. This command displays the Save Resource As dialog box where you enter the file name or the accelerator table resource. Then choose File|Add to Project to add the accelerator table resource file to your project.

Usually, you save an accelerator table with a project. You'd probably only save it in its own file if you create a separate project that contains just menus and accelerators. You can then link this to a number of projects that use the same basic menu structure.

## Editing an accelerator table as text

In addition to creating and modifying accelerators directly with the Accelerator editor, you can work with the accelerator's resource script.

To work with the resource script for an accelerator table, select the accelerator table from the Project window. Then choose Resource|Edit As Text. Resource Workshop brings up the source script for the resource in its internal text editor.

Don't spend any time inserting comments in your resource script or formatting the text, because the Resource Workshop incremental compiler does its own formatting and discards all comments.

**See Also**
Resource script language
Using a text editor

## Command input box

The Command input box is where you enter the item ID for the command the accelerator is to execute. You can enter either a resource ID or an identifier. This value must match the value in the associated menu resource. (It's displayed in the Menu editor's Item ID input box.)

If you enter an identifier name and see the message "Create a new identifier: <identifier name>?", you probably typed the name incorrectly. Click No and check the spelling.

If you deliberately enter a new identifier because you intend to add the associated menu item to the menu, click Yes. You see the New Identifier dialog box, where you add the new identifier. Be sure to enter a value that's different from the other item IDs in the associated menu.

If you don't see the "Create a new identifier" message, the identifier already exists. Enter a unique identifier before you continue.

**See Also**
Adding accelerator text to menus

## Key input box

The Key input box is where you enter the accelerator key (the key combination you want users to press to invoke the accelerator). Your accelerator should be consistent with accelerators in other Windows applications.

You can enter the key in manual mode or Key Value mode. If you're in manual mode, select the type of key with the Key Type radio buttons.

You can also enter the key combination with the Accelerator|Key Value menu command.

**See Also**
Key Value mode
Manual mode

## Manual mode

If you position the mouse on the Key Input box, or press Alt+Esc to exit from Key Value mode, you're in manual mode.

In manual mode, you specify all the information that defines the accelerator key. You must decide if the key is an ASCII key or a virtual key and enter it in the correct format. You're required to turn on the appropriate Key Type radio button.

To use Alt in combination with an ASCII key, check the Alt check box. To use Alt, Shift, and Ctrl with a virtual key, check one or more of the Modifiers check boxes.

**See Also**
ASCII key
Key Value mode
Virtual key

## Key Value mode

To enter Key Value mode, press Tab to move to the Key input box, or choose Accelerator|Key Value.

When you're in Key Value mode, the Accelerator editor does most of the work for you. Any key or key combination you press gets entered into the Key input box as the accelerator. The Accelerator editor decides if it's an ASCII key or a virtual key and turns on the correct Key Type radio button and Modifiers check boxes.

**See Also**
ASCII key
Manual mode
Virtual key

## Key Type radio buttons

The Key Type radio buttons indicate whether your accelerator uses ASCII or virtual keys. If you're in Key Value mode, the Accelerator editor sets these radio buttons automatically.

**ASCII**
Turn the ASCII radio button on if your accelerator uses an ASCII key.

**Virtual Key**
Turn the Virtual radio button on if your accelerator uses a virtual key.

**See Also**
ASCII key
Key Value mode
Virtual key

## Modifier check boxes

The Modifier check boxes define the keys the user holds down to activate the accelerator.

**Alt check box**
Turn the Alt check box on if the accelerator key combination includes the Alt key (for example, Alt+W).

**Shift check box**
Turn the Shift check box on if the accelerator combination includes the Shift key (for example, Shift+F1).

**Control check box**
Turn the Control check box on if the accelerator key combination includes the Ctrl key (for example, Ctrl+W).

**Invert Menu Item check box**
Turn the Invert Menu Item check box on to disable the flash feature. The flash feature is a built-in Windows function that flashes a menu-bar command when the user presses the accelerator key associated with that menu-bar command. The feature lets the user know which menu the accelerator key is on. Invert Menu Item is on by default when you create an accelerator.

## String tables

A string table holds error messages, prompts, or any other text strings your application needs to display. You can store multiple string tables in your project file or in resource files.

Typically, you define a separate string table for each logical grouping of your program. Defining strings of text as a separate resource makes it easy to edit the text without changing your source code.

When you work with string tables, you perform four tasks:

1. Start the String editor.
2. Create and edit string tables.
3. Save the string table.
4. Compile the resource into the executable file and test the string table.

**See Also**
Creating a new string table
Editing an existing string table
Saving a string table
Testing a string table
Using the String editor

## Creating a new string table

To create a new string table,

1. Open the project you want to add the string table to.
2. Choose Resource|New to display the New Resource dialog box.
3. In the Resource Type list box, double-click STRINGTABLE.
4. Click OK to place the bitmap in the current project file or choose another project file.

Resource Workshop opens the String editor and places a reference to the new string table in your Project window. You're ready to customize the string table.

**See Also**
Creating a new project
Customizing a string table
Opening an existing project
Using the String editor

## Editing an existing string table

To edit an existing string table, open the project in which the string table resource is stored. You can either

- double-click the string table name in the Project window, or
- highlight the string table name and choose Resource|Edit.

Resource Workshop opens the String editor. The string table you've chosen is loaded into the editor, so you can customize it. You can also edit the string table's resource script.

**See Also**
Customizing a string table
Editing a string table as text
Opening an existing project
Using the String editor

## Using the String editor

You use the String editor to customize string tables. The String editor screen is divided into three areas that represent the required parts of a string table entry:

- The ID source input box is where you enter the string's resource ID or identifier.
- The ID value display box shows the string's resource ID.
- The String input box is where you enter the text of the string.

**See Also**

Customizing a string table
Windows and string resources

## Windows and string resources

Each string in a string table must have a unique resource ID. Windows groups strings into segments that contain 16 strings each. Strings with IDs from 0 to 15 make up the first segment, 16 to 31 are in the second segment, and so on. When you compile resources, the strings are added to the executable file in segments.

At run time, the strings are loaded into memory in segments. Windows loads an entire string segment into memory each time your application requires a particular string. If you plan how you assign string IDs, you can reduce the amount of memory your application requires.

To use memory efficiently, you should group your strings logically. For example, if one part of your application requires 5 strings, number them from 0 to 4. If a second part of your application requires 9 strings, put them into the second segment by numbering the strings 16 to 24. Then Windows can load related strings all at once, without having to load other strings that aren't needed.

## Customizing a string table

Once you've loaded a string table into the String editor, you're ready to customize it. You customize a string table by:

- editing an existing string
- entering a new string

## Editing an existing string

The String editor makes it easy to change individual strings. To select a string with your mouse, click the string you want to edit. If you use a keyboard, press Tab, Up Arrow, or Down Arrow to move through the table. Place your cursor on the string you want to edit.

**Changing a string**
You can erase the values in the ID Source and String input boxes for any string, and then type new values. You can't directly change what's displayed in the ID Value display box, but the String editor updates it based on what you type in the ID Source input box.

**Restoring changed string values**
Choose Stringtable|Restore Item to restore changed values. You can also choose Edit|Undo.

**Deleting a string**
To delete a string, highlight the string and then choose either Edit|Cut or Stringtable|Delete Stringtable Item.

## Entering a new string

To enter a string in a new string table,

- If the string is new, start with step 3 below.
- If you're adding a string to a string table, start with step 1.

1. Select the string above the line where you want to add the new string.

2. Press Ins or choose Stringtable|New Stringtable Item.

3. Enter an ID Source for the string in the ID Source input box. Based on the ID Source that you type, the String editor displays the appropriate integer value in the ID Value display box.

4. Either press Tab or click the String input box, then type the text string.

5. Press Enter to accept the new value, or Ins to accept the value and insert a new one.

## Editing a string table as text

If you want to work with the resource script of a string table, select the string table in the Project window, then choose Resource|Edit As Text.

Resource Workshop brings up the source script for the resource in its internal text editor. You can change the string or the identifier.

**See Also**
Changing a string in the resource script
Changing an identifier in the resource script
Resource script language
Using a text editor

## Changing a string in the resource script

To change a string in the resource script,

1. Find the string you want to edit and make the necessary changes. Change only the text that appears between the quotation marks.

2. Double-click the editor's Control-menu box.

3. Click Yes in response to the prompt "Resource has changed. Compile?".

Resource Workshop compiles your changes and saves them. If there's a syntax error, you're put back in the text editor so you can correct it.

**See Also**
Resource script language

## Changing an identifier in the resource script

If you change the string's identifier in the text editor, you must have already added the identifier to the appropriate identifier file. If you haven't, you see the error "Expecting unsigned short integer." This error indicates that the compiler tried to interpret the identifier name, but couldn't find an identifier for it.

If the compiler can't find an identifier because one doesn't exist, you can add a new one:

1. Make the Project window the active window.
2. Choose Resource|Identifiers. You see the Identifiers dialog box.
3. Choose the New button.
4. Enter the new identifier name and value in the New Identifier dialog box.
5. Choose the file you want to store the identifier in and click OK.
6. Select the String table name in the Project window and choose Resource|Edit As Text.
7. Double-click the editor's Control-menu box.
8. Click Yes in response to the prompt "Resource has changed. Compile?".

## Saving a string table

It's a good idea to save your changes as you go along, rather than waiting for Resource Workshop to prompt you when you close the project.

Resource Workshop automatically places a STRINGTABLE entry in your resource script in the Project window when you create a new string table resource. To save your string table, you need to save the entire project. Choose File|Save Project.

## Testing a string table

To test your string table, you need to compile the resource script file into an executable file. Then you can run the executable file to see if your strings appear as you think they should.

## ID Source input box

The ID Source input box holds the string's ID. You can enter a resource ID or an identifier ID source. If you enter a resource ID, the String editor automatically enters it in the ID value field.

If you type an identifier, Resource Workshop checks to see if it already exists. If it does, the String editor inserts the identifier's integer value in the ID Value display box when you tab to the String input field.

If the Identifier doesn't exist, you see the New Identifier dialog box, where you add the new identifier.

## ID Value display box

The ID Value display box holds the appropriate integer value for the string's ID. You can't enter a value into the box or change the value in it. The String editor enters the value based on the ID Source that you type.

## String input box

The String input box holds the text of the string. Each string can be a maximum of 255 characters. It can contain any C-type escape sequences, including:

- \n (newline)
- \t (tab)
- \r (carriage return)
- \\ (backslash)
- \" (double quote)

When the Resource Workshop compiler encounters a C-type escape sequence in a string entry, it produces the corresponding ASCII hexadecimal value in the object code. It's up to your program to interpret the value correctly.

## Using the Paint editor

You use the Paint editor to create and edit any bitmapped resource, including bitmaps, cursors, fonts, and icons.

**Paint editor options**
Use the Options|Editor Options command to set the Paint editor options.

This command displays the Set Paint Editor Options dialog box, where you

- specify that the images in both window panes be updated when you draw
- display a grid on zoomed images
- save a custom color palette

**Paint editor features**
Although most of the Paint editor's features are the same for the bitmapped resource types, there are some features that are unique to each resource type. For example, the name of the third menu from the right is the same as the resource type: Icon for an icon resource, Font for a font resource, Bitmap for a bitmap resource, and Cursor for a cursor resource.

The Paint editor features whose functionality is generally the same for all types of bitmapped resources are:
Colors palette
Status line and bar
Tools palette
Window panes

## Colors palette

Bitmapped images drawn with the Paint editor are created on a grid of pixels. You create the image by setting each pixel to a foreground or background color. Strictly speaking, there's no difference between the foreground and background color. There is, however, a difference in how you choose foreground and background colors and in how they are created when you edit the bitmapped image.

The Paint editor's 16-color Colors palette makes it easy to choose foreground and background colors. You can work with a Colors palette even if your image is black and white.

You use the Colors palette to choose:

- a foreground color
- a background color
- transparent and inverted areas (for icons and cursors only)

You can move the palette anywhere you want it. You can also hide it and show it with the menu commands or the palette's control menu. If you're creating a bitmap or icon, you can choose the number of colors you want to include in your resource.

**See Also**

Choosing the number of colors for a resource
Customizing colors
Hide and showing the Colors palette

## Foreground color

The foreground color is the color you select and draw with the left mouse button. It's typically one of the colors you use to create the features of your resource, such as lines, boxes, shading, and so on. The foreground color is indicated by FG on the Colors palette. (If you select the same color for both foreground and background, you see FB on the Colors palette.)

To use a foreground color,

1. Click the left mouse button on the color you want in the Colors palette. You see the letters FG appear on that color (unless it's already selected as a background color, in which case you see FB).

2. Choose a tool that draws or paints. Click or drag with the left mouse button to draw or paint with the foreground color.

The Eraser tool operates in the opposite fashion from the drawing tools. Dragging it with the left mouse button produces the background color, and dragging it with the right mouse button produces the foreground color.

## Background color

The background color is the color you select and draw with the right mouse button and is usually the color that appears to underlie your drawing. It's also the color that's left behind when you select an area and delete or move it. The background color is indicated by BG on the Colors palette. (If you select the same color for both foreground and background, you see FB on the Colors palette.)

To use a background color,

1. Click the right mouse button on the color you want in the Colors palette. You see the letters BG appear on that color (unless it's already selected as a foreground color, in which case you see FB).

2. Choose a tool that draws or paints. Click or drag with the right mouse button to draw or paint with the background color.

The Eraser tool operates in the opposite fashion from the drawing tools. Dragging it with the left mouse button produces the background color, and dragging it with the right mouse button produces the foreground color.

## Transparent and inverted areas

Transparent and inverted color areas are unique to icon and cursor resources.

- A transparent area allows the desktop color behind the icon or cursor to show through.
- An inverted area reverses the desktop color at run time.

The designated transparent and inverted colors do not appear in your icon or cursor at run time. Instead, they are replaced by the desktop color or its inverse. The colors that you set as Transparent and Inverted should be colors that you won't use in your icon or cursor.

The default transparent color is the current desktop color set in the Windows Control Panel's color palette. If the desktop uses a dithered color, the default transparent color is the nearest solid color that Resource Workshop can provide. (If you have a 256-color device, this restriction does not apply; the default transparent color will always match the desktop color.)

You can change the transparent color to something other than the desktop color, but it will always revert to the desktop color each time you start Resource Workshop. Nevertheless, regions that you designate as transparent - using the default color or a color you assign - remain transparent and take on the current transparent color.

**Note:** You can change colors so that you have transparent and nontransparent regions that use the same color.

To change the colors the Paint editor displays for transparent and inverted areas,

1. Double-click the bar under Transparent or Inverted in the Colors palette.
2. Select either Transparent or Inverted as the foreground or background color.
3. If you're working with icons, choose Icon|Edit Foreground Color or Icon|Edit Background Color.
4. If you're working with cursors, choose Cursor|Set Transparent Color.

Resource Workshop displays the Set Transparent Color dialog box, where you can change the color.

## Hiding and showing the Colors palette

If you want to hide the Colors palette, you can close it by double-clicking the control menu icon in the upper left corner of the palette.

You can also use the Hide Palette and Show Palette commands. The menu where you find these commands depends on the type of resource you're currently editing. For example, when you edit an icon, you find the Show Palette and Hide Palette commands in the Icon menu.

Only one of these commands appears at a time. If the Colors palette is visible, you only see the Hide Palette command. The Show Palette command appears only if the Colors palette is hidden.

## Choosing the number of colors for a resource

When you create a bitmap or an icon, Resource Workshop displays a dialog box where you choose how many colors you want in your resource.

For bitmaps, this dialog box is the New Bitmap Attributes dialog box. For icons, it's the New Icon Image dialog box.

While you're editing a bitmap or icon, you can change the number of colors in the image using the Size And Attributes command. Depending on the type of resource you edit, you find the Size And Attributes command in the Bitmap menu or the Icon menu.

You can include up to 256 colors in your bitmap or icon. The number of colors you can use (and see in the Colors palette) depends on the type of display driver you're using with Windows. The standard Windows VGA driver supports a fixed set of only 16 colors.

## Customizing colors

When you're editing a color bitmap or icon, you can modify the Colors palette to include any colors supported by your display driver.

It won't make sense to do this if the Colors palette already includes all the colors supported by your computer. But if your display driver is capable of displaying 256 colors and you're working with a 16-color image, you can include any of the 256 colors in the 16-color Colors palette.

When you're editing a cursor or icon, you can change the colors used to display transparent or inverted areas of the image.

**See Also**
Modifying the Colors Palette

## Modifying the Colors palette

If you're editing a color bitmap or icon, you can edit a color in the Colors palette,

To edit a color, do one of the following:

- Double-click it.
- Use the Edit Foreground Color and Edit Background Color commands. These commands are in the Bitmap and the Icon menu.

Resource Workshop displays the Edit Color dialog box, where you can specify a new color.

## Edit Color dialog box

The Edit Color dialog box is where you customize colors. Display this dialog box by double-clicking a color in the Colors palette, or with the Edit Foreground Color and Edit Background Color commands in Bitmap or Icon menus.

When you use this dialog box to customize a 256-color device, you need to turn off the Save With Default Device Colors check box in the Set Paint Editor Options dialog box to allow Resource Workshop to save your customized palette before you leave the Paint editor.

**Palette index**
The Palette index shows where the customized color goes in the Colors palette.

**RGB input boxes**
The RGB input boxes are where you enter values for customized colors.

**RGB slide bars**
The RGB slide bars are where you choose customized colors.

**Default button**
The Default button retrieves the color from the default palette (a Windows stock object) that has the same index as the Palette index.

**System button**
The System button retrieves the color from the system palette that has the same index as the Palette index. This button is enabled for devices that support 256 colors or more. It's disabled for standard VGA displays.

**Color boxes**
The Color boxes display the color you requested and the closest color permitted.

When you're finished changing colors, click OK or press Enter to put the new color in the Colors palette.

## Palette index

The Palette index helps you identify where the customized color goes in the Colors palette.

The Colors palette boxes are numbered from left to right across each row, starting with zero at the top left box. For example, in the default 16-color Colors palette, the number starts with 0 for black and continues to 7 for dark gray. It starts on the left side of the next row with 8 for light gray, and continues across to 15 for white.

The Palette index value changes when you enter new values in the RGB input boxes or use the RGB slide bars to choose a color.

## RGB input boxes

The RGB input boxes are where you edit a color. Just type in new values in the left column for the red, green, and blue (RGB) components of the color. (You can also use the RGB slide bars on the right side of the dialog box.)

Resource Workshop displays the closest matching color for the new RGB values in the Color boxes at the top of the dialog box.

After you enter a value in the left column, click on another box in the column to get the new value to take effect.

## RGB slide bars

The RGB slide bars are where you edit a color. Just move the cursor on the Red, Green, and Blue slide bars until the color you want is displayed in the Color boxes. (You can also use the RGB input boxes on the left side of the dialog box.)

## Color boxes

The Color boxes display the color you requested and the closest color permitted.

**Requested Color box**
The Requested color box displays the closest matching color for the RGB values you've selected in the RGB input boxes or slide bars.

**Granted Color box**
The Granted color box matches the RGB values to the closest color permitted by the limits of the Colors palette. You only see the Granted color box change color if the closest match is different from the current color.

## Set Transparent Color dialog box

The Set Transparent Color dialog box is where you change the colors the Paint editor displays for the transparent and inverted areas in your icon or cursor. Display this dialog box with Cursor|Set Transparent Color, Icon|Edit Foreground Color, or Icon|Edit Background Color.

When you use this dialog box to work on a 256-color device , you need to turn off the Save With Default Device Colors check box in the Set Paint Editor Options dialog box to allow Resource Workshop to save your customized palette before you leave the Paint editor.

**RGB input boxes**
The RGB input boxes are where you choose values for customized colors.

**RGB slide bars**
The RGB slide bars are where you edit a color. Just move the cursor on the Red, Green, and Blue slide bars until the color you want is displayed in the Color boxes. (You can also use the RGB input boxes on the left side of the dialog box.)

**Color boxes**
The Color boxes display the color you requested and the closest color permitted.

When you finish changing colors, click OK or press Enter to put the new color in the Colors palette.

## RGB input boxes

The RGB input boxes are where you edit a color. Just enter new values in the left column for the red, green, and blue (RGB) components of the color. (You can also use the RGB slide bars on the right side of the dialog box.)

Resource Workshop displays the closest matching color for the new RGB values in the Color boxes at the top of the dialog box.

If you enter a value in the left column, click on another box in the column to get the new value to take effect.

## Color boxes

Three Color boxes are displayed at the top of the dialog box.

**Requested Color box**
The Requested color box displays the closest matching color for the RGB values you've selected in the RGB input boxes or slide bars.

**Granted Color box**
The Granted color box matches the RGB values to the closest color permitted by the limits of the color device. You only see the Granted color box change color if the closest match is different from the current color. This is the color of the transparent areas in your image.

**Inverse Color box**
The Inverse color box shows the inverse of the color in the Granted Color box. This is the color of the inverted areas in your image.

## Window panes

The Paint editor lets you look at two different views of the image you're creating or editing. You can split the window vertically to show the two views side-by-side, or you can split the window horizontally to show one view above the other.

Use the View menu to split the window. To display a single image, just move the line that divides the windows off the screen. To change the size of the windows, move the line.

One or both views can be zoomed.

**See Also**
Zooming images

## Status line and bar

The status line at the bottom of the Paint editor displays information about commands and paint tools. It's divided into left and right sides.

There's also a status bar at the top of the screen that shows information about the resource you're working with.

**Status line: right side**
The right side of the status line tells you which paint tool you're using and its pixel coordinates on the screen. Depending on the tool you're using, you might see color information.

**Status line: left side**
The left side of the status line displays information about the currently highlighted menu command. For example, if you choose the View menu to zoom an image, the status line displays "Magnify active window."

## Tools palette

When you open a resource in the Paint editor, the Tools palette is in the upper right of the edit window. You use the Tools palette to choose the Paint editor tool you want to work with.

You can move the Tools palette around and you can hide it or show it.

Here's a list of the tools in the Tools palette:

 Airbrush

 Ellipse

 Eraser

 Filled Ellipse

 Filled Rectangle

 Filled Rounded Rectangle

 Line

 Paintbrush

 Paint Can

 Pen

 Pick Rectangle

 Rectangle

 Rounded Rectangle

 Scissors

 Text

 Zoom

There are four styles at the bottom of the Tools palette:

 Airbrush shape

 Paintbrush shape

 Pattern

 Pen style

In addition, there's a Hand tool that you can use to move a zoomed image. It's not displayed in the Tools palette.

**See Also**
Hiding and showing the Tools palette

## Hiding and showing the Tools palette

To hide the Tools palette,

- Double-click the Tools palette's Control-menu box or choose Hide from the Tools palette Control menu.
- Choose the Hide Toolbox command. This command is located on the menu bar selection that corresponds to the type of resource you're working with.

To show the Tools palette,

- Choose the Show Toolbox command on the menu bar selection that corresponds to the type of resource you're working with.

## Pick Rectangle tool

You use the Pick Rectangle tool to select a rectangular area of your image for copying, moving, or deleting. To select an area, click the left mouse button and drag the mouse until the flashing outline surrounds the area you want. Release the mouse button. Clicking either mouse button outside the outline turns the area selection off.

When you select an area, you can use the Edit menu commands to cut, copy, delete, duplicate, or paste into the selected area. You can also use the mouse to move or duplicate the area. In addition, you can align or resize the selected area.

**See Also**
Aligning a selected area
Resizing a selected area

## Scissors tool

The Scissors tool performs basically the same function as the Pick Rectangle tool: it selects an area of an image. However, with the scissors you can select and move areas of any shape, not just rectangles. To select an area, click the left mouse button and drag the scissors until the flashing outline surrounds the area you want, then release the mouse button.

When you select an area, you can use the Edit menu commands to cut, copy, delete, duplicate, or paste into the selected area. You can also use the mouse to move or duplicate the area. In addition, you can align or resize the selected area.

**See Also**
Aligning a selected area
Resizing a selected area

## Zoom tool

You use the Zoom tool to zoom the entire image, or you can outline an area of an image that you'd like to zoom, and have Resource Workshop zoom that area.

**See Also**
Zooming the entire image
Zooming a portion of the image

## Zooming the entire image

To zoom in on the entire image, double-click the Zoom tool.

Resource Workshop increases magnification of the image to the next level: 400%, 800%, or 1600%. You can also choose View|Zoom In to perform the same function on the currently selected window. The Paint editor uses the center of the image as a reference when zooming the entire image.

To zoom out the entire image, select the Zoom tool, hold down the Shift key, and double-click the Zoom tool. Resource Workshop decreases magnification to the next level: 800%, 400%, or 100%. You can also choose View|Zoom Out to perform the same function on the currently selected window.

When you working with two window panes, zooming affects only the current window.

**See Also**
Zooming images

## Zooming a portion of the image

To select an area you want to magnify, click the left mouse button and drag the Zoom tool. When the flashing outline surrounds the area you want to zoom, release the mouse button. Resource Workshop zooms the area to the largest zoom percentage that will fit in the window pane.

To zoom the area back out, choose View|Zoom Out or hold down the Shift key and double-click the Zoom tool.

**See Also**
Zooming images

## Eraser tool

The Eraser tool works like a square paintbrush - you drag the tool over the area you want to erase. To erase an entire image, double-click the Eraser tool.

Use the left mouse button to reveal the current background color (BG on the Colors palette). Use the right mouse button to reveal the current foreground color (FG on the Colors palette).

Since you use the eraser to reveal colors, the buttons on the mouse are the opposite of other tools you use to paint. For example, when you use the Paintbrush tool, you use the left mouse button to paint the foreground color. But with the eraser, the left mouse button reveals the background color.

Before you use the eraser, you might want to check the current colors in the Colors palette.

**See Also**
Background color overview
Foreground color overview

## Pen tool

Use the Pen tool to paint free-form lines and shapes using the current line style. To sketch with the Pen tool, click a mouse button and drag the pen across your image. Release the mouse button when you finish sketching.

- Use the left mouse button to sketch with the current foreground color (FG on the Colors palette).
- Use the right mouse button to sketch with the current background color (BG on the Colors palette).

Before you paint a line, you might want to select the line style and the foreground and background colors.

**See Also**
Choosing a line style
Background color overview
Foreground color overview

## Paintbrush tool

Use the Paintbrush tool to paint free-form patterns using the current brush pattern and shape. To paint, click a mouse button and drag the paintbrush across your image. Release the mouse button when you finish painting.

When you choose the Paintbrush tool, you see a cursor that represents the current brush shape. The area painted by the paintbrush is always proportionally the same relative to the size of the image frame.

- Use the left mouse button to sketch with the current foreground color (FG on the Colors palette).
- Use the right mouse button to sketch with the current background color (BG on the Colors palette).

Before you use the paintbrush, you might want to select the foreground and background colors and the paintbrush shape and pattern.

**See Also**
Background color overview
Choosing a paint pattern
Choosing a brush shape
Foreground color overview

## Airbrush tool

The Airbrush tool paints free-form patterns on your image using the current brush pattern and shape.

To use the airbrush, you can either click a mouse button once and drag the airbrush, or you can click it repeatedly, as if you were repeatedly pressing the nozzle of a spray can. If you drag it slowly, it paints a thick pattern. If you drag it quickly, it paints a scattered, thinner pattern.

When you choose the Airbrush tool, you see a cursor that represents the current brush shape. The area painted by the airbrush is always proportionally the same relative to the size of the image frame.

- Use the left mouse button to sketch with the current foreground color (FG on the Colors palette).
- Use the right mouse button to sketch with the current background color (BG on the Colors palette).

Before you use the airbrush, you might want to select the foreground and background colors and specify the brush shape and pattern.

**See Also**
Background color overview
Choosing a paint pattern
Choosing a brush shape
Foreground color overview

## Paint Can tool

Use the Paint Can tool to fill an area of your image with the currently selected color. It fills in any area of your image that is a single color.

- If you use this tool on an area that's not entirely surrounded by other colors, the color leaks out into other parts of the image that are the same color as the original area.
- If you hold down the Shift key when you use the Paint Can, you replace all instances of the color you click on, contiguous or not.

To use the Paint Can tool, point the paint can's cross hairs in the portion of the image you want to fill, then click a mouse button.

- Use the left mouse button to sketch with the current foreground color (FG on the Colors palette).
- Use the right mouse button to sketch with the current background color (BG on the Colors palette).

Before you use the paint can, you might want to specify the foreground and background colors.

**Note:** Because of problems inherent to display drivers, the Paint Can tool doesn't always paint properly. To solve this problem, add `RWS_OwnFloodFill=1` to the [RWS_Icon] section of WORKSHOP.INI.

**See Also**
Background color overview
Foreground color overview

## Line tool

Use the Line tool to paint straight lines. Press the mouse button and drag the Line tool across your image. Release the mouse button when you've finished drawing the line.

If you want the lines you paint to be limited to 45-degree increments, hold down Shift as you draw the line. With Shift down, you can paint only a horizontal or vertical line or a line on a 45-degree angle.

- Use the left mouse button to sketch with the current foreground color (FG on the Colors palette).
- Use the right mouse button to sketch with the current background color (BG on the Colors palette).

Before you paint a line, you might want to specify the line style and choose the foreground and background colors.

**See Also**

Background color overview
Choosing a line style
Foreground color overview

## Text tool

To add text to your image, choose the Text tool and click where you want the text to begin. A flashing cursor appears and you can begin typing text.

Before you use the Text tool, you might want to specify how and where you want the text displayed:

- Use Text|Font to specify the typeface, size, and style of the text.
- Use Text|Align Left, Text|Align Center, and Text|Align Right to specify how text is aligned.

Text is always displayed in the current foreground color. Before you type text, you might want to specify the foreground color by clicking the left mouse button on the color you want in the Colors palette.

**See Also**

Adding text to a resource
Foreground color overview

## Empty frame tools

You use one of these tools to paint an empty frame in your image:
Ellipse
Rectangle
Rounded Rectangle

To paint an empty frame,

1. Choose the tool you want.
2. Point the tool's cross hair where you want to start a corner of the frame.
3. Click a mouse button and drag the frame tool until the frame outline surrounds the area you want.
4. Release the mouse button.

Use the left mouse button to sketch with the current foreground color (FG on the Colors palette). Use the right mouse button to sketch with the current background color (BG on the Colors palette).

Before you paint a frame, you might want to specify the frame style and choose the foreground and background colors.

**See Also**
Background color overview
Choosing a line style
Foreground color overview

## Ellipse tool

Use the Ellipse tool to paint an ellipse-shaped empty frame in your image.

**See Also**
Empty frame tools

## Rectangle tool

Use the Rectangle tool to paint a rectangular empty frame in your image.

**See Also**
Empty frame tools

## Rounded Rectangle tool

Use the Rounded Rectangle tool to paint an empty frame shaped like a rounded rectangle in your image.

**See Also**
Empty frame tools

## Filled-in frame tools

Use one of these tools to paint filled-in frames in your image:
Filled Ellipse
Filled Rectangle
Filled Rounded Rectangle

These tools paint a frame using the current line style. The frame is filled using the current pattern and color. Specify a null pen width in the Set Pen Styles dialog box if you don't want Resource Workshop to put an outline around the filled-in pattern.

To paint a filled-in frame,

1. Choose the tool you want.
2. Point the tool's cross hairs where you want to start a corner of the frame.
3. Click a mouse button and drag the frame tool until the frame outline surrounds the area you want.
4. Release the mouse button.

Use the left mouse button to sketch with the current foreground color (FG on the Colors palette). Use the right mouse button to sketch with the current background color (BG on the Colors palette).

Before you paint a frame, you might want to specify the frame style and pattern and choose the foreground and background colors.

**See Also**
Background color overview
Choosing a line style
Choosing a paint pattern
Foreground color overview

## Filled Rectangle tool

Use the Filled Rectangle tool to paint a rectangular filled-in frame in your image.

**See Also**
Filled-in frame tools

## Filled Rounded Rectangle tool

Use the Filled Rounded Rectangle tool to paint a filled-in frame, in the shape of a rounded rectangle, in your image.

**See Also**
Filled-in frame tools

## Filled Ellipse tool

Use the Filled Rounded Ellipse tool to paint an ellipse-shaped filled-in frame in your image.

**See Also**
Filled-in frame tools

## Hand tool

Sometimes when you display a zoomed image, not all of it fits in the display. You can use the Hand tool to move the image around to see other parts of it.

Unlike other tools, the Hand tool isn't included in the Tools palette. But you can change any tool (except the Text tool) into a hand by holding down Ctrl.

The Hand tool is a grabbing tool because you just click the hand on the image and drag it in the direction you want it to move.

**See Also**
Moving the image around

## Pattern style

Some of the tools in the Tools palette paint a pattern on your image. These tools are:
Airbrush
Filled Ellipse
Filled Rectangle
Filled Rounded Rectangle
Paintbrush

Click the Pattern selection to display the Set Pattern dialog box, where you choose the pattern that's painted when you use these tools.

**See Also**
Choosing a paint pattern

## Pen style

Some of the tools in the Tools palette draw a line on your image. These tools are:
Line
Ellipse
Pen
Rectangle
Rounded Rectangle

Click the Pen style to display the Set Pen Style dialog box, where you choose the line style for these tools.

**See Also**
Choosing a line style

## Airbrush shape

The Airbrush shape shows the current shape of the Airbrush tool. Click the Airbrush shape to display the Airbrush Shape dialog box, where you choose the brush shape.

**See Also**
Choosing a brush shape

## Paintbrush shape

The Paintbrush shape shows the current shape of the Paintbrush tool. Click the Paintbrush shape to display the Brush Shape dialog box, where you choose the brush shape.

**See Also**
Choosing a brush shape

## Zooming images

Resource Workshop can show you an image at its actual size or it can zoom an image up to 1600%. You can use the Zoom tool or the View menu to zoom the image in one or both of the two window panes in the Paint editor. Resource Workshop always zooms the current image.

If you zoom an image to a size that's too large to fit in the pane, and the image is displayed at its true size in the other window pane, Resource Workshop places a dotted rectangle over the unzoomed image. This dotted rectangle indicates the portion of the image currently displayed in the zoomed window pane.

To help you control images on a pixel-by-pixel basis, you can display a grid on any zoomed image by choosing Options|Editor Options and turning on the Grid On Zoomed Windows check box. Each square of the grid represents a single pixel.

**See Also**
Moving a zoomed image around
Using the zoom accelerators

## Using the zoom accelerators

Three accelerators are listed on the View menu. They are easy to remember and make zooming much quicker.

Instead of choosing the View commands repeatedly until you've adjusted the image to your liking, it's much easier to press Ctrl+Z or Ctrl+O a few times. If you select the Zoom tool, you can simply double-click in the image to zoom in and Shift+double-click to zoom out. You can also press Ctrl+A to return the image to its actual size.

| View command | Accelerator | Mouse action on Zoom tool |
|---|---|---|
| Zoom In | Ctrl+Z | Double-click |
| Zoom Out | Ctrl+O | Shift+double-click |
| Actual size | Ctrl+A | None |

## Moving a zoomed image around

If a zoomed image is too large to fit in a window pane, Resource Workshop displays scroll bars to let you move the image around and see different parts of it.

You can also use the Hand tool to grab an image and move it. To turn the current tool into a Hand tool, press Ctrl. Then click in the image and drag it in the direction you want it to move. You might have to click and drag several times if you want to move a zoomed image more than the total width or length of the window pane.

If the image is also displayed unzoomed in another window pane, you can watch the dotted outline move as you look at different parts of the zoomed image.

## Adding text to a resource

You can add text to any resource that was created with the Paint editor. To add text to a resource, click the Text tool. Resource Workshop displays a flashing cursor. Click on the location in your image where you want text to start. Then type the text you want.

To determine how text is displayed, use the Text menu commands either before or immediately after you type the text. You can change the text color by selecting a new color from the Colors palette. You can also press the right mouse button if you want the text to be displayed in the current background color.

If you click another tool or click in another area of the image after entering text, you can't change anything in the text you've just entered. At that point, the text becomes just another part of the resource.

**See Also**
Aligning text
Choosing fonts, size, and text style

## Aligning text

To align text, you use Text|Align Left, Text|Align Center, and Text|Align Right.

Each of these commands controls where the text is displayed in relation to where you initially clicked before you typed the text. You can use an Align command before you type text or immediately after you finish typing. After you click to make another selection, you can't change the alignment of the text you've just typed.

**See Also**
Adding text to a resource

## Choosing fonts, size, and text style

To choose how text is displayed, use Text|Font to display the Select Font dialog box, where you choose the typeface, size, and style of the text.

After you click to make another selection, you can't change the font of the text you've just typed.

**See Also**
Adding text to a resource

## Choosing a brush shape

You can paint images in a resource using the Paintbrush tool or the Airbrush tool. You choose the shape of these tools.

The current paintbrush and airbrush shapes are always displayed as the top two of the four styles at the bottom of the Tools palette.

You can choose a paintbrush shape or an airbrush shape in one of the following ways:

- For a paintbrush shape, use the Options|Brush Shape command.
- For an airbrush shape, use the Options|Airbrush Shape command.
- Click the Paintbrush shape or the Airbrush shape style selection in the Tools palette.

If you choose a Paintbrush shape, you see the Brush Shape dialog box. If you choose an Airbrush shape, you see the Airbrush Shape dialog box.

## Choosing a paint pattern

Certain paint tools can paint a pattern on your image. These tools are:
Airbrush
Filled Ellipse
Filled Rectangle
Filled Rounded Rectangle
Paintbrush

The current pattern is displayed in the lower right corner of the Tools palette.

You can choose a pattern in one of the following ways:

- Choose Options|Pattern.
- Click the Pattern style selection in the Tools palette.

You see the Set Pattern dialog box, where you choose a pattern.

## Choosing a line style

You can control the line style produced by any of the following tools:
Line
Ellipse
Pen
Rectangle
Rounded Rectangle

The current line style is displayed in the lower left corner of the Tools palette.

You can choose a line style in the following ways:

- Choose Options|Pen Style.
- Click the Pen style selection in the Tools palette.

You see the Set Pen Style dialog box, where you choose a line style.

## Aligning a selected area

You can align a selected area of an image with the top, bottom, sides, or center of the current edit window. Aligning the selected area moves it to the location you specify, just as if you had selected the area and moved it with the mouse.

To align an area you've just selected, choose Options|Align. The Align Selection dialog box appears, where you choose the align options you want.

You select the area with the Pick Rectangle tool or the Scissors tool.

**See Also**
Pick Rectangle tool overview
Scissors tool overview

## Resizing a selected area

You can resize or move an area of an image you've selected. The Paint editor stretches or compresses the image inside the selection area accordingly. To resize an area, choose Options|Size. The Stretch Selection dialog box appears when you resize the image.

You select the area with the Pick Rectangle tool or the Scissors tool.

**See Also**
Pick Rectangle tool overview
Scissors tool overview

## Storage Format dialog box

The Storage Format dialog box is where you choose the storage format for your resource.

**Source button**
The Source button creates your resource in script format. Saving a resource as resource script puts it directly in your project file. This is usually how you save resources.

**Binary button**
The Binary button creates your resource in binary format that's referenced in your project file. This option saves a resource as a separate file that is linked to the current project.

## Source button

Select the Source button if you want to create your bitmap, cursor, font, or icon resource in source form as a resource script. Saving a resource as resource script puts it directly in your project file. This is usually how you save resources.

This option "embeds" the resource in the current project. The resource does not exist as a separate file and cannot be used in any other project.

When you create an icon, you then see the New Icon Image dialog box, where you specify characteristics of the icon.

When you create a bitmap, you see the New Bitmap Attributes dialog box, where you specify characteristics of the bitmap.

When you create fonts or cursors, you're put in the Paint editor.

## Binary button

Select the Binary button if you want to store your bitmap, cursor, font, or icon resource in a separate file that's referenced in your project file. This option saves a resource as a separate file that is linked to the current project. Choose this option to share the resource across several projects.

Resource Workshop displays the New File Resource dialog box, where you specify the file in which to store the resource. When you've made your selections, click OK.

When you create an icon, you then see the New Icon Image dialog box, where you specify characteristics of the icon.

When you create a bitmap, you see the New Bitmap Attributes dialog box, where you specify characteristics of the bitmap.

When you create fonts or cursors, you're put in the Paint editor.

## Icons

Icons are small bitmapped images, 64 x 64, 32 x 32, or 32 x 16 pixels in size. Windows programs typically use customized icons to represent minimized windows.

To design icons, use the Resource Workshop Paint editor.

Working with icons involves four basic steps:

1. Start the Paint editor.
2. Create or edit an icon.
3. Test the icon.
4. Save the icon.

**See Also**
Creating a new icon
Editing an existing icon
Saving an icon
Testing an icon
Using the Paint editor

## Creating a new icon

You can add a new icon to a project file, or create the icon in a standalone file. To add a new icon to an .RC (or .DLG) file,

1. Open a project or create a new one.
2. Choose Resource|New. Resource Workshop displays the New Resource dialog box.
3. In the Resource Type list box, double-click ICON.
4. Click OK to place the icon in the current project file or choose another project file.
5. In the Storage Format dialog box, choose the Source button to create an icon that's embedded in the project file.
6. Resource Workshop displays the New Icon Image dialog box, where you enter the size and color format of the new image.
7. Click OK to start customizing your icon.

After you choose the storage format, double-click the image in the Icon window or select the image and choose Images|Edit Image to customize it.

**See Also**
Creating a new icon in a standalone file
Creating a new project
Customizing an icon
Opening an existing project
Using the Paint editor

## Creating a new icon in a standalone file

To create a standalone icon file with the extension .ICO,

1. Open a project or create a new one.
2. Choose Resource|New. Resource Workshop displays the New Resource dialog box.
3. In the Resource Type list box, double-click ICON.
4. Click OK to place the icon in the current project file or choose another project file.
5. In the Storage Format dialog box, choose the Binary button to create a standalone icon file.
6. In the New File Resource dialog box, enter the name of the icon file and the project in which it will be referenced. Click OK.
7. Resource Workshop displays the New Icon Image dialog box, where you enter the size and color format of the new image.
8. Click OK to start customizing your icon.

To link a standalone icon file to a project file, choose File|Add to Project.

If you choose File|New Project and select .ICO in the New Project dialog box, you automatically create a standalone icon file. Resource Workshop immediately starts the Paint editor.

**See Also**
Creating a new project
Customizing an icon
Opening an existing project
Using the Paint editor

## Editing an existing icon

To edit an existing icon,

1. Open an existing project.
2. Double-click the name of the icon you want to edit, or select it and choose Resource|Edit.
3. If there's more than one image in the resource, the Icon window, is displayed. Choose the image you want to edit.

Once you choose the image, use the Paint editor to customize it.

In addition to creating and modifying icons directly with the Paint editor, you can also change the icon's resource script. It's unlikely that you'll want to do this because the script is almost entirely a series of hexadecimal values.

**See Also**
Customizing an icon
Editing an icon as text
Opening an existing project
Using the Paint editor

## Icon window

If there is more than one icon image in the selected resource, all the images are listed in the Icon window. Typically, multiple images represent different color formats of the same icon, such as a 2-color and a 16-color format.

To select an image,

1. Find the icon image you want to edit.

2. Double-click it, or select it and choose Images|Edit Image.

Resource Workshop displays the icon in the Paint editor.

**See Also**
Using the Paint editor

## Customizing an icon

After the icon is open in the Paint editor, you can customize it. Before you begin working on your icon, zoom it. You can change the icon's transparent and inverted areas and its attributes. You can also delete icon resources and images.

**Design issues**
Before you start, you should have an idea of what the icon represents to the user. The icon should be simple enough to fit into a 64 x 64, 32 x 32, or 32 x 16 pixel area.

Think about where the user is most likely to display your icon. Background colors and patterns can affect the icon's transparent and inverted areas.

**Drop shading**
Drop shading is a technique you can use to make your icon look three dimensional. For example, to make a black box look three dimensional, draw a gray border on the right side and bottom of the box.

**See Also**
Adding an image to an icon resource
Changing an icon's attributes
Deleting an icon resource or image
Zooming images

## Adding an image to an icon resource

You may want to put different color formats of the same icon in one icon resource. These color variations on the same icon are called images.

The reason the icon resource supports different color formats is that Windows picks a color format based on the ability of the display hardware to support the format. Windows picks a 2-color format for a monochrome display driver and a 16-color format for the standard Windows VGA driver.

Windows 3.x doesn't fully support the 256-color version of an icon, even if your display hardware supports it. Your program must supply its own support for 256 colors.

Here's how to add a new image to an existing icon resource,

1. Open a project.
2. Double-click the ICON entry you want to edit or select it and choose Resource|Edit.
3. Choose Images|New Image.
4. Resource Workshop displays the New Icon Image dialog box, where you choose the size and color format of the new image.
5. Choose the same size as the existing image and a new color format and click OK.
6. Double-click the new icon, or select it and choose Images|Edit Image. You see the Paint editor where you customize the icon as needed.

Typically, what you do next is open one of the existing icon images and copy it into the new (still blank) image. You might also have to customize the image if the colors are translated in a way that changes the form of the icon.

**See Also**
Customizing an icon
Opening an existing project
Using the Paint editor

## Changing an icon's attributes

You can change an icon's attributes - resolution or color format - with the Icon|Size and Attributes command.

This command brings up the Icon Image Attributes dialog box, where you change an icon's size and color format.

Changing the attributes actually changes the icon, as contrasted with using View|CGA Resolution [32 x 16] to view the icon at other resolutions.

**See Also**
Copying an icon image to a new color format

## Copying an icon image to a new color format

You might want to have different color formats of the same image for different displays. To create another version, without starting over, you can copy and paste the image:

1. Create a new image for your icon.
2. Open the image you want to copy in the Paint editor.
3. Choose Edit|Select All to select the entire image.
4. Choose Edit|Copy (or press Ctrl+Ins) to copy the icon image to the Clipboard.
5. Use Control|Close to close the window.
6. Double-click the new ICON entry in the Icon window.
7. Choose Edit|Paste (or press Shift+Ins) to paste the icon into the new image. It is displayed in the new color.

## Deleting an icon resource or image

You can delete an icon resource or an icon image. Deleting an icon resource deletes all the images in that resource. Deleting an icon image removes that single image in the icon resource.

**Deleting an icon resource**

To delete an icon resource, select it in the Project window, then:

- Press the Del key or choose Edit|Delete to completely delete it.
- Choose Edit|Cut to cut the resource into the Windows clipboard so you can paste it elsewhere.

**Deleting an icon image**

To remove an image from an icon resource,

1. Open the icon resource in Project window by double-clicking it or selecting it and choosing Resource|Edit.

2. In the Icon window, select the image entry you want to delete, then:

- Press the Del key or choose Edit|Delete to completely delete it.
- Choose Edit|Cut to cut the image into the Clipboard so you can paste it elsewhere.

## Testing an icon

The easiest way to test an icon is to use the Icon|Test command. You can also bind the icon resource to an executable file and then run the file to see what the icon looks like.

Two primary reasons for testing an icon are

- to see how a color icon looks in black and white when you move it around.
- to see how transparent and inverted areas look against various backgrounds.

The first test is easy to perform: just minimize the window and move the icon around.

The second test takes a little more work. Here's how:

1. Minimize the Paint editor window that contains the icon. Make sure the icon is visible and is in a spot where the Resource Workshop application workspace is its background.

2. Change tasks to the Program Manager.

3. Choose the Control Panel. When it appears, move it to the top of the screen.

4. Choose Color. In the Color dialog box, choose Color Palette.

5. Click in the area immediately surrounding the box with "Window Text" in it. You should see "Application Workspace" appear in the Screen Element box above the color palette.

6. Click on a color, then click OK.

7. Change tasks back to Resource Workshop and check the icon against the new background color.

At this point, you should be able to see both Resource Workshop and the Color dialog box at the same time. You can select another color by simply clicking on Colors in the Control Panel dialog box and performing steps 4 through 7.

When you change the color of the application workspace, Windows changes the color in the Resource Workshop window, where you can check the icon against the new color.

## Saving an icon

It's a good idea to save your icon as you go along, rather than waiting for Resource Workshop to prompt you when you close the project.

There are two primary options you can use to save changes you've made to an icon resource. You can save the entire project or save the icon resource as a file.

**See Also**
Saving an icon resource as a file
Saving the entire project

## Saving an icon resource as a file

If you just want to save an icon resource, you can choose Resource|Save Resource As to save the resource as a file.

You'll probably do this step only if you want to put the icon resource in binary format in a separate file for the first time. To save the resource as a file,

1. Choose Resource|Save Resource As.

2. Resource Workshop displays the Save Resource As dialog box where you can enter a new file name or choose the file name from the Files list.

3. Click OK or press Enter.

4. Resource Workshop asks you if you want the reference in the project file to refer to this external file from now on. Clicking Yes causes all future changes to the icon to be saved in binary format in the external icon file and not in the project file or in any previous icon file.

**See Also**
Saving an icon

## Editing an icon as text

To work with the resource script for an icon, select the icon in the Project window and choose Resource|Edit As Text. Resource Workshop brings up the source script for the resource in its internal text editor.

The only readily comprehensible parts of an icon resource script are the first and second lines ("BEGIN" and the resource name) and the last line ("END"). Everything between the second line and the last line is hexadecimal code. If you like, you can edit this code to see the effect on the icon, but do so at your own risk.

**See Also**
Using a text editor

## Cursors

Cursors are bitmapped images 32 x 32 pixels in size that represent the mouse pointer's current location on the screen. A Windows application often has a number of different cursors that represent different program functions.

Windows provides a set of standard cursors you can use in your programs. In addition, you can create your own customized cursors to represent different functions of the program.

To design cursors, you use the Paint editor.

Working with cursors involves four basic steps:

1. Start the Paint editor.
2. Create or edit a cursor.
3. Test the cursor.
4. Save the cursor.

**See Also**
Creating a new cursor
Editing an existing cursor
Saving a cursor
Testing a cursor
Using the Paint editor

## Creating a new cursor

You can add a new cursor to a project file, or create the cursor in a standalone file. To add a new cursor to an .RC (or .DLG) file,

1. Open a project or create a new one.
2. Choose Resource|New. Resource Workshop displays the New Resource dialog box.
3. In the Resource Type list box, double-click CURSOR.
4. Click OK to place the cursor in the current project file or choose another project file.
5. In the Storage Format dialog box, choose the Source button to create a cursor that's embedded in the project file.

Resource Workshop opens the Paint editor, where you customize the new cursor resource.

**See Also**
Creating a new cursor in a standalone file
Creating a new project
Customizing a cursor
Opening an existing project
Using the Paint editor

## Creating a new cursor in a standalone file

To create a standalone cursor file with the extension .CUR,

1. Open a project or create a new one.
2. Choose Resource|New. Resource Workshop displays the New Resource dialog box.
3. In the Resource Type list box, double-click CURSOR.
4. Click OK to place the cursor in the current project file or choose another project file.
5. In the Storage Format dialog box, choose the Binary button to create a standalone cursor file.
6. In the New File Resource dialog box, enter the name of the cursor file and the project in which it will be referenced. Click OK.

Resource Workshop opens the Paint editor, where you customize the new cursor resource. To link the standalone cursor file to a project file, choose File|Add to Project.

If you choose File|New Project and select .CUR in the New Project dialog box, you automatically create a standalone cursor file. Resource Workshop immediately starts the Paint editor.

**See Also**
Creating a new project
Customizing a cursor
Opening an existing project
Using the Paint editor

## Editing an existing cursor

To edit an existing cursor,

1. Open an existing project.

2. Double-click the CURSOR resource you want to edit or select it and choose Resource|Edit.

Once you've selected the cursor, use the Paint editor to customize it.

In addition to creating and modifying cursors directly with the Paint editor, you can also change the cursor's resource script. It's unlikely that you'll want to do this because the script is almost entirely a series of hexadecimal values.

**See Also**
Customizing a cursor
Editing a cursor resource as text
Opening an existing project
Using the Paint editor

## Customizing a cursor

After you have the cursor open in the Paint editor, you can customize it. Before you begin working on your cursor, you should zoom it. You can change the cursor's transparent and inverted areas. You can also delete cursors.

As you're customizing your cursor, you can see what it would look like in other resolutions with the View|CGA Resolution [32 x 16] command. Remember that Resource Workshop creates only 32 x 32 pixel cursors - when you use this command, all you change is the view of the cursor; it's still a 32 x 32 pixel cursor.

**Design issues**
Before you start, you should have an idea of what your cursor is intended to represent. A typical use of a custom cursor is to represent the task the user is performing.

The hot spot is the cursor's active area where the user clicks to activate the task represented by the cursor. You need to make the hot spot obvious.

Don't make the cursor complicated. It should be simple enough to fit into a 32 x 32 pixel area. Think about where the user is most likely to display your cursor. Background colors and patterns can affect the cursor's transparent and inverted areas.

**See Also**
Deleting a cursor
Setting a cursor's hot spot
Zooming images

## Setting a cursor's hot spot

An important consideration when you customize a cursor is where to put the hot spot (the cursor's active area). The hot spot is the single pixel in the cursor that fixes the location when the user places the cursor and clicks to make a selection.

Here's how you set hot spots:

1. Use the View|Zoom In command to zoom in on the cursor image until it's big enough to let you precisely choose the pixel coordinates for the hot spot.

2. Display a grid on the zoomed image.

3. Select the Line tool.

4. Point to the location on the zoomed image where you want the hot spot and look at the coordinates displayed on the right side of the status line. Make a note of these coordinates.

5. Choose Cursor|Set Hot Spot.

6. Enter the hot spot's pixel coordinates in the Set Hot Spot dialog box.

**See Also**
Customizing a cursor

## Pixel coordinates

Pixel coordinates are in horizontal (x) and vertical (y) units. The upper left pixel of the cursor image is x=0 and y=0. The lower right pixel for a 32 x 32 cursor is x=31 and y=31; for a 32 x 16 cursor its x=31 and y=16.

Pixel coordinates are displayed on the right side of the Paint editor status line.

## Deleting a cursor

To delete a cursor resource, select it in the Project window, then do one of the following:

- Press the Del key or choose Edit|Delete to completely delete it.
- Choose Edit|Cut to cut the resource into the Windows clipboard so you can paste it elsewhere.

## Testing a cursor

Anytime you want, you can test your cursor by choosing Cursor|Test.

Resource Workshop turns the current cursor into a test version of your cursor. You can move it around to see how it looks on different color backgrounds. When you finish testing, just click the mouse to continue customizing your cursor.

## Saving a cursor

It's a good idea to save your cursor as you go along, rather than waiting for Resource Workshop to prompt you when you close the project.

There are two primary options you can use to save changes you've made to a cursor resource. You can save the cursor resource as a file or save the entire project.

**See Also**
Saving a cursor resource as a file
Saving the entire project

## Saving a cursor resource as a file

If you just want to save the cursor resource, you can choose Resource|Save Resource As to save the resource as a file.

You'll probably do this step only if you want to put the cursor resource in binary format in a separate file for the first time. To save the resource as a file,

1. Choose Resource|Save Resource As.
2. Resource Workshop displays the Save Resource As dialog box, where you can enter a new file name or choose the file name from the Files list.
3. Click OK or press Enter.
4. Resource Workshop asks you if you want the reference in the project file to refer to this external file from now on. Clicking Yes causes all future changes to the cursor to be saved in binary format in the external cursor file and not in the project file or in any previous cursor file.

**See Also**
Saving a cursor

## Editing a cursor resource as text

In addition to creating and modifying cursors directly with the Paint editor, you can work with the cursor's resource script

To work with the resource script for a cursor, select the cursor resource in the Project window by clicking it. Then choose Resource|Edit As Text. Resource Workshop brings up the source script for the resource in its internal text editor.

The only readily comprehensible parts of a cursor resource script are the first and second lines ("BEGIN" and the resource name) and the last line ("END"). Everything between the second line and the last line is hexadecimal code. If you like, you can edit this code to see the effect on the cursor, but do so at your own risk.

**See Also**
Using a text editor

## Bitmaps

Bitmap resources display graphic images in your Windows program. Windows programs use bitmaps to represent scroll bar arrows, the Minimize and Maximize buttons, and so on.

To create bitmaps, you use the Resource Workshop Paint editor.

Working with bitmaps involves four basic tasks:

1. Start the Paint editor.
2. Create or edit a bitmap.
3. Test the bitmap.
4. Save the bitmap.

**See Also**
Creating a new bitmap
Editing an existing bitmap
Saving a bitmap
Testing a bitmap
Using the Paint editor

## Creating a new bitmap

You can add a new bitmap to a project file, or create the bitmap in a standalone file. To add a new bitmap to an .RC (or .DLG) file,

1. Open a project or create a new one.
2. Choose Resource|New. Resource Workshop displays the New Resource dialog box.
3. In the Resource Type list box, double-click BITMAP.
4. Click OK to place the bitmap in the current project file, or choose another project file.
5. In the Storage Format dialog box, choose the Source button to create a bitmap that's embedded in the project file.

Resource Workshop opens the Paint editor where you customize the new bitmap resource.

**See Also**
Creating a new bitmap in a standalone file
Creating a new project
Customizing a bitmap
Opening an existing project
Using the Paint editor

## Creating a new bitmap in a standalone file

To create a standalone bitmap file with the extension .BMP,

1. Open a project or create a new one.
2. Choose Resource|New. Resource Workshop displays the New Resource dialog box.
3. In the Resource Type list box, double-click BITMAP.
4. Click OK to place the bitmap in the current project file or choose another project file.
5. In the Storage Format dialog box, choose the Binary button to create a standalone bitmap file.
6. In the New File Resource dialog box, enter the name of the bitmap file and the project in which it will be referenced. Click OK.

Resource Workshop opens the Paint editor where you customize the new bitmap resource. To link the standalone bitmap file to a project file, choose File|Add to Project.

If you choose File|New Project and select .BMP in the New Project dialog box, you automatically create a standalone bitmap file. Resource Workshop immediately starts the Paint editor.

**See Also**
Creating a new project
Customizing a bitmap
Opening an existing project
Using the Paint editor

## Editing an existing bitmap

To edit an existing bitmap,

1. Open an existing project.

2. Double-click the BITMAP resource you want to edit, or select it and choose Resource|Edit.

Once you choose the image, use the Paint editor to customize the bitmap.

In addition to creating and modifying bitmaps directly with the Paint editor, you can also change the bitmap's resource script. It's unlikely that you'll want to do this because the script is almost entirely a series of hexadecimal values.

**See Also**
Customizing a bitmap
Editing a bitmap resource as text
Opening an existing project
Using the Paint editor

## Customizing a bitmap

After you open the bitmap in the Paint editor, you can customize it. Before you begin working on your bitmap, you should zoom it.

You can choose the bitmap's foreground and background colors, and change its attributes. You can also delete bitmap resources.

You can't, however, add transparent and inverted areas to your bitmap.

**See Also**
Changing a bitmap's attributes
Deleting a bitmap
Zooming images

## Changing a bitmap's attributes

You can change a bitmap's attributes with the Bitmap|Size and Attributes command.

This command brings up the Set Bitmap Attributes dialog box, where you choose:

- the bitmap image size
- whether the image will shrink or stretch if you change the overall image size
- the number of colors used in the image
- how the bitmap is stored

## Deleting a bitmap resource

To delete a bitmap resource, select it in the Project window, then

- Press the Del key or choose Edit|Delete to completely delete it.
- Choose Edit|Cut to cut the resource into the Windows clipboard so you can paste it elsewhere.

## Saving a bitmap

It's a good idea to save your bitmap as you go along, rather than waiting for Resource Workshop to prompt you when you close the project.

There are two primary options you can use to save changes you've made to a bitmap resource: You can save the bitmap resource as a file or save the entire project.

**See Also**
Saving the bitmap resource as a file
Saving the entire project

## Saving a bitmap resource as a file

If you only want to save the bitmap resource, you can choose Resource|Save Resource As to save the resource as a file.

You'll probably do this step only if you want to put the bitmap resource in binary format in a separate file for the first time. To save the resource as a file,

1. Choose Resource|Save Resource As.

2. Resource Workshop displays the Save Resource As dialog box where you can enter a new file name or choose the file name from the Files list.

3. Click OK or press Enter.

4. Resource Workshop asks if you want the reference in the project file to refer to this external file from now on. Clicking Yes causes all future changes to the bitmap to be saved in binary format in the external bitmap file and not in the project file or in any previous bitmap file.

## Testing a bitmap

To test your bitmap, you need to compile the bitmap resource and bind it to an executable file. Then you can run the executable file to see what the bitmap looks like.

## Editing a bitmap resource as text

In addition to creating and modifying bitmaps directly with the Paint editor, you can work with the bitmap's resource script.

To work with the resource script for a bitmap, select the bitmap from the Project window. Then choose Resource|Edit As Text. Resource Workshop brings up the source script for the resource in its internal text editor.

The only readily comprehensible parts of a bitmap resource script are the first and second lines ("BEGIN" and the resource name) and the last line ("END"). Everything between the second line and the last line is hexadecimal code. If you like, you can edit this code to see the effect on the bitmap, but do so at your own risk.

**See Also**
Using a text editor

## New Bitmap Attributes dialog box

The New Bitmap Attributes dialog box is where you choose the size and color format for a new image.

**Size input boxes**
The Size input boxes are where you choose the size of your bitmap.

**Colors radio buttons**
The Colors radio buttons determine the number of colors in the image.

## Fonts

A font is a collection of data used by a computer to draw individual characters on an output device. The font contains data that describes that overall collection of images, such as the typeface name, the suggested size, the character set, and so on. The font also contains information the computer needs to draw each character.

Windows supports three types of fonts: raster fonts, vector fonts, and TrueType fonts. Raster fonts contain a bitmapped image of each character. Vector fonts contain a series of drawing commands for each character. TrueType fonts, which are an enhanced vector font, are described in your Windows documentation. Resource Workshop creates and edits raster fonts only.

Typically, you use Resource Workshop to create picture fonts: small bitmaps that you want to group together. Although you could create a picture as a bitmap image, picture fonts and bitmaps aren't interchangeable. They differ in the way images are stored and retrieved at run time.

To create fonts, you use the Resource Workshop Paint editor. When you work with fonts, you see a status line at the top of the screen. The images in the font resource are displayed in the scroll bar on the right side of the screen.

Working with font resources involves five basic steps:

1. Start the Paint editor.
2. Create or edit a font image.
3. Save the font resource as part of a project file or in a separate file.
4. After exiting Resource Workshop, add the font resource to a special resource only DLL with a .FON extension.
5. Insert a call to the .FON file in your program, compile it, and test the font resource.

**See Also**

Adding a font resource to your application
Creating a new font resource
Differences in using font and bitmap resources
Editing an existing font resource
Saving a font resource
Scroll bar and status line
Testing a font resource
Using the Paint editor

## Differences in using font and bitmap resources

There are several reasons why you might want to define images as part of a font resource instead of a separate bitmap resource:

- It's simpler to write Windows code to load a font into memory and paint it. Loading and painting the same image that's stored as a bitmap is more complex.
- A font resource can contain up to 256 images. When your program needs a set of bitmapped images, you can create them as part of a single font resource, rather than in separate bitmap resources. Then at run time, you only need to load a single resource.
- A font resource can store multiple images more efficiently than the same image stored as individual bitmap resources. However, a font resource has a certain amount of memory overhead; each time you load a font into memory, Windows also loads the font header.

**See Also**

Defining a header for a font resource

## Scroll bar and status line

The font scroll bar and status line display information about the font resource you're editing.

**Scroll bar**
The images in the font resource are displayed on the right side of the Paint editor screen. Scroll through the images and click the one you want to edit.

**Status line**
The status line across the top of the font image displays:

- the face name assigned to the font on the left
- the current character you're editing on the right

**Face name**
The name you've assigned to the font is displayed on the left side of the status line. You use the Face Name input box in the Font Header Information dialog box to assign the face name.

**Current character**
The current character you're working on is displayed on the right side of the status line. You use the Character input boxes in the Font Size Information dialog box to map font images to the ANSI character set.

## Creating a font resource

You can add a new bitmap to a project file, or create the bitmap in a standalone file. To add a new bitmap to an .RC (or .DLG) file,

1. Open a project or create a new one.
2. Choose Resource|New. Resource Workshop displays the New Resource dialog box.
3. In the Resource Type list box, double-click FONT.
4. Click OK to place the font in the current project file or choose another project file.
5. In the Storage Format dialog box, choose the Source button to create a font that's embedded in the project file.

After you choose the storage format, Resource Workshop opens the Paint editor, where you customize the font.

**See Also**
Creating a standalone font file
Creating a new project
Customizing a font
Opening an existing project
Using the Paint editor

## Creating a standalone font file

To create a standalone font file with the extension .FNT,

1. Open a project or create a new one.
2. Choose Resource|New. Resource Workshop displays the New Resource dialog box.
3. In the Resource Type list box, double-click FONT.
4. Click OK to place the font in the current project file or choose another project file.
5. In the Storage Format dialog box, choose the Binary button to create a standalone font file.
6. In the New File Resource dialog box, enter the name of the font file and the project in which it will be referenced. Click OK.

Resource Workshop opens the Paint editor, where you customize the new font resource. To link the standalone font file to a project file, choose File|Add to Project.

If you choose File|New Project and select .FNT in the New Project dialog box, you automatically create a standalone font file. Resource Workshop immediately starts the Paint editor.

**See Also**
Creating a new project
Customizing a font
Opening an existing project
Using the Paint editor

## Editing an existing font resource

To edit an existing font resource,

1. Open an existing project.

2. Double-click the FONT resource you want to edit or select it and choose Resource|Edit. Each FONT entry in the Project window represents one font resource, though a single font can contain many images.

Once you choose the image, use the Paint editor to customize the font.

In addition to creating and modifying bitmaps directly with the Paint editor, you can change the font's resource script. It's unlikely that you'll do this because the script is almost entirely a series of hexadecimal values.

**See Also**
Customizing a font
Editing the font resource as text
Opening an existing project
Using the Paint editor

## Customizing a font resource

After you have the font resource open in the Paint editor, you can customize it. Before you begin working on your font, you should zoom it. Use the scroll bar and status line when you work with your font.

You can define the font's character set, create a variable-width font, and define the font header. You can also delete a single font image from the font resource or delete the entire resource.

By default, the Paint editor loads the first font image it finds in the font resource. To work on other fonts, you can click on any of the font images on the right side of the Paint editor screen.

**See Also**
Creating variable-width fonts
Defining and adding characters for a font
Defining a header for a font resource
Deleting a font image
Zooming images

## Defining and adding characters for a font

When you create a new font resource, it includes only one 8 x 8 pixel image. Usually you want more than one image in your font resource. You also might want to specify a different size for your font images.

To specify more than one image in a font resource and to change the size of font images, use the Font|Font Size menu command to display the Font Size Information dialog box.

**Setting the number of images**
Use the Character input boxes to choose how many images to include in your font. The First and Last values determine how many characters (images), the font resource will hold. If you set the number too low, you can change the values at any time.

**Mapping the character set**
In the Character input boxes, enter a range of decimal codes to use in mapping your font images to the ANSI character set. The ANSI values to which you map the images must be in the range 0 to 255.

**Defining the font size**
The images in a font resource can be variable-width or fixed-width.

- Fonts containing letters or images that can vary in width are called variable-width or proportional fonts. For example, in a variable-width font the letter "m" is considerably wider than the letter "i."
- Fonts in which the characters or images are all the same width are called fixed-width or monospaced fonts. Most typewriters use fixed-width fonts.

Variable-width characters usually take less space and are more pleasing to the eye than fixed-width font characters.

**See Also**
Creating variable-width fonts

## Creating variable-width fonts

To create a variable-width images or characters in your font resource,

1. Choose Font|Font Size. You see the Font Size Information dialog box.
2. Enter 0 (zero) in the Width input box.
3. Enter the maximum width, in pixels, for the images in the font resource in the Maximum Width input box. Click OK to save your selections.
4. Choose Font|Character Width. You see the Character Width dialog box, where you can adjust the character width size.
5. In the Width input box, enter a value that's less than or equal to the value in the Maximum Width display box.
6. Turn on the Stretch Current Characters check box to resize the font, so that existing images stretch or shrink, based on new width value.

## Defining a header for a font resource

Every font resource includes a header that contains general information about the font, such as typeface name and copyright information. If you define a font resource that consists of alphabetic characters, the header defines type style and size for all characters in the font.

To define the header for a font, choose Font|Header. You see the Font Header Information dialog box, where you define the:

- Font Version
- Face Name
- Device
- Copyright
- Font Attributes
- Type Size
- Leading
- Ascent

## Deleting a font image

To delete a font image,

1. Open the font resource in the Paint editor.
2. Choose Edit|Select All to select the font image.
3. Choose Edit|Cut, Edit|Delete, or press the Del key. The image disappears. The image might still appear on the right side of the screen, but will disappear when you choose another image to edit.
4. Choose File|Save Project to save your change.

## Saving a font resource

It's a good idea to save your font images as you go along, rather than waiting for Resource Workshop to prompt you when you close the project.

There are two primary options you can use to save changes you've made to a font resource. You can save the entire project or save the font resource as a file.

**See Also**
Saving the entire project
Saving the font resource as a file

## Saving a font resource as a file

If you just want to save a font resource, you can choose Resource|Save Resource As to save the resource as a file.

You'll probably do this step only if you want to put the font resource in binary format in a separate file for the first time. To save the resource as a file,

1. Choose Resource|Save Resource As.

2. Resource Workshop displays the Save Resource As dialog box, where you can enter a new file name or choose the file name from the Files list.

3. Click OK or press Enter.

4. Resource Workshop asks you if you want the reference in the project file to refer to this external file from now on. Clicking Yes causes all future changes to the font to be saved in binary format in the external font file and not in the project file or in any previous font file.

**See Also**
Saving a font

## Adding font resources to your application

Although Resource Workshop stores your fonts in your project file or a .FNT file, you also need to create a special, resource-only DLL that has a .FON extension. You can do this using Borland's Pascal or C++. You then load the DLL into your program with the Windows function **AddFontResource**.

Before you create the .FON file, save the project file as a resource file. Use the Resource|Save Resource As command.

First, put all the fonts you plan to put in the .FON file into an .RC file, then save it as a .RES file. One way to do this is to save all your fonts in separate .FNT files, open a new project file of type .RC, add all your font resources to it, and save the .RC file as a .RES file.

**Creating an .FON file**
Here's how to create a sample .FON file with Pascal or C++ and Turbo Assembler:

1. Use Turbo Assembler to compile the following dummy assembly-language program:
   ```
   end
   ```

2. Create a .DEF file for the .FON file with contents similar to the following:
   ```
   LIBRARY FontLib
   DESCRIPTION 'FONTRES 100,96,96 : Font Library Description'
   STUB 'WINSTUB.EXE'
   DATA NONE
   EXETYPE WINDOWS
   ```

3. Compile and link all the components into an .FON file, then bind the resources into the file. You can use a MAKE file similar to the following (this file assumes that you called everything FontLib):
   ```
   FontLib.fon: FontLib.res FontLib.obj FontLib.def
       TLINK /P-/A=16/Twd FontLib.obj,FontLib.fon,,, FontLib.def
       RLINK FontLib.res FontLib.fon
   ```

**See Also**
Font file types
Saving the font resource as a file

## Testing a font resource

To test your font, add an **AddFontResource** statement to your application that uses the .FON file. Recompile the application and run it with the new font loaded.

## Editing the font resource as text

In addition to creating and modifying fonts directly with the Paint editor, you can work with the font's resource script.

To work with the resource script for a font, select the font from the Project window by clicking it. Then choose Resource|Edit As Text. Resource Workshop brings up the source script for the resource in its internal text editor.

The only readily comprehensible parts of a font resource script are the first and second lines ("BEGIN" and the resource name) and the last line ("END"). Everything between the second line and the last line is hexadecimal code. If you like, you can edit this code to see the effect on the font, but do so at your own risk.

**See Also**
Using a text editor

## User-defined resources

In addition to defining standard resource types, you can also define your own resource types. After you create a new resource type, you can add any number of user-defined resources of this type to your project.

User-defined resources contain data that don't fit into one of the standard resource types. For example, if you want to create a character string resource that's longer than the STRINGTABLE limit of 255 characters, you can define your own resource type and store your character strings there.

You can also include metafiles in your project as user-defined resources. A metafile is a type of bitmap (in source form, it's a collection of Graphics Device Interface (GDI) calls) that's not only easier to scale and more device-independent than the standard bitmap resource, but also often takes up less storage space than a bitmap resource.

When you define a new resource, you can store data as part of the resource definition in a project file or as a separate file. You can also use the RCDATA resource type to add data to your application.

Working with user-defined resources involves five basic tasks:

1. Create a user-defined resource type.
2. Add a user-defined resource to your project.
3. Edit a user-defined resource.
4. Test the user-defined resource.
5. Save the user-defined resource.

**See Also**
Adding a user-defined resource
Creating a resource type
Editing a user-defined resource
Saving a user-defined resource
Testing a user-defined resource

## Creating a resource type

Before you can add user-defined resource data to your project, you must first create a type for it. Here's how:

1. Open an existing project or create a new one.

2. Choose Resource|New. Resource Workshop displays the New Resource dialog box.

3. Choose the New Type button to bring up the New Resource Type dialog box.

4. Enter a name for the resource type you're creating. For example, if you're creating a resource to contain a large block of text, you could name your new resource type TEXT. Once you've entered the name, click OK.

5. If your project is an .RC file, Resource Workshop asks you if want to create an identifier for the new resource type. Click OK.

6. Enter a value for the identifier in the New Identifier dialog box. This value is the ID that Windows and your program will associate with this identifier type. Click OK. If you use a resource ID, it must be greater than 255, because Windows reserves the values 1 through 255 for standard resources.

7. Click OK again when the New Resource dialog box is displayed.

From now on, whenever you create a new resource, you see that resource type listed in the New Resource dialog box with all the standard resource types.

**See Also**
Creating a new project
Adding a user-defined resource
Opening an existing project
Resource compiler files

## Adding a user-defined resource

After you've created a resource type, you can add a resource of that type to your project, as follows:

1. Open a project.
2. Choose Resource|New. Resource Workshop displays the New Resource dialog box.
3. In the Resource Type list box, double-click your user-defined resource.
4. Click OK to place the resource in the current project file or choose another project file.
5. Resource Workshop opens the text editor with a blank definition for your user-defined resource.

**See Also**
Creating a resource type
Editing a user-defined resource
Opening an existing project
Using the RCDATA resource type

## Using the RCDATA resource type

You can use the predefined RCDATA resource type to add a data resource to your application. It works the same as a user-defined resource type.

The primary difference between the two is addressability: you might prefer to have many different types of user-defined resources rather than just one type, RCDATA.

To add an RCDATA resource to your project:

1. Choose Resource|New.
2. Add a new resource whose type is RCDATA. You see a blank RCDATA definition in the text editor.
3. Add your resource data.

## Editing a user-defined resource

When you edit a user-defined resource, you work with its resource script.

To edit a user-defined resource,

1. Open a project. You can either create a resource type or open an existing one.

2. Select the user-defined resource type to which you want to add a resource.

3. Open the text editor by:

- Double-clicking the name of the resource you want to edit.
- Selecting the resource name and choosing Resource|Edit or Resource|Edit as Text.

Once you've brought the resource script up in the text editor, you can add or change data. To add data to your resource, do one of the following:

- Use the text editor to enter data between the BEGIN and END statements.
- Store the data in a separate file and add the file name to the end of the user-defined resource statement. Delete the BEGIN and END statements.
- Use the File|Add to Project command.

After you make any changes, you must recompile the resource to save your changes. If you exit without recompiling, you lose all your changes.

**See Also**
Embedding resource data in the project file
Entering data in the resource script
Opening an existing project

## Embedding resource data in the project file

You can add data to your resource by storing data in a separate file. The disadvantage to this approach is that if something happens to the file, the data is lost. Another option is to embed the external data into the project file script. Here's how:

1. Choose File|Add To Project to display the Add To Project dialog box.
2. Choose "User Resource Data" in the File Type list.
3. Enter the file name in the File Name input box. (The file must use a non-standard file extension.)
4. The Custom Resource dialog box is displayed.
5. Double-click the resource type. The resource appears in the Project window.

If you select the new resource and choose Resource|Edit, you see that the resource data is in hexadecimal format. For that reason, you should keep the external data file available in case you want to edit the resource script later.

## Entering data in the resource script

Use the text editor to change data that's between the BEGIN and END statements. Don't use this editor to do much formatting, since Resource Workshop rearranges the text when it compiles or decompiles the resource.

Here are some guidelines:

- The data can include any mix of numeric values and strings.

- You can use hexadecimal, octal, or decimal notation to represent numeric values. Use either 0x (a zero followed by the letter x) or $ (a dollar sign) as the leading characters for hexadecimal notation. This notation supports only 16-bit values. If you want to use an odd number of hexadecimal values, use a hexstring.

  A hexstring is a string of hexadecimal values enclosed in single quotation marks. The compiler ignores any spaces you insert to make the hex codes more readable.

- If you include text strings in your resource, enclose the strings in quotation marks, like this: "string". Strings aren't automatically null-terminated. To terminate a string with a null character, type \0 (backslash zero) at the end of the string.

## Testing a user-defined resource

You can't test user-defined resources within Resource Workshop. You need to use Resource Workshop to compile the project file that contains the resource and bind it to your executable file. You can then run your program and test the resource.

## Saving a user-defined resource

It's a good idea to save your resource as you go along, rather than waiting for Resource Workshop to prompt you when you close the project.

There are two primary options you can use to save changes you've made to a user-defined resource. You can save the entire project or save the user-defined resource as a file.

**See Also**
Saving the entire project
Saving the user-defined resource as a file

## Saving a user-defined resource as a file

If you want to save a user-defined resource, you can choose Resource|Save Resource As to save the resource as a file.

To save the resource as a file,

1. Choose Resource|Save Resource As.

2. Resource Workshop displays the Save Resource As dialog box, where you enter a new file name or choose the file name from the Files list.

3. Click OK or press Enter.

Resource Workshop asks you if you want the reference in the project file to refer to this external file from now on. Clicking Yes causes all future changes to the user-defined resource to be saved in the external file and not in the project file or in any previous user-defined resource file.

## Borland resource command line tools

Resource Workshop includes three resource-related command-line tools: RLINK.EXE, BRCC.EXE, and BRC.EXE:

- The Borland Resource Compiler (BRC.EXE) is a resource compiler shell with a high degree of compatibility with other resource compilers.
- BRCC.EXE is a command-line version of Resource Workshop's resource compiler.
- RLINK.EXE is a true Windows resource linker.

**See Also**

Differences between Borland and Microsoft compilers

## RLINK.EXE

RLINK is a true Windows resource linker. It accepts as input one or more object files (.RES) and a single Windows executable file.

It links the resources by fixing up string tables, icons and cursors, and name tables and then binds these linked resources to the executable. RLINK attempts to make your application run faster by building a preload section for the Windows loader.

**Syntax**

```
rlink [switches] <filename>.RES  <filename>.EXE
```

**Note:** <filename> indicates that you must supply a file name. The input and output names don't have to be the same; you can specify a .RES input file and an .EXE output file with different names.

**Note:** The .RES file name is optional.

| Switch | Description |
|---|---|
| @<filename> | Takes instructions from the specified command file. |
| -d | Removes resources from the .EXE file (no .RES file is specified). |
| -e | Moves the default DLL location into EMS memory. |
| -fe<filename> | Renames the output .EXE file. |
| -fi<filename> | Specifies additional input .RES files. |
| -k | Disables the contiguous preload of segments and resources in the .EXE file. Segments are kept in the order in which they appear in the .DEF file. |
| -l | Compiles an application that uses LIM 3.2 EMS. |
| -m | When Windows is running with LIM 4.0, causes each instance of an application task to be assigned to a separate EMS bank (the Multiple Instance flag). |
| -p | Creates a private DLL (called only by one application), optimizing Windows' use of memory. |
| -t | Creates an application that runs only in protected mode (Windows Standard or 386 enhanced mode). |
| -v | Prints progress messages (verbose listing). |
| -vx | Lists resources but does not bind to .EXE file |
| -30 | Marks the application as targeted for Windows 3.0 (or above). This is the default setting. |
| -31 | Marks the application as targeted for Windows 3.1 (or above). |

**See Also**

BRC.EXE
BRCC.EXE
RLINK examples

## RLINK examples

The following example marks the .EXE file with the default Windows version (3.0) and creates a preload area.

```
rlink <filename>.EXE
```

To mark the .EXE file for Windows 3.1, the switch must be explicitly stated:

```
rlink -31 <filename>.EXE
```

This example binds the resources in the .RES file into the .EXE file and creates a preload area.

```
rlink <myres>.RES <myfile>.EXE
```

This example links the resources in the two .RES files, binds them to the .EXE file, and creates a preload area.

```
rlink -fi<myres1>.RES <myres2>.RES <myfile>.EXE
```

The next example combines the program code in the input .EXE file with the resources in the input .RES file, produces an output .EXE file with a new name, and creates a preload area.

```
rlink -fe<myres1>.exe <myres2>.RES <myfile>.EXE
```

The final example takes input from an .RLK command file. It then links the resources in three .RES files, binds them to the .EXE file, and creates a preload area.

```
rlink @<filename>.rlk
```

The command file (<filename>.RLK) contains:

```
-fi<filename>.res
-fi<filename>.res
<filename>.res
<filename>.exe
```

**See Also**

BRC.EXE
BRCC.EXE
RLINK.EXE

# BRCC.EXE

BRCC is a command-line version of Resource Workshop's resource compiler. It accepts a resource script file (.RC) as input and produces a resource object file (.RES) as output.

**Syntax**

```
brcc [switches] <filename>.RC
```

| Switch | Description |
|---|---|
| -d<name>[=<string>] | Defines a preprocessor symbol. |
| -fo<filename> | Renames the output .RES file. (By default, BRCC creates the output .RES file with the same name as the input .RC file.) |
| -i<path> | Adds one or more directories (separated by semicolons) to the include search path. |
| -r | This switch is ignored. It is included for compatibility with other resource compilers. |
| -v | Prints progress messages (verbose). |
| -x | Deletes the current include path. |
| -30 | Creates a Windows 3.0-compatible .RES file. |
| -31 | Creates a Windows 3.10-compatible .RES file. |
| -? or -h | Displays switch help. |

**Symbols**

Like Resource Workshop's resource compiler, BRCC predefines common resource-related Windows constants such as WS_VISIBLE and BS_PUSHBUTTON. In addition, two special compiler-related symbols are defined: RC_INVOKED and WORKSHOP_INVOKED.

These symbols may be used in the source text in conjunction with conditional preprocessor statements to control compilation. For example, the following construct can greatly speed up compilation:

```
#ifndef WORKSHOP_INVOKED
#include "windows.h"
#endif
```

**See Also**

BRC.EXE
BRCC examples
RLINK.EXE

## BRCC examples

The following example adds two directories to the include path and produces a .RES file with the same name as the input .RC file.

```
brcc -i<dir1>;<dir2> <filename>.RC
```

This example produces an output .RES file with a name different from the input .RC file's.

```
brcc -fo<filename>.RES <filename>.RC
```

**See Also**

BRC.EXE
BRCC.EXE
RLINK.EXE

## Borland Resource Compiler (BRC.EXE)

The Borland Resource Compiler (BRC) is a resource compiler shell with a high degree of compatibility with other resource compilers. It accepts command lines formatted for those resource compilers and invokes either BRCC or RLINK or both, depending on the command-line syntax.

**Syntax**

```
brc [switches] <filename>.RC [<filename>.EXE]
```

**Note:** <filename> indicates that you must supply a file name. The input and output names don't have to be the same; you can specify an .RC input file and an .EXE output file with different names.

| Switch | Description |
|---|---|
| -d | Defines a symbol you can test with the #ifdef preprocessor directive. |
| -fo<filename> | Renames the .RES file. |
| -fe<filename> | Renames the .EXE file. |
| -fi<filename> | Specifies additional input .RES files. |
| -i<path> | Adds one or more directories (separated by semicolons) to the include search path. |
| -r | Creates a .RES file only. The compiled .RES file is not added to the .EXE file. |
| -v | Prints progress messages (verbose listing). |
| -x | Directs the compiler to ignore the INCLUDE environment variable when it searches for include or resource files. |
| -30 | Marks the application as targeted for Windows 3.0 (or above). This is the default setting. |
| -31 | Marks the application as targeted for Windows 3.1 (or above). |
| The following switches are invalid when the -r switch is specified: | |
| -e | Moves the default DLL location into EMS. |
| -k | Disables the contiguous preload of segments and resources in the .EXE file. Segments are kept in the order in which they appear in the .DEF file. |
| -l | Compiles an application that uses LIM 3.2 EMS. |
| -m | When Windows is running with LIM 4.0, causes each instance of an application task to be assigned to a separate EMS bank (the Multiple Instance flag). |
| -p | Creates a private DLL that is called only by one application, optimizing Windows' use of memory. |
| -t | Creates an application that runs only in protected mode (Windows Standard or 386 enhanced mode). |

**Note:** If you use the Microsoft 3.1 Resource Compiler, the -30 and -31 switches are invalid when the -r switch is specified. This is because the Microsoft 3.1 Resource Compiler can't produce 3.0-compatible .RES files.

**See Also**

BRCC.EXE
BRC examples
RLINK.EXE

## BRC examples

The following statement compiles the .RC file, creates a .RES file, and adds the .RES file to the executable file.

```
brc <filename>.RC [<filename>.EXE]
```

BRC automatically seeks an .EXE file with the same name as the .RC file. You need only specify the .EXE file if its name is different from the .RC file's.

The following statement creates a .RES file, but not an .EXE file. If you name an .EXE file in the command line, BRC ignores it.

```
brc -r <filename>.RC
```

The following statement compiles a Windows 3.1 version of a DLL file. You must give the DLL file's extension (.DLL, .DRV, or .EXE) in the command line.

```
brc -31 <filename>.DLL
```

The following statement adds an existing .RES file to an executable file. The .EXE filename is required only if it differs from the .RES filename.

```
brc <filename>.RES [<filename>.EXE]
```

**See Also**

BRC.EXE
BRCC.EXE
RLINK.EXE

## Differences between Borland and Microsoft compilers

The Resource Workshop resource compiler is almost completely Microsoft-compatible and is significantly enhanced over the Microsoft Resource Compiler in a number of ways.

The following features are improvements over the Microsoft compiler:

- The Resource Workshop compiler allows text descriptions of bitmapped resources (icons, cursors, bitmaps, and fonts), while the Microsoft compiler does not. The text descriptions are written using the resource script language.

- The Resource Workshop compiler supports numeric constant expressions for every numeric field, while the Microsoft compiler doesn't.

- Resource Workshop has added a new fundamental data type, the hexstring. This data type consists of a variable number of hexadecimal digits that describe data bytes, surrounded by single quotation marks.

- The Resource Workshop compiler supports references to files in RCDATA resources as well as in user-defined resources. Support of file references removes the only distinction between user-defined resources and RCDATA resources. If you use the Microsoft resource compiler to compile an RCDATA resource that contains a file reference, you'll get a syntax error.

The Microsoft Resource Compiler and Resource Workshop are incompatible in the following areas:

- Complex constant expressions
- Duplicate resource IDs
- Expressions in resources IDs and resource type IDs
- Floating END statements
- Floating operators in expressions
- Hexadecimal numbers in resources IDs and resource type IDs
- Numbers with leading zeros
- Macros in include directives
- Missing operators in expressions
- Parsing of the CAPTION statement
- Preprocessor token pasting
- Resource IDs greater than 32767
- Support of the CTLDATA statement
- The #undef preprocessor directive
- Valid characters in resource names

## Complex constant expressions

Resource Workshop supports full C-language constant expressions in place of a simple number anywhere in a resource script where a number is allowed. The Microsoft Resource Compiler supports only simple expressions.

For example, the following expression is correctly evaluated by Resource Workshop, but fails using the Microsoft Resource Compiler:

```
3 * (1 + 2) - 1
```

The most common example of this incompatibility is often seen in ICON statements in DIALOG templates. The following statement causes an error in Resource Workshop, but not in the Microsoft Resource Compiler:

```
ICON   3 -1, 10, 10, 0, 0
```

Resource Workshop interprets "3 -1" as an expression that evaluates to 2. The Microsoft Resource compiler interprets "3 -1" as two separate fields. If you add a comma after the first number, both compilers interpret the statement correctly:

```
ICON   3, -1, 10, 10, 0, 0
```

**See Also**
Resource script language

## Duplicate resource IDs

To allow resources to be accessed at run time, Resource Workshop enforces the rule that resource IDs or names must be unique within each resource type; the Microsoft Resource Compiler does not.

The following statements cause Resource Workshop to display an error:

```
1 ICON file1.ico
1 ICON file2.ico
```

**See Also**
Resource script language

## Expressions in resources IDs and resource type IDs

Resource Workshop supports expressions in resource IDs; the Microsoft Resource Compiler does not. For example, the following statement compiles correctly using Resource Workshop, but fails using the Microsoft Resource Compiler:

```
101 + 1000 BITMAP vga.bmp
```

The Microsoft Resource Compiler parses "101" as a resource ID, "+" as a resource type name, "1000" as a file name, and then fails. Resource Workshop correctly emits a bitmap resource with an ID equal to 1101.

**See Also**

Resource script language

## Floating END statements

Resource Workshop does not allow END statements with no corresponding BEGIN; the Microsoft Resource Compiler does.

For example, the following script fragment causes an error in Resource Workshop:

```
1 RCDATA
BEGIN
  0
END

END
```

**See Also**

Resource script language

## Floating operators in expressions

Resource Workshop's expression parser does not allow "floating" operators in constant expressions; the Microsoft Resource Compiler does. For example, the following expression is flagged as an error in Resource Workshop:

```
WS_SYSMENU | WS_CAPTION |
```

To correct the error, remove the last bitwise OR operator:

```
WS_SYSMENU | WS_CAPTION
```

**See Also**

Resource script language

## Hexadecimal numbers in resources IDs and resource type IDs

Resource Workshop supports hexadecimal numbers in resource IDs; the Microsoft Resource Compiler does not.

For example, the following statement compiles correctly using Resource Workshop, but fails using the Microsoft Resource Compiler:

```
0x0001  ICON file.ico
```

The Microsoft Resource Compiler emits an icon resource with the name "0x0001". Resource Workshop emits an icon resource with an ID equal to 1.

**See Also**

Resource script language

## Numbers with leading zeros

Because of inconsistencies in the Microsoft Resource Compiler's treatment of numbers with leading zeros, you shouldn't use them in preprocessor expressions or identifiers. The Resource Workshop compiler is consistent in interpreting any numeric constant preceded by a zero and used as part of an identifier or a preprocessor expression as an octal number.

However, the Microsoft Resource Compiler interprets numbers with leading zeros in preprocessor expressions as octal numbers, but interprets the same numbers in identifiers as decimal numbers.

For example, the Microsoft Resource Compiler would interpret the expression 010+1 as a 9 in the following preprocessor expression, but as an 11 in the string table identifier.

```
#IF (9 == 010+1)
   STRINGTABLE
   BEGIN
       010+1, "Bug"
   END
#ENDIF
```

**See Also**

Resource script language

## Macros in include directives

Resource Workshop does not support macro expansion in include directives. For example, the following fragment causes a compile error:

```
#define MYFILE "afile.h"
#include MYFILE
```

**See Also**
Resource script language

## Missing operators in expressions

Resource Workshop's expression parser requires that all operators required for an expression be present. The Microsoft Resource Compiler assumes that a missing operator is a bitwise OR operator.

For example, the following expression is flagged as an error in Resource Workshop:

```
WS_SYSMENU  WS_CAPTION
```

To correct the error, add the bitwise OR operator:

```
WS_SYSMENU | WS_CAPTION
```

**See Also**

Resource script language

## Parsing of the CAPTION statement

The Microsoft Resource Compiler is order-dependent in the way it handles the CAPTION statement in a DIALOG template; the Resource Workshop compiler is not. The following fragments illustrate this difference.

In the following example, there is no explicit STYLE statement:

```
1 DIALOG 10, 10, 100, 100
CAPTION "Caption"
BEGIN
END
```

Both compilers default to WS_POPUP | WS_BORDER | WS_SYSMENU, which is OR'd with WS_CAPTION. The style of the resulting DIALOG template is

```
WS_POPUP | WS_BORDER | WS_SYSMENU | WS_CAPTION
```

In this example, the STYLE statement precedes the CAPTION statement:

```
2 DIALOG 10, 10, 100, 100
STYLE WS_POPUP
CAPTION "Caption"
BEGIN
END
```

Both compilers OR the two together. The style of the resulting DIALOG template is

```
WS_POPUP | WS_CAPTION
```

The CAPTION statement precedes the STYLE statement in this example:

```
3 DIALOG 10, 10, 100, 100
CAPTION "Caption"
STYLE WS_POPUP
BEGIN
END
```

The Microsoft Resource Compiler, having encountered an explicit STYLE statement, clears any preceding implicit or preset styles. The resulting DIALOG template has the style:

```
WS_POPUP
```

By contrast, the Resource Workshop compiler ORs the CAPTION and STYLE statements to produce a DIALOG template with the style:

```
WS_CAPTION | WS_POPUP
```

**See Also**

CAPTION
DIALOG
Resource script language
STYLE

## Preprocessor token pasting

Resource Workshop does not support token pasting in preprocessor statements. Token pasting applies only to the C language. If you're using Resource Workshop with a Pascal compiler, you can disregard this.

## Resource IDs greater than 32767

Although the Microsoft documentation states that resource IDs can be any number between 1 and 65535, you must actually use resource IDs that fall between 1 and 32767 to ensure that the IDs are unique.

The reason is that resource binders like RLINK or the Microsoft Resource Compiler OR all resource IDs with 0x8000 when they build the resource directory in the executable file, producing a value from 32768 to 65535.

**See Also**
RLINK.EXE

## Support of the CTLDATA statement

Resource Workshop supports the CTLDATA statement for use with custom controls. The Microsoft Resource Compiler does not support CTLDATA.

## The #undef preprocessor directive

Resource Workshop has limited support for the #undef preprocessor directive. You can use it only with #defines that are not referenced by a resource.

If you use #undef with a #define that's a resource identifier, you get a fatal compiler error when compiling the RC file under Resource Workshop.

The #undef preprocessor directive applies only to the C language. If you're using Resource Workshop with a Pascal compiler, you can disregard this.

**See Also**
Directives

## Valid characters in resource names

Resource Workshop follows standard C-language practice for valid characters in resource names.

- It accepts any alphanumeric characters in the ANSI character set (including accented non-English-language characters) plus the underscore.
- Nonalphanumeric characters such as slash, backslash, or the plus sign are not valid.
- The resource name must start with a letter or the underscore; it cannot start with a number.

The Microsoft Resource Compiler does not restrict resource names to alphanumeric characters. For example, a resource named "Test-Bar" is interpreted as "Test minus Bar" by the Borland Resource Compiler, but is interpreted as "Test-Bar" by the Microsoft Resource Compiler.

## Resource Script Language

When you create resources, Resource Workshop builds a resource script file in your project window. Most of the time, you won't need to use this language. Occasionally, however, you might want to edit a resource script with a text editor.

Each resource script statement specifies a resource to be included in your executable file. You can list the primary statements in any order you choose; there is no "program flow" in a resource script file, although you must follow the syntax given for each statement carefully.

**See Also**

Alphabetical listing of language elements
Functional listing of language elements

## Alphabetical listing of the Resource Script language elements

This is an alphabetical listing of the Resource Script language elements.

For a functional listing, go to the Functional listing of Resource Script Language elements Help screen.
For an overview of the language, go to the Resource Script Language Help screen.

SCROLLBAR class style constants
STATIC class style constants
STRINGTABLE
User-defined resources
VERSIONINFO
Window style constants

## Functional listing of the Resource Script language elements

This is a functional listing of the Resource Script language elements.

For an alphabetical listing, go the Alphabetical listing of Resource Script Language elements Help screen. For an overview of the language, go to the Resource Script Language Help screen.

**Constants**

**DIALOG**
- CAPTION
- CHECKBOX
- CLASS
- COMBOBOX
- CONTROL
- CTEXT
- DEFPUSHBUTTON
- EDITTEXT
- FONT
- GROUPBOX
- ICON
- LISTBOX
- LTEXT
- MENU
- PUSHBUTTON
- RADIOBUTTON
- RTEXT
- SCROLLBAR
- STYLE

**Directives**
- #define
- #elif
- #else
- #endif
- #error
- #ifdef
- #if
- #ifndef
- #include
- #line
- #pragma
- RCINCLUDE

**MENU**
- MENUITEM
- POPUP

**Statements, binary resource**
- BITMAP
- CURSOR
- FONT
- ICON

**Statements, multiple-line**
- ACCELERATOR
- DIALOG
- MENU

RCDATA
STRINGTABLE

**User-defined resources**

**Version stamping**
VERSIONINFO

## Binary resource statement

Binary resource statements identify a resource and name the file that contains the resource.

For example,
```
pencil CURSOR MOVEABLE pencil.cur
```
names the resource "pencil" and identifies it as a CURSOR resource located in the file PENCIL.CUR.

The syntax of all binary resources statements looks like this:
```
resource-name resource-type [load-type] [memory-option] filename
```

Binary resource statements specify bitmaps, cursors, fonts, and icons. Each statement has its own syntax. The binary resources statements are:
BITMAP
CURSOR
FONT
ICON

**See Also**
Multiple-line statements

## Multiple-line statements

Multiple-line statements specify the contents of a resource.

Here's an example of a multiple-line statement for a menu resource:

```
mainmenu MENU PRELOAD
BEGIN
  POPUP "&File"
  BEGIN
    MENUITEM "&New", 100
    MENUITEM "&Open", 101
    MENUITEM "&Close", 102, GRAYED
    MENUITEM "&Save", 103
    MENUITEM "Save &As", 104
    MENUITEM SEPARATOR
    MENUITEM "&Print", 105
    MENUITEM "&Draft Printing", 107, CHECKED
    MENUITEM SEPARATOR
    MENUITEM "E&xit", 106
  END
    MENUITEM "&Help", 200
END
```

This example specifies the main menu of an application. The main menu contains the File and Help menus.

File is a pop-up menu (commonly known as a drop-down menu) with several options, each defined with a MENUITEM substatement contained within the BEGIN and END keywords. This is characteristic of all multiple-line statements. Like other multiple-line statements, this statement permits you to include multiple substatements between the words BEGIN and END.

Multiple-line statements specify accelerators, string resources, raw data resources, dialog boxes, and menus. Each statement has its own syntax. The multiple-line statements are:
ACCELERATOR
DIALOG
MENU
RCDATA
STRINGTABLE

**See Also**
Binary resource statement

## Directives

Directives are special statements that affect how the resource script file is compiled. They do not define resources. Unlike other resource script statements, directives are case sensitive.

You can use directives to:

- assign values to names you use in other resource script statements
- include the contents of other files
- specify how the script file should be compiled

If you are a C or C++ programmer, the directive statements will look familiar to you. They are identical to those you use when compiling your programs.

Here are the directives:

| Directive | Description |
|---|---|
| #define | Assigns a given value to the name you specify. |
| #elif | Controls conditional compilation and marks an optional block. Used with #if. |
| #else | Controls conditional compilation and marks an optional block. Used with #if, #ifdef, #ifndef, and #elif. |
| #endif | Controls conditional compilation and marks the end of a block. Used with #if, #ifdef, #ifndef, and #elif. |
| #error | Displays user-defined error message. |
| #if | Marks the start of conditional compilation. |
| #ifdef | Marks the start of conditional compilation if a specified name has been #defined. |
| #ifndef | Marks the start of conditional compilation if a specified name has not been #defined. |
| #include | Puts the contents of the named file into your resource script before it's compiled. |
| #line | Ignores text on the specified line number. |
| #pragma | Directive is ignored. |
| RCINCLUDE | Puts the contents of the named file into your resource script before it's compiled. |

## #define

```
#define identifier text
```

**Parameters**

| | |
|---|---|
| identifier | Only recognized and replaced by the compiler when it is followed by whitespace and occurs outside a quoted string. |
| text | Can contain any valid character. Is normally terminated by the end of the line on which the #define occurs. text can also be empty. To continue text onto another line, place a backslash character (\) at the end of the source line. |

**Remarks**

#define is a directive that causes the compiler to substitute text for each subsequent reference to identifier in the resource source text. Both the #define directive and the identifier are case sensitive.

There are two #define identifiers that are always defined in Resource Workshop:

- RC_INVOKED
- WORKSHOP_INVOKED

Testing for these values in an #if or #ifdef directive always yields 1.

Parameterized #define directives that use the syntax `#define(parameter-list)` are not supported by Resource Workshop and can't be referenced. However, #define directives of this type can be included in .H files as long as they are not referenced.

The #define directive is a subset of the #define statement.

**See Also**

#if
#ifdef
Examples

## #elif

```
#elif constant-expression
```

**Parameters**

| | |
|---|---|
| constant-expression | Evaluated as a signed long integer. If constant-expression evaluates to a nonzero value, the text following the #elif directive is processed by the compiler. Otherwise, the text between the #elif and a following #elif or #endif directive is ignored by the compiler. |

**Remarks**

#elif is a directive that is used in conjunction with the #if directive to control conditional compilation. This directive is lost if it occurs within a text resource and that resource is edited by Resource Workshop.

The #elif keyword is case sensitive; #ELIF causes a compile error.

**See Also**

#define
#if
#endif
Examples

## #else

```
#else
```

**Remarks**
#else is a directive that is used in conjunction with the #if, #ifdef, #ifndef, and #elif directives to control conditional compilation.

The compiler processes source code following the #else directive until the next #endif directive only when none of the previous conditional directives within the current scope have been processed.

If any previous conditional directives within the current scope have been processed, the source code between the #else and the following #endif will be ignored. This directive is lost if it occurs within a text resource and that resource is edited by Resource Workshop.

The #else keyword is case sensitive; #ELSE causes a compile error.

**See Also**
#define
#elif
#endif
#if
#ifdef
#ifndef
Examples

## #endif

```
#endif
```

**Remarks**
#endif is a directive that marks the end of the scope of the current conditional compilation. It is used in conjunction with the #if, #ifdef, #ifndef, #else, and #elifelif directives to control conditional compilation.

This directive is lost if it occurs within a text resource and that resource is edited by Resource Workshop.

The #endif keyword is case sensitive; #ENDIF will cause a compile error.

**See Also**
#elif
#else
#if
#ifdef
#ifndef
Examples

## #error

```
Error: filename line# : #error directive encountered: error text
```

**Remarks**
#errors is a directive that displays a user-defined error message.

This directive usually catches an undesired compile-time condition. In the normal case, the condition would be false. If the condition is true, you want the compiler to print an error message and stop the compilation. You do this by placing an #error directive in a conditional that is true for the undesired case.

## #if

```
#if constant expression
#if [!]defined identifier
#if [!]defined (identifier)
```

**Parameters**

When #if is followed by constant expression, the constant expression is evaluated as a signed long integer:

- If the resulting value is nonzero, the source text following the #if directive until the next #else, #elif, or #endif directive is processed by the compiler.
- If the resulting value is zero, the source text following the #if directive until the next #else, #elif, or #endif directive is ignored by the compiler.

The other formats of the #if directive can be used in place of the #ifdef and #ifndef directives. The keyword must be lowercase, but it can be preceded by a ! operator. The following identifier (optionally surrounded by parentheses) is searched for in the compiler's table of #defines. If the #define is found, the source text following the #if directive is processed. The ! operator reverses the effect of the preceding test.

**Remarks**

#if is a directive that marks the start of conditional compilation. #if directives can be nested. This directive is lost if it occurs within a text resource and that resource is edited by Resource Workshop.

The #if keyword is case sensitive; #IF will cause a compile error.

**See Also**

#define
#elif
#else
#endif
#ifdef
#ifndef
Examples

## #ifdef

```
#ifdef identifier
```

**Parameters**

identifier — Searched for in the compiler's table of #defines.

- If identifier is found, the source text following the #ifdef directive until the next #else, #elif, or #endif is processed.
- If identifier is not found, the source text following the #ifdef directive until the next else, #elif, or #endif is ignored.

**Remarks**

#ifdef is a directive that marks the start of conditional compilation.

The #ifdef keyword is case sensitive; #IFDEF will cause a compile error.

**See Also**

#define
#elif
#else
#endif
Examples

## #ifndef

```
#ifndef identifier
```

**Parameters**

| | |
|---|---|
| identifier | Searched for in the compiler's table of #defines.<br>- If identifier is found, the source text following the #ifndef directive until the next #else, #elif, or #endif is ignored.<br>- If identifier is not found, the source text following the #ifdef directive until the next #else, #elif, or #endif is processed. |

**Remarks**

#ifndef is a directive that marks the start of conditional compilation.

The #ifndef keyword is case sensitive; #IFNDEF will cause a compile error.

**See Also**

#define
#elif
#else
#endif
#ifdef
Examples

## #include

```
#include "filename"
#include <filename>
```

**Parameters**

filename — Can be absolute, relative to the current directory, or in one of the directories identified by the INCLUDE environment variable. Must be surrounded by either a pair of double-quote characters ("") or a left angle bracket (<) and a right angle bracket (>).

Resource Workshop treats both forms identically to maintain compatibility with the Microsoft RC compiler. Also to maintain compatibility, paired backslash characters ("\\") are treated as a single backslash.

**Remarks**

#include is a directive that causes the Resource Workshop compiler to open the named source file and process directives and resource definitions contained in that file.

Only directives of the format #... are processed in a header or C source file that is #included. All other data (for example, comments, structure definitions, code, resource definitions) within #included files are ignored.

This prevents Resource Workshop from seeing or processing RCINCLUDE directives in an #included file in header and C source files.

You can also #include Pascal constants in a Pascal include file or in a unit. The file must not contain any data other than constants.

The #include keyword is case sensitive; #INCLUDE will cause a compile error.

**See Also**

Examples
RCINCLUDE

## #line

```
#line integer_constant
```

**Remarks**
#line is a directive that causes the Resource Workshop compiler to ignore text on the specified line.

## #pragma

The #pragma directive is ignored by the Resource Workshop compiler.

# ACCELERATOR

```
resource-name ACCELERATOR
BEGIN
  keystroke, acc-ID, [keystroke-type] [modifier key] [NOINVERT]
END
```

**Parameters**

| | |
|---|---|
| resource-name | Text identifier or numeric ID for this resource. The name or number must be unique within the ACCELERATOR resource type. Numeric IDs must be positive integers. |
| keystroke | Value of the character code that activates this accelerator. The field can be either a character in double quotes or a numeric value.<br>- If a character in quotes is used, it can be preceded by a carat (^) to indicate that it is a control character (for example "^a" indicates Ctrl-A).<br>- If a numeric value is used, it is either the ASCII value of the accelerator key or the value for a virtual key, depending on the keystroke-type. |
| acc-ID | User-assigned integer value that identifies the string. Uniqueness is not required of IDs: Two different key combinations can be used to invoke the same command. The ID is sent in the wParam of a WM_COMMAND message. The high-order word of the lParam of the WM_COMMAND message is set to 1 to indicate the message resulted from an accelerator key. |
| keystroke-type | Valid only if the keystroke field contains a numeric value. In such cases, this field indicates whether the keystroke number is an ASCII value or virtual key value as follows:<br>ASCII Keystroke field is ASCII value.<br>VIRTKEY Keystroke field is virtual key value. |
| modifier key | Indicates if any modifier keys must be held down while typing the keystroke to activate the accelerator. If this field is missing, no modifier keys are required. Valid modifier keys are Shift, Alt, and Ctrl. The SHIFT and CTRL modifiers have no effect unless the VIRTKEY keystroke-type is used.<br>SHIFT Shift key must be held down.<br>CONTROL Ctrl key must be held down.<br>ALT Alt key must be held down. |
| NOINVERT | Disables highlighting of the menu title of the accelerated menu item when using accelerator keys for actions which have no menu item equivalent. If the NOINVERT field is missing, these menu titles are highlighted. |

**Remarks**

ACCELERATOR is a multiple-line statement that defines keyboard shortcuts for menu items and other program control actions.

The accelerator resource associates one or more accelerator keys with a corresponding accelerator command ID (acc-ID). Although acc-IDs must be numeric, the #define can be used to simplify access by the program.

**See Also**

#define
Examples

## BITMAP

```
resource-name BITMAP [load-type] [memory-option] filename
```

**Parameters**

| | |
|---|---|
| resource-name | Text identifier or numeric ID for this resource. The identifier or number must be unique within the BITMAP resource type. Numeric IDs must be positive integers. |
| load-type | Specifies when the resource is loaded into memory. |
| memory-option | Specifies how the resource is loaded into memory. |
| filename | The name of the DOS file containing the bitmap data. A relative or full path name can be used to specify files which are not in the current working directory. The data in the specified file is included in the current project. |

**Remarks**

BITMAP is a binary resource statement that associates a file containing bitmap resource data with a resource name and causes the bitmap data to be included in the current project.

You can write a BITMAP statement with the same syntax as RCDATA.

**See Also**

Examples
RCDATA

## CHECKBOX

```
CHECKBOX text, control-ID, x, y, width, height, [c-style]
```

**Parameters**

| | |
|---|---|
| text | Text string in double quotes. It is displayed next to the button. It can be specified as a keyboard accelerator mnemonic. |
| control-ID | A numeric identifier for this control. This number must be a unique short integer. Windows uses the control-ID to indicate which control has been selected. |
| x, y | Horizontal and vertical positions, respectively, of the button relative to the dialog window in which it appears. These numbers are specified in dialog units. |
| width, height | The width and height of the control (specified in dialog units). |
| c-style | The control style constant (an unsigned long integer value that is interpreted as a series of bit flags). Constants can be combined with a bitwise OR to apply multiple styles to the control. The field can contain these window style constants:<br>WS_DISABLED<br>WS_TABSTOP<br>WS_GROUP<br>and any of the BUTTON class style constants. The default styles are WS_TABSTOP and BS_CHECKBOX. |

**Remarks**

A DIALOG definition that places a check box control in a dialog box. It's valid only within a DIALOG definition.

A check box is a square button with descriptive text to the left or right of the button. Normally, text is displayed to the right of the button. You can change this to the left side with the appropriate c-style.

A check box control maintains a state: It is either checked (in which case, a large X fills the square) or unchecked. Check box controls do not change state automatically: The application programmer must change their state in response to BN_CLICKED notification messages. For that reason, it's a good idea to use the auto-check box control. The CONTROL definition must be used to specify an auto-check box.

A check box is a member of the button class. When the user presses the mouse button in the control's area, the button is highlighted. When the mouse button is released, it is returned to normal, and a message is sent to the parent window indicating that the button has been pressed.

**See Also**

DIALOG
Examples

## COMBOBOX

```
COMBOBOX ID, x, y, width, height, [c-style]
```

**Parameters**

| | |
|---|---|
| ID | Numeric identifier for this control. This number must be a unique short integer. Windows uses the control ID to indicate which control has been selected. |
| x, y | Integer values that specify the x- and y-coordinates of the upper left corner of the COMBOBOX control. These values are specified in dialog units. |
| width, height | Integer values that specify the size of the control. These values are specified in dialog units. |
| c-style | The control style constant (an unsigned long integer value that is interpreted as a series of bit flags). Constants can be combined with a bitwise OR to apply multiple styles to the control. The field can contain these window style constants:<br>WS_DISABLED<br>WS_GROUP<br>WS_TABSTOP<br>WS_VSCROLL<br>and any of the COMBOBOX class style constants. The default styles are WS_TABSTOP and CBS_SIMPLE. |

**Remarks**

A DIALOG definition that creates a combo box, a combination of either a static text field or an edit field with a list box.

If a static text field is used, this field displays the current selection in the list box. If an edit field is used, the selection is typed in and the list box highlights the first item that matches the typed entry.

**See Also**

Examples

# CONTROL

```
CONTROL text, control-ID, c-class, c-style, x, y, width, height
```

**Parameters**

| | |
|---|---|
| text | Text string in double quotes. This text is displayed, if appropriate, for the type of control specified. It can be specified as a keyboard accelerator mnemonic. |
| control-ID | A numeric identifier. This number can be a unique short integer, although unreferenced static controls are usually given a control ID of -1 to document that they are truly static. Windows uses the control ID to indicate which control has been selected. |
| c-class | A byte or a string indicating the control's class. The standard classes have predefined constants that can be used to identify them: c-class must be one of these contants:<br>BUTTON<br>COMBOBOX<br>EDIT<br>LISTBOX<br>SCROLLBAR<br>STATIC |
| c-style | The control style constant (an unsigned long integer value that is interpreted as a series of bit flags). Constants can be combined with a bitwise OR to apply multiple styles to the control. The options available to you depend on the c-class value. There is no default style. The high-order word of this field is reserved for Windows. The low-order word is available for user-defined control styles. |
| x, y | Horizontal and vertical positions of the control relative to the dialog window in which it appears. These numbers are specified in dialog units. |
| width, height | The width and height of the control. These numbers are specified in dialog units. |

**Remarks**

CONTROL is a DIALOG definition that places any type of control in a dialog box. It is valid only within a DIALOG definition and can duplicate the function of these definitions:

CHECKBOX
COMBOBOX
CTEXT
DEFPUSHBUTTON
EDITTEXT
GROUPBOX
ICON
LISTBOX
LTEXT
PUSHBUTTON
RADIOBUTTON
RTEXT
SCROLLBAR

**See Also**

DIALOG
Examples

## CTEXT

```
CTEXT text, control-ID, x, y, width, height, [c-style]
```

**Parameters**

| | |
|---|---|
| text | A text string, enclosed in quotes, that appears in the dialog window. It can be specified as a keyboard accelerator mnemonic.<br><br>A common Windows programming technique is to label an edit control by preceding it with a static text control. This label contains a mnemonic for the edit field, such as "&Name". If the CTEXT control doesn't use the WS_TABSTOP control style, Windows will set the focus in the next control. |
| control-ID | A numeric identifier. The control ID is used by Windows and an application to identify the control within the dialog. If there is no need to refer to the control at run time (as is most often the case with static controls), usually this field is set to -1 to document that it is truly static. |
| x, y | Horizontal and vertical positions of the text relative to the dialog window in which the text appears. These numbers are specified in dialog units. |
| width, height | The size of the control. These numbers are specified in dialog units. The static text appears centered within the specified area. If the text is too wide to fit on a single line, Windows automatically word wraps the text to multiple lines. |
| c-style | The control style constant (an unsigned long integer value that is interpreted as a series of bit flags). More than one control style can be combined using a bitwise OR. Defaults are the STATIC class style constant SS_CENTER and the window style constant WS_GROUP. |

**Remarks**

CTEXT is a DIALOG definition that specifies a text string, its attributes, and where it's located in the dialog box. CTEXT defines centered static text in a dialog window. The text is centered within the specified rectangle. If the text is too wide to fit within the rectangle, it is automatically wrapped.

CTEXT is valid only in a DIALOG definition.

**See Also**

DIALOG
Examples
LTEXT
RTEXT

## CURSOR

```
resource-name CURSOR [load-type] [memory-option] filename
```

**Parameters**

| | |
|---|---|
| resource-name | Text identifier or numeric ID for this resource. The identifier or number must be unique within the CURSOR resource type. Numeric IDs must be positive integers. |
| load-type | Specifies when the resource is loaded into memory. |
| memory-option | Specifies how the resource is loaded into memory. |

**Remarks**

CURSOR is a binary resource statement that causes cursor data to be included in the current project.

**See Also**

Examples

## DEFPUSHBUTTON

```
DEFPUSHBUTTON text, control-ID, x, y, width, height, [c-style]
```

**Parameters**

| | |
|---|---|
| text | A text string in double quotes. This text is displayed inside the button area. It can be specified as a keyboard accelerator mnemonic. |
| control-ID | A numeric identifier for this control. It must be a unique integer. The control ID is used by Windows to indicate which control has been selected. By convention, only one DEFPUSHBUTTON is included in a dialog. It is given the control ID IDOK (1).<br><br>When a user presses the Enter key in a modal dialog box, Windows sends a WM_COMMAND message with wParam set to IDOK. Also, when the user presses the Esc key, Windows sends a WM_COMMAND with wParam set to IDCANCEL (2). |
| x, y | Horizontal and vertical positions of the button relative to the dialog window in which it appears. These numbers are specified in dialog units. |
| width, height | The width and height of the control. These numbers are specified in dialog units. |
| c-style | The control style constant (an unsigned long integer value that is interpreted as a series of bit flags). Constants can be combined with a bitwise OR to apply multiple styles to the control. The field can contain these window style constants:<br>WS_DISABLED<br>WS_GROUP<br>WS_TABSTOP<br>and any of the BUTTON class style constants. The default styles are WS_TABSTOP and BS_DEFPUSHBUTTON. |

**Remarks**

DEFPUSHBUTTON is a DIALOG definition that places a default push button control in a dialog box.

A DEFPUSHBUTTON is a square button containing text describing its action. This control is almost identical to a normal push button, except it has a heavy border that indicates to the user it is the default action for this dialog window.

DEFPUSHBUTTON is valid only within a DIALOG definition. A DEFPUSHBUTTON is a member of the BUTTON class.

**See Also**

DIALOG
Examples

## DIALOG

```
resource-name DIALOG [load-type] [memory-option] x, y, width, height
[STYLE w-style] [CAPTION w-cap]
[MENU res-name] [CLASS w-class]
[FONT f-spec]
BEGIN
  dialog-controls
END
```

**Parameters**

| | |
|---|---|
| resource-name | Text identifier or numeric ID for this resource. The identifier or number must be unique within the DIALOG resource type. Numeric IDs must be positive integers. |
| load-type | Specifies when the resource is loaded into memory. |
| memory-option | Specifies how the resource is loaded into memory. |
| x, y | Horizontal and vertical positions of the upper left corner of the dialog window's client area. These coordinates can either be relative to the window's parent or owner window, or relative to the origin of the screen. This is determined by the window's style setting. Dialog windows are positioned relative to their parent or owner window unless the dialog window style constant DS_ABSALIGN is used. |
| width, height | The size of the client area of the window in dialog units. |
| STYLE | The window style of the dialog box. The window style specifies whether the box is a pop-up or a child window. |
| CAPTION | The title of the dialog box. The title appears in the box's caption bar (if it has one). The default caption is empty. |
| MENU | The dialog box's menu. If no statement is given, the dialog box has no menu. |
| CLASS | The class of the dialog box. If no statement is given, the Windows standard dialog class will be used as the default. |
| FONT | The font with which Windows will draw text in the dialog box. The font must have been previously loaded, either from the WIN.INI file or by calling the LoadResource function. |
| dialog-controls | Definition(s) that specify the content of the dialog windows. This includes static and editable text, various boxes and buttons, controls, and icons. Here are the dialog control statements:<br>CHECKBOX<br>COMBOBOX<br>CONTROL<br>CTEXT<br>DEFPUSHBUTTON<br>EDITTEXT<br>GROUPBOX<br>ICON<br>LISTBOX<br>LTEXT<br>PUSHBUTTON |

RADIOBUTTON
RTEXT
SCROLLBAR

**Remarks**
DIALOG is a multiple-line statement that specifies a dialog window.

A dialog window includes the window style, class, size, location, and the controls which will appear in the window. Dialog windows can contain text, check boxes, various buttons, icons, controls, list boxes, and so on.

**See Also**
Examples

## STYLE (DIALOG statement)

Statement in the DIALOG definition that defines the window style of the dialog box. The window style specifies whether the box is a pop-up or a child window. If you don't specify a style, the default values WS_POPUP, WS_BORDER, and WS_SYSMENU are used.

**Parameter**

| | |
|---|---|
| w-style | The window style constant or dialog window style constant. w-style is an integer value or a predefined name. |

## CAPTION (DIALOG statement)

Statement in the DIALOG definition that defines the dialog box title. The title appears in the box's caption bar (if it has one). The default caption is empty.

If CAPTION is not present in the DIALOG definition, the caption bar is empty.

**Parameter**

| | |
|---|---|
| w-cap | An ASCII character string enclosed in double quotes. |

## MENU (DIALOG statement)

Statement in the DIALOG definition that associates a menu resource with the dialog.

If MENU is not present in the DIALOG definition, no menu is associated with the dialog box.

**Parameter**

| | |
|---|---|
| res-name | The resource identifier or numeric ID of the associated menu. |

## CLASS (DIALOG statement)

Statement in the DIALOG definition that overrides the normal processing of a dialog box. It converts a dialog box to a window of the specified class; and depending on the class, could give undesirable results. Do not use the predefined control-class names with this statement.

Using a custom dialog class provides additional control over the behavior of the dialog window. In order to create a custom dialog class, you must set the cbWndExtra field of the WNDCLASS structure to at least as many bytes as used by DLGWINDOWEXTRA, the default dialog class. However, the additional control provided by the custom class is illusory because much of the Windows dialog manager is implemented in the IsDialogMessage function, not the dialog window procedure.

If no CLASS statement is given, the Windows standard dialog class will be used as the default.

**Parameter**

w-class An integer or text string that specifies the desired window class.

## FONT (DIALOG statement)

Statement in the DIALOG definition that defines the font with which Windows will draw text in the dialog box. The font must have been previously loaded, either from the WIN.INI file or by calling the LoadResource function.

**Parameter**

f-spec — The font specification. It consists of a point size (in points) followed by a font typeface string in double quotation marks (for example, 12, "Helv").

Windows uses the bold (approximately 700) weight for the font when this field is used. To use a lighter-weight attribute, use the WM_SETFONT message at run time to set the font.

## EDITTEXT

```
EDITTEXT control-ID, x, y, width, height, [c-style]
```

**Parameters**

| | |
|---|---|
| control-ID | Numeric identifier for this control. This number must be a unique integer. The control ID is used by Windows to indicate which control has been selected. |
| x, y | Horizontal and vertical positions of the edit text control relative to the dialog window in which it appears. These numbers are specified in dialog units. |
| width, height | The width and height of the edit text control. These numbers are specified in dialog units. |
| c-style | The control style constant (an unsigned long integer value that is interpreted as a series of bit flags). Constants can be combined with a bitwise OR to apply multiple styles to the control. The field may contain the window style constants:<br>WS_DISABLED<br>WS_GROUP<br>WS_HSCROLL<br>WS_TABSTOP<br>WS_VSCROLL<br>and any of the EDIT class style constants. The default styles are WS_BORDER, WS_TABSTOP, and ES_LEFT. |

**Remarks**

EDITTEXT is a DIALOG definition that places an editable text field in a dialog box.

Text can be edited using the mouse, cursor keys, Backspace, and Del. An edit text control belongs to the EDIT class.

EDITTEXT is valid only within a  definition.

**See Also**

DIALOG
Examples

## FONT

```
resource-num FONT [load-type] [memory-option] filename
```

**Parameters**

| | |
|---|---|
| resource-num | The resource number. It can be either a unique name (text) or an integer value that is unique within the font resource type. This number identifies the font; you can't use resource names for fonts. |
| load-type | Specifies when the resource is loaded into memory. |
| memory-option | Specifies how the resource is loaded into memory. |
| filename | The name of the DOS file containing the font data. A full path name can be used to specify files which are not in the current working directory. The data in the specified file is included in the current project. |

**Remarks**

FONT is a binary resource statement definition that associates a file containing font resource data with a resource number. It also causes the font data to be included in the current project.

**See Also**

Examples

# GROUPBOX

```
GROUPBOX text, control-ID, x, y, width, height, [c-style]
```

**Parameters**

| | |
|---|---|
| text | A text string in double quotes. This text is displayed at the top of the group box. It can be specified as a keyboard accelerator mnemonic. |
| control-ID | A numeric identifier for this control. This number can be a unique integer, but is often -1. The control ID is used by Windows to indicate which control has been selected. |
| x, y | Horizontal and vertical positions of the group box relative to the dialog window in which it appears. These numbers are specified in dialog units. |
| width, height | The width and height of the group box. These numbers are specified in dialog units. |
| c-style | The control style constants (an unsigned long integer value that is interpreted as a series of bit flags). Constants can be combined with a bitwise OR to apply multiple styles to the control. The field may contain these window style constants:<br>WS_DISABLED<br>WS_TABSTOP<br>and any of the BUTTON class style constants. The default styles are WS_TABSTOP and BS_GROUPBOX. |

**Remarks**

GROUPBOX is a DIALOG definition that places a static group box in a dialog box.

You usually use a group box to visually group a number of related controls. The user can then easily see which controls are related. The text field of the group box definition usually describes the group. A group box consists of a rectangle of the specified size, with a text title overwriting the left part of the top line of the rectangle.

GROUPBOX is valid only within a DIALOG definition.

**See Also**

DIALOG
Examples

# ICON

For a type 1 icon definition:

```
resource-name ICON [load-type] [memory-option] filename
```

For a type 2 icon definition:

```
ICON resource-name, control-ID, x, y, width, height, [c-style]
```

**Parameters**

For a type 1 icon definition:

| | |
|---|---|
| resource-name | Text identifier or numeric ID for this resource. The identifier or number must be unique within the ICON resource type. Numeric IDs must be integers. |
| load-type | Specifies when the resource is loaded into memory. |
| memory-option | Specifies how the resource is loaded into memory. |
| filename | The name of the DOS file containing the icon data. A full path name can be used to specify files which are not in the current working directory. The data in the specified file is included in the current project. |

For a type 2 icon definition:

| | |
|---|---|
| resource-name | The identifier of the icon resource to be included in the dialog window. This resource identifier is assigned to the icon using a type 1 icon definition. Note that this is not the icon's file name. If the resource name is a text identifier, it must be enclosed in quotes. |
| control-ID | A numeric identifier for this control. This number must be a unique integer. The control ID is used by Windows to indicate which control has been selected. |
| x, y | Horizontal and vertical positions of the icon relative to the dialog window in which the icon appears. These numbers are specified in dialog units. |
| width, height | Ignored. The icon is automatically sized. |
| c-style | The control style constant (an unsigned long integer value that is interpreted as a series of bit flags). Constants can be combined with a bitwise OR to apply multiple styles to the control. The only permissible style is the STATIC class style constant, SS_ICON. |

**Remarks**

There are two types of icon definitions:

| | |
|---|---|
| type 1 icon | A binary resource statement that associates a file containing icon resource data with a resource name. |
| type 2 icon | A DIALOG definition that specifies a static icon control in a dialog box. |

The type 1 icon definition tells Resource Workshop in which file an icon's data is contained, and gives this icon the specified resource-name. Data from the indicated file is included in the executable file when the application is built.

The type 2 icon definition refers to an icon that was specified with a type 1 definition (or defined in a free-form resource). Type 2 icon definitions place static icon controls within a dialog window. This type of ICON definition is only valid within a DIALOG definition.

**See Also**

Examples

# LISTBOX

```
LISTBOX control-ID, x, y, width, height, [c-style]
```

**Parameters**

| | |
|---|---|
| control-ID | Numeric identifier for this control. This number must be a unique integer. The control ID is used by Windows to indicate which control has been selected. |
| x, y | Horizontal and vertical positions of the list box relative to the dialog window in which it appears. These numbers are specified in dialog units. |
| width, height | The width and height of the control. These numbers are specified in dialog units. |
| c-style | The control style constant (an unsigned long integer value that is interpreted as a series of bit flags). Constants can be combined with a bitwise OR to apply multiple styles to the control. The field may contain these window style constants:<br>WS_BORDER<br>WS_DISABLED<br>WS_VSCROLL<br>and any of the LISTBOX class style constants. Default values are WS_VSCROLL, WS_BORDER, and LBS_NOTIFY. |

**Remarks**

LISTBOX is a DIALOG definition that places a list box control in a dialog box.

A list box is an area where multiple text strings are displayed. A list box is a member of the list box class.

LISTBOX is valid only within a DIALOG definition.

**See Also**

DIALOG
Examples

## LTEXT

```
LTEXT text, control-ID, x, y, width, height, [c-style]
```

**Parameters**

| | |
|---|---|
| text | Text string, enclosed in quotes, that appears in the dialog window. It can be specified as a keyboard accelerator mnemonic.<br><br>A common Windows programming technique is to label an edit control by preceding it with a static text control. The static label contains a mnemonic for the edit field, such as "&Name". If the LTEXT control does not use the window style constant WS_TABSTOP, Windows sets the focus in the next control. |
| control-ID | A numeric identifier for this control. This number must be a unique integer. The control ID is used by Windows to indicate which control has been selected. If there is no need to refer to the control at run time (as is most often the case with static controls), this field is by convention set to -1 to document that it is truly static. |
| x, y | Horizontal and vertical positions of the text relative to the dialog window in which the text appears. These numbers are specified in dialog units. |
| width, height | The size of the control in dialog units. The static text appears centered within the specified area. |
| c-style | The control style constant (an unsigned long integer value that is interpreted as a series of bit flags). More than one control style can be combined using a bitwise OR. The default style is the STATIC class style constants SS_LEFT and the window style WS_GROUP. The WS_GROUP style is always set for static text. |

**Remarks**

LTEXT is a DIALOG definition that defines left-justified static text in a dialog box.

LTEXT specifies a text string, its attributes, and where it is located in the dialog window. The text is aligned flush left within the specified rectangle. If the text is too wide to fit within the rectangle, it automatically wraps.

This definition can only appear within a DIALOG definition.

**See Also**

CTEXT
DIALOG
Examples
RTEXT

## MENU

```
resource-name MENU [load-type] [memory-option]
BEGIN
  item-definitions
END
```

**Parameters**

| | |
|---|---|
| resource-name | Text identifier or numeric ID for this resource. The identifier or number must be unique within the MENU resource type. Numeric IDs must be integers. |
| load-type | Specifies when the resource is loaded into memory. |
| memory-option | Specifies how the resource is loaded into memory. |
| item-definitions | Contain one or more MENUITEM or POPUP definitions. The MENUITEM definition specifies individual menu items, and the POPUP definition describes a pop-up menu (also known as a drop-down menu). |

**Remarks**

MENU is a multiple-line statement that defines a menu resource and specifies which menu items appear on this menu.

A MENU definition contains the item definitions, MENUITEM and POPUP.

**See Also**

Examples

# MENUITEM

```
MENUITEM [item-text] [item-ID] [item-attributes]
```
or
```
MENUITEM SEPARATOR
```

**Parameters**

item-text — Describes the menu selection. It is a character string enclosed in double quotes. The item-text also indicates which character in the menu item serves as the menu mnemonic.

The item text string can also include the following escape codes: \a and \t. The \a escape code aligns the following text flush right. The \t escape code inserts a tab in the item text aligning the text. It should only be used within a POPUP definition and not for items in the main menu bar.

item-ID — An integer value that identifies the menu item. Windows uses the value to indicate which menu item has been selected.

item-attributes — Accepts one or more keywords that describe the menu item's state (active, inactive, grayed, checked, etc.) The keywords GRAYED and INACTIVE are mutually exclusive. Other keywords can be used together by combining their values with a bitwise OR.

| | |
|---|---|
| GRAYED | Item is unavailable for selection. |
| INACTIVE | Item is never available for selection. |
| CHECKED | Checkmark appears next to item. Not supported for top-level menu items. |
| HELP | Appears on right side of menu bar. |
| MENUBREAK | Item begins a new menu column. |
| MENUBARBREAK | Item begins a new menu row. |

SEPARATOR — Indicates that this menu item is a separator line, rather than a selectable item. Separator lines are horizontal lines separating menu items when used in pop-up menus, and vertical lines separating menu titles when used in the menu bar.

**Remarks**

A MENU definition that defines a single menu item or a menu separator.

The MENUITEM definition associates an item ID with a specific menu element. MENUITEM can be used only within a MENU or POPUP definition.

**See Also**

Examples
MENU

## POPUP

```
POPUP [popup-name] [popup-attributes]
BEGIN
  item-definitions
END
```

**Parameters**

popup-name — The name of the pop-up menu. This name is typically shown in the menu bar. It is a character string in double quotes. The pop-up name also indicates which character in the menu item is to serve as the menu mnemonic.

The item text string can also include the following escape codes: \a and \t. The \a escape code aligns the following text flush right. The \t escape code inserts a tab in the item text aligning the text. It should be used only within a POPUP definition and not for items in the main menu bar.

popup-attributes — One or more keywords that describe the pop-up menu's state (active, inactive, grayed, checked, etc.).The keywords GRAYED and INACTIVE are mutually exclusive. Other keywords can be used together by combining their values with a bitwise OR.

| | |
|---|---|
| GRAYED | Item is unavailable for selection. |
| INACTIVE | Item is never available for selection. |
| CHECKED | Checkmark appears next to item. Not supported for top-level menu items. |
| HELP | Appears on right side of menu bar. |
| MENUBREAK | Item begins a new menu column. |
| MENUBARBREAK | Item begins a new menu row. |

item-definitions — One or more MENUITEM or POPUP definitions. The MENUITEM definition specifies individual menu items, and the POPUP definition describes a pop-up menu.

**Remarks**

POPUP is a MENU definition that defines a pop-up menu (also known as a drop-down menu).

POPUP contains a number of MENUITEM definitions, which specify the individual items in the pop-up menu. POPUP can be used only within a MENU definition.

**See Also**

Examples
MENU

## PUSHBUTTON

```
PUSHBUTTON text, control-ID, x, y, width, height, [c-style]
```

**Parameters**

| | |
|---|---|
| text | Text string in double quotes. This text is displayed inside the button area. It can be specified as a keyboard accelerator mnemonic. |
| control-ID | A numeric identifier for this control. This number must be a unique integer. The control ID is used by Windows to indicate which control has been selected. |
| x, y | Horizontal and vertical positions of the button relative to the dialog window in which it appears. These numbers are specified in dialog units. |
| width, height | The width and height of the control. These numbers are specified in dialog units. |
| c-style | The control style constant (an unsigned long integer value that is interpreted as a series of bit flags). Constants can be combined with a bitwise OR to apply multiple styles to the control. The field can contain any of the window style constants.<br>WS_TABSTOP<br>WS_DISABLED<br>WS_GROUP<br>and any of the BUTTON class style constants. The default styles are WS_TABSTOP and BS_PUSHBUTTON. |

**Remarks**

PUSHBUTTON is a DIALOG definition that places a push button control in a dialog box.

A push button is a square button containing text describing its action. It is a member of the button class. PUSHBUTTON is valid only within a DIALOG definition.

**See Also**

DIALOG
Examples

# RADIOBUTTON

```
RADIOBUTTON text, control-ID, x, y, width, height, [c-style]
```

**Parameters**

| | |
|---|---|
| text | Text string in double quotes. This text is displayed next to the button. It can be specified as a keyboard accelerator mnemonic. |
| control-ID | A numeric identifier for this control. This number must be a unique integer. The control ID is used by Windows to indicate which control has been selected. |
| x, y | Horizontal and vertical positions of the button relative to the dialog window in which it appears. These numbers are specified in dialog units. |
| width, height | The width and height of the control. These numbers are specified in dialog units. |
| c-style | The control style constant (an unsigned long integer value that is interpreted as a series of bit flags). Constants can be combined with a bitwise OR to apply multiple styles to the control. The field may contain the window style constants:<br>WS_DISABLED<br>WS_GROUP<br>WS_TABSTOP<br>and any of the BUTTON class style constants.. The default styles are WS_TABSTOP and BS_RADIOBUTTON. |

**Remarks**

RADIOBUTTON is a DIALOG definition that places a radio button control in a dialog box.

A radio button is a round button with text to the left or right of the button describing the button. Normally, text is displayed to the right of the button. You can change this to the left side with the appropriate c-style.

Radio buttons maintain a state: they are either checked or unchecked. A checked radio button has a small black circle within the outer circle to show that it is checked. Radio buttons require application program intervention to maintain their state. To simplify programming, use the auto-radio button which is defined with the CONTROL definition.

A radio button is a member of the button class. When the user presses the mouse button in the control's area, the button is highlighted. When the mouse button is released, the radio button is returns to normal, and a message is sent to the parent window indicating that the button was pressed.

Never perform any action that changes the keyboard focus (or for that matter any action which toggles the state of the application) in response to the BN_CLICKED message, since a radio button sends this notification each time it receives the keyboard focus.

RADIOBUTTON is valid only within a DIALOG definition.

**See Also**

DIALOG
Examples

## RCDATA

```
resource-name RCDATA [load-type] [memory-option]
BEGIN
  resource-data
END
```

**Parameters**

| | |
|---|---|
| resource-name | The text identifier or numeric ID for this resource. The identifier or number must be unique. Numeric IDs must be integers. |
| load-type | Specifies when the resource is loaded into memory. |
| memory-option | Specifies how the resource is loaded into memory. |
| resource-data | One or more lines of data in standard C language format. Data can consist of any mix of numeric values and strings. Numeric values can be represented in hex, octal, or decimal. Strings are placed inside double quotation marks. String values are not automatically null terminated. To terminate a string with a null character, simply include a \0 at the end of the string. |

**Remarks**

RCDATA is a multiple-line statement that lets the user include any type of data directly in an .RC file.

The resource data is associated with the specified resource name, and is included in the executable file for access at run time.

**See Also**

Examples

## RCINCLUDE

```
rcinclude filename
```

**Parameters**

| | |
|---|---|
| filename | Can be absolute or relative to the current directory. The INCLUDE environment path is not searched for files which are rcincluded. |

**Remarks**

RCINCLUDE is a directive that causes Resource Workshop to open the named source file and process directives and resource definitions contained in that file.

RCINCLUDE directives can be nested up to 19 levels. Note, however, that the Microsoft Resource Compiler does not support nested RCINCLUDE statements.

The rcinclude keyword is not case sensitive: rcinclude, RCINCLUDE, and RcInClUde are all processed identically by Resource Workshop.

**See Also**

Examples

## RTEXT

```
RTEXT text, control-ID, x, y, width, height, [c-style]
```

**Parameters**

| | |
|---|---|
| text | A text string, enclosed in quotes, that appears in the dialog window. It can be specified as a keyboard accelerator mnemonic.<br><br>A common Windows programming technique is to label an edit control by preceding it with a static text control. The label contains a mnemonic for the edit field, such as "&Name". If the CTEXT control does not use the window style constant WS_TABSTOP, Windows sets the focus in the next control. |
| control-ID | A numeric identifier for this control. This number can be a unique integer. The control ID is used by Windows to indicate which control has been selected. If there is no need to refer to the control at run time (as is most often the case with static controls), this field is by convention set to -1 to document that it is truly static. |
| x, y | Horizontal and vertical positions of the text relative to the dialog window in which the text appears. These numbers are specified in dialog units. |
| width, height | The size of the control in dialog units. The static text appears centered within the specified area. |
| c-style | The control style constant (an unsigned long integer value that is interpreted as a series of bit flags). More than one control style can be combined using a bitwise OR. The window style WS_GROUP is always set for static text. The default style is the STATIC style constant SS_RIGHT and the window style WS_GROUP. |

**Remarks**

RTEXT is a DIALOG definition that defines right-justified static text in a dialog box.

RTEXT specifies a text string, its attributes, and where it is located in the window. The text is aligned flush right within the specified rectangle. If the text is too wide to fit within the rectangle, it automatically wraps.

This definition can only appear within a DIALOG definition.

**See Also**

CTEXT
DIALOG
Examples
LTEXT

## SCROLLBAR

```
SCROLLBAR control-ID, x, y, width, height, [c-style]
```

**Parameters**

| | |
|---|---|
| control-ID | A numeric identifier for this control. This number must be a unique identifier. The control-ID is used by Windows to indicate which control has been selected. |
| x, y | Horizontal and vertical position of the scroll bar control relative to the dialog window in which it appears. These numbers are specified in dialog units. |
| width, height | The size of the control in dialog units. |
| c-style | The control style constant (an unsigned long integer value that is interpreted as a series of bit flags). Constants may be combined with a bitwise OR to apply multiple styles to the control. The field can combine any of the window style constants:<br>WS_DISABLED<br>WS_GROUP<br>WS_TABSTOP<br>and any of the SCROLLBAR class style constants. The default style is SBS_HORZ. |

**Remarks**

SCROLLBAR is a DIALOG definition that defines a scroll bar, a rectangular control with direction arrows on both ends and a movable scroll box.

A scroll bar can appear anywhere in a window. When the user clicks the control with a mouse, a message is sent to the parent window.

**See Also**

Examples

## STRINGTABLE

```
STRINGTABLE [load-type] [memory-option]
BEGIN
  string-ID, string
END
```

**Parameters**

| | |
|---|---|
| load-type | Specifies when the resource is loaded into memory. |
| memory-option | Specifies how the resource is loaded into memory. |
| string-ID | A user-assigned integer value that identifies the string. Each string ID must be unique. The string ID is used at run time by the LoadString function to determine which string is being requested by the program. |
| string | An ASCII string in standard C language format. |

**Remarks**

STRINGTABLE is a multiple-line statement that specifies null-terminated ASCII strings that can be accessed by the program.

Each string is assigned a unique unsigned short integer string ID. Strings are read in for access at run time by calling the LoadString function with the desired string ID.

The string table mechanism is a convenient method to keep text strings separate from code for easy update and possible translation into foreign languages.

Although string IDs must be numeric, the #define preprocessor directive can be used to simplify access by the program. (See the second example.)

**See Also**

Examples

## User-defined resources

```
resource-name type-ID [load-type] [memory-option] filename
```

**Parameters**

| | |
|---|---|
| resource-name | Text identifier or numeric ID for this resource. The identifier or number must be unique within each user-defined resource type. Numeric IDs must be integers. |
| type-ID | The type-identifier for the custom resource type. This type identifier should be a text identifier or an integer number. User-defined type identifier numbers must be greater than 255. (Numbers 1 to 255 are reserved by Resource Workshop for predefined resource types and future expansion.) |
| load-type | Specifies when the resource is loaded into memory. |
| memory-option | Specifies how the resource is loaded into memory. |
| filename | The name of the DOS file containing the user data. A full path name can be used to specify files which are not in the current working directory. The data in the specified file is included in the current project. |

**Remarks**

A user-defined resource definition associates a file containing user-defined resource data with a resource name and a type ID.

This definition includes data from the specified file in the current project. This definition can also use the syntax of RCDATA.

**See Also**

Examples

## VERSIONINFO resource

The VERSIONINFO resource is a version stamper for Windows 3.1 .EXE files. It is a collection of data used by several Windows API functions (i.e. GetFileVersionInfo, VerInstallFile, and so on) that are typically used by Windows-based installation programs.

**See Also**
VERSIONINFO

## VERSIONINFO

```
versionID VERSIONINFO fixed-info
BEGIN
  block-statements
END
```

**Parameters**

versionID — Version information resource identifier. Must be set to 1.

fixed-info — Version information that consists of:

| Statement | Description |
|---|---|
| FILEVERSION version | Binary version number for file. A 64-bit, integer number. |
| PRODUCTVERSION version | Binary version number for product that file is distributed with. A 64-bit, integer number. |
| FILEFLAGSMASK fileflags | Specifies valid bits in FILEFLAGS statement. If a bit is set, the corresponding bit in FILEFLAGS is valid. |
| FILEFLAGS fileflags | Boolean attributes of file. |
| FILEOS fileos | Operating system file designed for. |
| FILETYPE filetype | File type. |
| FILESUBTYPE subtype | File function. subtype is 0 unless type parameter in FILETYPE VFT_DRV, VFT_FONT, or VFT_VXD. |

block-statements — One or more version information blocks. Blocks are either string information or variable information.

String information block:

```
BLOCK "StringFileInfo"
BEGIN
    BLOCK "lang-charset"
    BEGIN
        VALUE "string-name", "value"
              .
              .
              .
    END
END
```

where:

| | |
|---|---|
| lang-charset | Hex string for language and character set. |
| string-name | Name of a value in the block. |
| VALUE | Character string specifying value of corresponding string-name. There can be more than one value statement. |

Variable information block:

```
BLOCK "VarFileInfo"
BEGIN
    VALUE "Translation",
      langID, charsetID
            .
            .
```

```
END
```

where:

| | |
|---|---|
| langID/charsetID | Language and character set identifier.There can be more than one identifier pair. Each pair must be separated from the preceding pair with a comma. |

**Remarks**

VERSIONINFO creates a version information resource. The resource contains the:

- file version number
- product version number
- operating system
- file type
- file function

This resource is intended for use with File Installation functions.

## fileflags

Parameter used in the VERSIONINFO statement. To use these constants, VER.H must be included in your resource-definition file.

| Value | Meaning |
|---|---|
| VS_FF_DEBUG | Debugging - file contains debugging information or was compiled with debugging features enabled. |
| VS_FF_INFOINFERRED | Incorrect version information - file contains a dynamically-created version information resource with possible empty or incorrect blocks. Do not use this value in version information resources created with the VERSIONINFO statement. |
| VS_FF_PATCHED | Patch - file has been modified; is not identical to the original shipping file of the same version number. |
| VS_FF_PRERELEASE | Pre-release - file is a development version, not a commercially released product. |
| VS_FF_PRIVATEBUILD | Private build - file was not built using standard release procedures. When you use this value, include the PrivateBuild string in the string-name parameter. |
| VS_FF_SPECIALBUILD | Special build - file was built by the original company using standard release procedures, but is a variation of the standard file of the same version number. When you use this value, include the SpecialBuild string in the string-name parameter. |

## fileos

Parameter used in the VERSIONINFO statement. To use these constants, VER.H must be included in your resource-definition file.

| Value | Meaning |
|---|---|
| VOS_UNKNOWN | File designed for unknown operating system. |
| VOS_DOS | File designed for MS-DOS. |
| VOS_NT | File designed for Windows NT. |
| VOS_WINDOWS16 | File designed for Windows 3.0 or later. |
| VOS_WINDOWS32 | File designed for Windows 32-bit. |
| VOS_DOS_WINDOWS16 | File designed for Windows 3.0 or later running on MS-DOS. |
| VOS_DOS_WINDOWS32 | File designed for Windows 32-bit running on MS-DOS. |
| VOS_NT_WINDOWS32 | File designed for Windows 32-bit running on Windows NT. |

The values 0x00002L, 0x00003L, 0x20000L and 0x30000L are reserved.

## filetype

Parameter used in the VERSIONINFO statement. To use these constants, VER.H must be included in your resource-definition file.

| Value | Meaning |
|---|---|
| VFT_UNKNOWN | File type is unknown to Windows. |
| VFT_APP | File contains an application. |
| VFT_DLL | File contains a dynamic-link library. |
| VFT_DRV | File contains a device driver. If you use this constant, include a more specific description of the driver in FILESUBTYPE. |
| VFT_FONT | File contains a font. If you use this constant, include a more specific description of the font file in FILESUBTYPE. |
| VFT_VXD | File contains a virtual device. If you use this constant, include a more specific description of the device in FILESUBTYPE. |
| VFT_STATIC_LIB | File contains a static-link library. |

All other values are reserved for use by Microsoft.

## subtype

Parameter used in the VERSIONINFO statement. To use these constants, VER.H must be included in your resource-definition file.

subtype can be one of the following values if FILETYPE specifies VFT_DRV:

| Value | Meaning |
|---|---|
| VFT2_UNKNOWN | Driver type is unknown to Windows. |
| VFT2_DRV_COMM | File contains a communications driver. |
| VFT2_DRV_PRINTER | File contains a printer driver. |
| VFT2_DRV_KEYBOARD | File contains a keyboard driver. |
| VFT2_DRV_LANGUAGE | File contains a language driver. |
| VFT2_DRV_DISPLAY | File contains a display driver. |
| VFT2_DRV_MOUSE | File contains a mouse driver. |
| VFT2_DRV_NETWORK | File contains a network driver. |
| VFT2_DRV_SYSTEM | File contains a system driver. |
| VFT2_DRV_INSTALLABLE | File contains an installable driver. |
| VFT2_DRV_SOUND | File contains a sound driver. |

subtype can be one of the following values if FILETYPE specifies VFT_FONT:

| Value | Meaning |
|---|---|
| VFT2_UNKNOWN | Font type is unknown to Windows. |
| VFT2_FONT_RASTER | File contains a raster font. |
| VFT2_FONT_VECTOR | File contains a vector font. |
| VFT2_FONT_TRUETYPE | File contains a TrueType font. |

subtype must be the virtual-device identifier included in the virtual device control block if FILETYPE specifies VFT_VXD.

All values not listed are reserved.

## langID and charsetID

Parameter used in the VERSIONINFO statement. To use these constants, VER.H must be included in your resource-definition file.

langID can be one of the following values:

| Value | Language |
|---|---|
| 0x0401 | Arabic |
| 0x0402 | Bulgarian |
| 0x0403 | Catalan |
| 0x0404 | Traditional Chinese |
| 0x0405 | Czech |
| 0x0406 | Danish |
| 0x0407 | German |
| 0x0408 | Greek |
| 0x0409 | U.S. English |
| 0x040A | Castilian Spanish |
| 0x040B | Finnish |
| 0x040C | French |
| 0x040D | Hebrew |
| 0x040E | Hungarian |
| 0x040F | Icelandic |
| 0x0410 | Italian |
| 0x0411 | Japanese |
| 0x0412 | Korean |
| 0x0413 | Dutch |
| 0x0414 | Norwegian - Bokmal |
| 0x0415 | Polish |
| 0x0416 | Brazilian Portuguese |
| 0x0417 | Rhaeto-Romanic |
| 0x0418 | Romanian |
| 0x0419 | Russian |
| 0x041A | Croato-Serbian (latin) |
| 0x041B | Slovak |
| 0x041C | Albanian |
| 0x041D | Swedish |
| 0x041E | Thai |
| 0x041F | Turkish |
| 0x0420 | Urdu |
| 0x0421 | Bahasa |
| 0x0804 | Simplified Chinese |
| 0x0807 | Swiss German |
| 0x0809 | U.K. English |
| 0x080A | Mexican Spanish |
| 0x080C | Belgian French |
| 0x0810 | Swiss Italian |
| 0x0813 | Belgian Dutch |
| 0x0814 | Norwegian - Nynorsk |
| 0x0816 | Portuguese |
| 0x081A | Serbo-Croatian (cyrillic) |
| 0x0c0C | Canadian French |
| 0x100C | Swiss French |

charsetID can be one of the following values:

| Value | Character set |
|---|---|
| 0 | 7-bit ASCII |
| 932 | Windows, Japan (Shift - JIS X-0208) |
| 949 | Windows, Korea (Shift - KSC 5601) |
| 950 | Windows, Taiwan (GB5) |
| 1200 | Unicode |
| 1250 | Windows, Latin-2 (Eastern European) |
| 1251 | Windows, Cyrillic |
| 1252 | Windows, Multilingual |
| 1253 | Windows, Greek |
| 1254 | Windows, Turkish |
| 1255 | Windows, Hebrew |
| 1256 | Windows, Arabic |

## string-name

Parameter used in the VERSIONINFO statement. To use these constants, VER.H must be included in your resource-definition file.

| Name | Value |
|---|---|
| Comments | Additional information for diagnostic purposes. Optional. |
| CompanyName | The company that produced the file. Required. |
| FileDescription | File description. You can display this string in a list box during installation. Required. |
| FileVersion | File version number. Required. |
| InternalName | File internal name. If file does not have internal name, use original filename, without extension. Required. |
| LegalCopyright | File copyright notices. Optional. |
| LegalTrademarks | Trademarks and registered trademarks that apply to file. Optional. |
| OriginalFilename | Original file name, not including path. Required. |
| PrivateBuild | Information about private version of file. Required if VS_FF_PRIVATEBUILD is set in FILEFLAGS. |
| ProductName | Name of product file is distributed with. Required. |
| ProductVersion | Version of product file is distributed with. Required. |
| SpecialBuild | Specifies how this version of file differs from standard version. Required if VS_FF_SPECIALBUILD is set in FILEFLAGS. |

## Constants

Here are the predefined constants:
Control window style constants
Dialog window style constants
Window style constants

## Window style constants

Here are the predefined constants for window styles.

| Style | Description |
|---|---|
| WS_BORDER | Window has a thin border (1 pixel wide on EGA and VGA displays). |
| WS_CAPTION | Window has a title bar and thin border, unless WS_EX__DLGMODALFRAME is set in CreateWindowEx (see DS_MODALFRAME). |
| WS_CHILD | Window is a child window. All controls within a dialog box have this style. This style cannot be used with WS_POPUP or WS_OVERLAPPED. |
| WS_CLIPCHILDREN | When the programmer obtains a device context for this window using either the GetDC or BeginPaint functions, the device context's clipping region excludes the area(s) occupied by any child window(s). If this style is not set for a window, it's possible for a parent to paint over one or more of its child windows.<br><br>This style is not normally applicable to dialog windows because the user program does not usually paint within a dialog window, leaving that function to the Windows dialog manager. |
| WS_CLIPSIBLINGS | When the programmer obtains a device context for this window using either the GetDC or BeginPaint functions, the device context's clipping region excludes the area(s) occupied by any sibling child window(s). This style requires the use of WS_CHILD. If this style is not set for a window, it is possible for a child to paint over one or more of its sibling windows.<br><br>This style is not normally applicable to dialog controls, because the user program does not usually paint within a dialog window, leaving that function to the Windows dialog manager. |
| WS_DISABLED | Window is disabled at the time of creation. No user input is allowed in the window, thus the control cannot be selected using the keyboard or mouse. For some controls, this style causes the text for the control to be grayed.<br><br>This style can be changed at run time by calling the EnableWindow function. |
| WS_DLGFRAME | Window has a double border and no title bar.<br><br>This style was standard for Windows 2 dialog windows. Windows 3 dialog windows, by convention, use the WS_CAPTION, WS_SYSMENU, and DS_MODALFRAME styles. The Windows 3 convention lets a user move the dialog window in order to see the information underneath. |
| WS_GROUP | This style is used to group controls for user access with the arrow keys. WS_GROUP is a start of group marker. Only the first control in each group should have the WS_GROUP flag set. The next group |

| | |
|---|---|
| | begins with the next WS_GROUP.<br><br>Because this style flag has the same numeric value as WS_MAXIMIZEBOX, it can be used only with child windows. |
| WS_HSCROLL | Window has a horizontal scroll bar along the bottom of the window.<br><br>The WS_HSCROLL style should not be confused with a scroll bar control. A scroll bar control is an independent child window which can be moved anywhere within its parent's client area, while the WS_HSCROLL creates a scroll bar that is part of the window's frame (also called non-client area). |
| WS_ICONIC | Window is created initially in the iconic state. This style should only be used for WS_OVERLAPPED windows. |
| WS_MAXIMIZE | Window is maximized when first shown. This style should only be used for WS_OVERLAPPED windows. |
| WS_MAXIMIZEBOX | Window contains a maximize button in the title bar. This style should only be used for windows with the WS_CAPTION style. |
| WS_MINIMIZE | Window is minimized when first shown. This style should only be used for WS_OVERLAPPED windows. |
| WS_MINIMIZEBOX | Window contains a minimize button in the title bar. This style should only be used for windows with the WS_CAPTION style. |
| WS_OVERLAPPED | Window is a top-level window. Usually used for the main window of an application. This style cannot be used with WS_POPUP, or WS_CHILD styles. |
| WS_OVERLAPPEDWINDOW | Window has WS_OVERLAPPED, WS_CAPTION, WS_SYSMENU, and WS_THICKFRAME attributes. |
| WS_POPUP | Window is the pop-up type. Cannot be used with WS_CHILD or WS_OVERLAPPED styles. |
| WS_POPUPWINDOW | Window has WS_POPUP, WS_BORDER, and WS_SYSMENU attributes. |
| WS_SIZEBOX | Window has a thick frame. Valid only for windows with scroll bars or a title bar. |
| WS_SYSMENU | Window has a system menu in the title bar. Valid only for windows with title bars (WS_CAPTION style). |
| WS_TABSTOP | This style indicates that when the user presses Tab in a preceding control, the input focus changes to this control. Controls without this style can still be activated using the mouse, or using the arrow keys, if part of a group. |
| WS_THICKFRAME | Window has a thick frame and can be sized by the user. |
| WS_VISIBLE | Window is visible when created. All controls within a dialog have this |

| | |
|---|---|
| | style by default. |
| WS_VSCROLL | Window contains a vertical scroll bar at its left side. The WS_VSCROLL style should not be confused with a scroll bar control. A scroll bar control is an independent child window, which can be moved anywhere within its parent's client area, while the WS_VSCROLL style creates a scroll bar that is part of the window's frame (also called non-client area). |

## Dialog window style constants

Here are the predefined constants for dialog window styles.

| Style | Description |
|---|---|
| DS_ABSALIGN | This style indicates that the x- and y-coordinates specified for the window's position are relative to the screen's origin rather than to the parent or owner window. |
| DS_SYSMODAL | Makes a system modal window. No other windows can be selected by the user when this window is displayed. |
| DS_LOCALEDIT | Storage for edit text controls (class "edit") is, by default, allocated in the global heap. By using DS_LOCALEDIT, you can force the Windows dialog manager to create the edit text controls in such a way that the storage is allocated on the local heap of the module whose instance handle is passed to the dialog manager in the call to DialogBox or CreateDialog.<br><br>If you want to use the EM_SETHANDLE or EM_GETHANDLE messages in the dialog hook or window function, you must use this style. |
| DS_SETFONT | This style must be set by a resource compiler for a dialog window that contains the optional font data in the dialog binary data structure. Because it is set by the resource compiler, it must never be used by a user. |
| DS_MODALFRAME | This style is used in conjunction with WS_CAPTION to create a movable but unsizeable window with a caption and a modal dialog frame for the lower three borders instead of a thin border. |
| DS_NOIDLEMSG | When a modal dialog box is displayed by an application, Windows periodically sends the dialog's owner window function WM_ENTERIDLE messages. If for some reason, a programmer wants to suppress these messages, use this style. |

## Control window style constants

Here are the predefined constants for control window styles.

BUTTON class style constants
COMBOBOX class style constants
EDIT class style constants
LISTBOX class style constants
SCROLLBAR class style constants
STATIC class style constants

## BUTTON class style constants

The BUTTON class includes controls that function like physical buttons. The user "presses" the control using the mouse or keyboard, and the control performs some specific action. Buttons include push buttons, radio buttons, check boxes, and so on.

The predefined constants for the BUTTON class style are:

| Style | Description |
|---|---|
| BS_3STATE | A checkbox with three states: checked, not checked, and grayed. When the user clicks on the control, Windows sends a BN_CLICKED message to the control's parent window indicating that the button has been clicked. |
| BS_AUTO3STATE | A checkbox with three states: checked, not checked, and grayed. When the user clicks the mouse button on the control, the button automatically toggles to the next state. Windows sends a BN_CLICKED message to the control's parent window indicating that the button has been clicked. |
| BS_AUTOCHECKBOX | A checkbox with two states: checked and not checked. When the user clicks the mouse button on the control, the button automatically toggles to the next state. Windows sends a BN_CLICKED message to the control's parent window indicating that the button has been clicked. |
| BS_AUTORADIOBUTTON | A radio button that toggles automatically. When the user clicks on the button, it is checked and other buttons in the group are unchecked. Windows notifies the control's parent window, indicating that the button has been clicked (BN_CLICKED). |
| BS_CHECKBOX | A checkbox with two states: checked and not checked. When the user clicks on the control, Windows sends a BN_CLICKED message to the control's parent window indicating that the button has been clicked. |
| BS_DEFPUSHBUTTON | Default push button. This style of control is a push button with a heavy border. The heavy border informs the user that this button is the default action for the window, when the Enter key is pressed. A BN_CLICKED message is sent to the parent window when the user clicks the mouse or presses Spacebar inside the button area.<br><br>Only one button of this type should be used in a dialog window, and you should be give it the control ID IDOK (1). |
| BS_LEFTTEXT | This style indicates that the control's text name should be displayed to the left of the button, rather than the default right-hand side.<br><br>This style can be used with check boxes, BS_3STATE and BS_AUTO3STATE check boxes, and radio buttons. |
| BS_OWNERDRAW | A button drawn by its owner (parent) window. |
| BS_PUSHBUTTON | Push button. This style of control is a rectangle containing text, with a rectangular border. A BN_CLICKED message is sent to the parent window when the user clicks the mouse or presses Spacebar inside the button area.<br><br>With some display drivers the control is partially filled with gray, giving the |

| | |
|---|---|
| | button a shaded appearance. |
| BS_RADIOBUTTON | Radio button. When the user clicks on the button, Windows notifies the control's parent window, indicating that the button has been clicked (BN_CLICKED). When using radio buttons, you must use the CheckRadioButton function to select and deselect radio buttons within a group when the BN_CLICKED notification is sent to the parent window or dialog function. |

## COMBOBOX class style constants

The COMBOBOX class is used for a box that is a combination of a list box and an edit control. The list box can be displayed at all times (CBS_SIMPLE) or dropped by clicking in a drop-down button next to the edit control. The edit control may or may not allow user entry.

The predefined constants for the COMBOBOX class are:

| Style | Description |
|---|---|
| CBS_AUTOHSCROLL | A combo box that scrolls the text in the edit control. When a user enters text beyond the boundary of the rectangle, the text will scroll. |
| CBS_DISABLENOSCROLL | A combo box with a disabled vertical scroll bar when the list box doesn't contain enough items to scroll. If this style is not used, the scroll bar won't be displayed. |
| CBS_DROPDOWN | A combo box with a list box displayed only when the user selects the drop-down arrow next to the edit control. The edit control displays the current selection. |
| CBS_DROPDOWNLIST | A combo box with a list box with a static text item instead of an edit control that displays the current selection. The list box is displayed only when the user selects the drop-down arrow next to the static text. |
| CBS_HASSTRINGS | An owner-draw combo box with a list box containing items made up of strings. Memory and pointers for the strings are handled by the combo box. Your application can then use CB_GETTEXT to retrieve the text for a particular string. |
| CBS_OEMCONVERT | A combo box that converts text typed into the edit control from the ANSI character set to the OEM character set and back again to ANSI. When your application calls the AnsiToOem function to convert an ANSI string to OEM characters, this style ensures characters are converted properly. Usually this style is used for combo boxes that contain filenames. Use CBS_OEMCONVERT only with combo boxes created with the CBS_SIMPLE or CBS_DROPDOWN styles. |
| CBS_OWNERDRAWFIXED | An owner-draw fixed-height combo box. The list box's owner draws the contents of the list box. Each list box item is the same height. |
| CBS_OWNERDRAWVARIABLE | An owner-draw variable-height combo box. The list box's owner draws the contents of the list box. List box items can be of variable heights. |
| CBS_SIMPLE | A combo box with the list box always displayed. It's edit control displays the list box's current selection. |
| CBS_SORT | A combo box that sorts all the items displayed in the list box. |

## EDIT class style constants

The EDIT class is used for editable text fields. These fields are a rectangular area with space for one or more lines of text. The text can be edited by the user with the mouse or keyboard.

The predefined constants for the EDIT class are:

| Style | Description |
|---|---|
| ES_AUTOHSCROLL | Automatic horizontal scroll. When this style is specified and the user enters more text than will fit in the control's rectangle, the text is automatically scrolled. Scrolling occurs 10 characters at a time. If automatic horizontal scrolling is not specified and ES_MULTILINE is specified, text will automatically wrap to the next line when the right side of the control's rectangle is reached.<br><br>The EM_LIMITTEXT message can be used by to place an absolute limit on the number of characters allowed in the edit text control. |
| ES_AUTOVSCROLL | Automatic vertical scroll. Text is scrolled vertically when the user presses the Enter key on the last line of text which fits in the control's rectangle. Scrolling occurs one page at a time. (A page is defined as the number of lines visible in the edit text control.) If automatic vertical scrolling is not specified, the control will beep if the user attempts to enter more lines than will fit in the control.<br><br>Note that the ES_LIMITTEXT message can be used by the programmer to place an absolute limit on the number of characters allowed in the edit text control. |
| ES_CENTER | Edit text is centered horizontally in control's rectangle.<br><br>The ES_MULTILINE style must be set for this style to have any effect on the appearance of the text. |
| ES_LEFT | Edit text is aligned flush left. |
| ES_LOWERCASE | Edit text is converted to lower case. |
| ES_MULTILINE | Edit text can occupy more than one line in the control's rectangle. The number of lines allowed by the control is determined by the size of the control's client area.<br><br>The ES_MULTILINE flag can be used with ES_AUTOVSCROLL and ES_AUTOHSCROLL to make an automatically scrolling window. Additionally, the WS_HSCROLL and WS_VSCROLL styles can be specified to give manual scrolling ability to the control. In such cases, the edit control manages its own scrolling, with no intervention required. If the edit control has no scroll bars, scrolling can be managed by sending the control WM_HSCROLL and WM_VSCROLL messages. |
| ES_NOHIDESEL | No hide selection. Normally text selected in an edit text control is highlighted only when the control has the input focus. The ES_NOHIDESEL flag specifies that the selected text in edit control remains highlighted even when the control doesn't have the input focus. |
| ES_OEMCONVERT | Edit text is converted from the ANSI character set to the OEM character set and back again to ANSI. When your application calls the AnsiToOem function to |

| | |
|---|---|
| | convert an ANSI string to OEM characters, this style ensures characters are converted properly. Usually this style is used for combo boxes that contain filenames. |
| ES_PASSWORD | Edit text is displayed as an asterisk (*) as it's typed in. |
| ES_READONLY | Edit text is read-only; the user can't enter or edit text. |
| ES_RIGHT | Edit text is aligned flush right. The text ends at the right side of control's rectangle.<br><br>Note that the ES_MULTILINE style must be set for this style to have any effect on the appearance of the text. |
| ES_UPPERCASE | Edit text is converted to upper case. |
| ES_WANTRETURN | When the user presses Enter, ES_WANTRETURN puts Enter in the edit buffer, rather than performing the action for the default key. This style applies to multiline text only. |

## LISTBOX class style constants

The LISTBOX class is used for list boxes. These boxes contain a list of strings or application program drawn items from which the user can select. LISTBOX is typically used for lists of file names, fonts, styles, etc.

The predefined constants for the LISTBOX class are:

| Style | Description |
|---|---|
| LBS_DISABLENOSCROLL | A list box with a disabled vertical scroll bar when it doesn't contain enough items to scroll. If this style is not used, the scroll bar won't be displayed. |
| LBS_EXTENDEDSEL | A list box with multiple items that are selected using the SHIFT key and the mouse or special key combinations. |
| LBS_HASSTRINGS | An owner-draw list box that contains string items. The list box maintains the memory and pointers for the string items. The LB_GETTEXT message retrieves the text for a particular item. |
| LBS_MULTICOLUMN | A multicolumn list box that is scrolled horizontally. The LB_SETCOLUMNWIDTH message sets the width of the columns. |
| LBS_MULTIPLESEL | A list box where more than one string in the list box can be selected. |
| LBS_NOINTEGRALHEIGHT | A list box that is not resized when partial items are displayed. Its size is exactly the size specified by the application when it created the list box. |
| LBS_NOREDRAW | The content of the list box is not redrawn when a change is made. This setting can be changed at run time by sending a WM_SETREDRAW message to the control. |
| LBS_NOTIFY | Notifies the parent window when the user clicks or double clicks in a control. |
| LBS_OWNERDRAWFIXED | A list box where the owner is responsible for drawing the contents of the list box. The items in the list box are the same height. The owner window receives a WM_MEASUREITEM message when the list box is created. When a visual aspect of the list box changes, the owner window receives a WM_DRAWITEM message . |
| LBS_OWNERDRAWVARIABLE | A list box where the owner is responsible for drawing the contents of the list obx. The items in the list box are variable in height. The owner window receives a WM_MEASUREITEM message when the list box is created. When a visual aspect of the list box chnages, the owner windows receives a WM_DRAWITEM message. |
| LBS_SORT | Strings in the list box are displayed in alphabetical order. |
| LBS_STANDARD | Strings in the list box are displayed in alphabetical order and parent window receives an input message when the user clicks or double-clicks a string. There are borders on all sides of the list box. |
| LBS_USETABSTOPS | Allows a list box to recognize and expand tab characters when |

| | |
|---|---|
| | drawing its strings. The default tab positions are 32 dialog units. |
| LBS_WANTKEYBOARDINPUT | When the list box has the focus and a user presses a key, the owner of the list box receives a WM_VKEYTOITEM or a WM_CHARTOITEM message. The application can perform special processing on the keyboard input. |

## SCROLLBAR class style constants

The SCROLLBAR class is used for the standard scroll bar control and for size boxes. Scroll bar controls have the same appearance as a window's scroll bar, except they can be positioned anywhere within a window.

The predefined constants for the SCROLLBAR class are:

| Style | Description |
|---|---|
| SBS_BOTTOMALIGN | The scroll bar appears the standard height, aligned with the bottom side of the control's rectangle. Valid for horizontal scroll bars only. |
| SBS_HORZ | Horizontal scroll bar. If SBS_BOTTOMALIGN or SBS_TOPALIGN is not specified, the scroll bar occupies the control's client area. |
| SBS_LEFTALIGN | The scroll bar appears the standard width, aligned with the left side of the control's rectangle. Valid for vertical scroll bars only. |
| SBS_RIGHTALIGN | The scroll bar appears the standard width, aligned with the right side of the control's rectangle. Valid for vertical scroll bars only. |
| SBS_SIZEBOXTOPLEFTALIGN | The size box appears the standard size, aligned in the upper-left corner of the control's rectangle. Valid for size boxes only. |
| SBS_SIZEBOX | Size box. If SBS_SIXEBOXTOPLEFTALIGN or SBS_SIZEBOXBOTTOMRIGHTALIGN is not specified, the size box occupies the control's client area. |
| SBS_SIZEBOXBOTTOMRIGHTALIGN | The size box appears the standard size, aligned in the lower-right corner of the control's rectangle. Valid for size boxes only. |
| SBS_TOPALIGN | The scroll bar appears the standard height, aligned with the top side of the control's rectangle. Valid for horizontal scroll bars only. |
| SBS_VERT | Vertical scroll bar. If SBS_RIGHTALIGN or SBS_LEFTALIGN is not specified, the scroll bar occupies the control's client area. |

## STATIC class style constants

The STATIC class is used for static controls. These controls include non-editable text and icons.

The predefined constants for the STATIC class are:

| Style | Description |
|---|---|
| SS_BLACKFRAME | Static item's rectangle; displayed with a frame in the system color COLOR_WINDOWFRAME. |
| SS_BLACKRECT | Static item's rectangle; filled with the system color COLOR_WINDOWFRAME. |
| SS_CENTER | Static item is text displayed centered in the control's rectangle. Text too long to fit is automatically wrapped to the next line. |
| SS_GRAYRECT | Static item's rectangle; filled with the system color COLOR_BACKGROUND. |
| SS_GRAYFRAME | Static item's rectangle; displayed with a frame in the system color COLOR_BACKGROUND. |
| SS_LEFT | Static item is text displayed left-justified in the control's rectangle. Text too long to fit is automatically wrapped to the next line. |
| SS_RIGHT | Static item is text displayed right-justified in the control's rectangle. Text too long to fit is automatically wrapped to the next line. |
| SS_WHITEFRAME | Static item's rectangle; displayed with a frame in the system color COLOR_WINDOW. |
| SS_WHITERECT | Static item's rectangle is filled with the system color COLOR_WINDOW. |

## load-type

load-type specifies when resources are loaded into memory.

Use the following keywords for load-type. LOADONCALL is the default load type if no keyword is specified.

| Keyword | Value |
|---|---|
| PRELOAD | Resource loaded at program start. |
| LOADONCALL | Resource is loaded when referenced by the application. |

## memory-option

memory-option specifies how resources are loaded into memory.

Use the following keywords for memory-option. The resource can be fixed at the same address at which memory-option is loaded, or it can be relocatable.

Additionally,the resource can be discardable or nondiscardable. MOVEABLE and DISCARDABLE are the default values.

| Keyword | Value |
|---|---|
| DISCARDABLE | Can be purged to make space. |
| FIXED | Resource stays at same address. |
| IMPURE | Resource is modified after loading. |
| MOVEABLE | Resource can be relocated in memory. |
| NONDISCARDABLE | Must stay in memory. |
| PURE | Resource is not modified after it is loaded. |

## Keyboard accelerator mnemonic

To specify a keyboard accelerator mnemonic, place an ampersand (&) before the letter which is to serve as the mnemonic.

This key is underlined by Windows when displayed, and the keyboard focus is assigned to the control when the user presses the mnemonic key with the Alt modifier key held down. To include an ampersand in the displayed text, use two ampersands (&&).

## Menu mnemonic

A menu mnemonic is used to activate the command using the keyboard.

Place the ampersand (&) character in front of the character in the item text which is to serve as the mnemonic. To include the & character in the actual item text, use &&. To include the single quote character in the item text, use two single quote characters.

## #define example

After you compile this example, SelectAll and IDS_SAMPLE have the value 1. IDS_ERROR has the value 2.

```
#define SelectAll  1
#define IDS_SAMPLE 1
#define IDS_ERROR  IDS_SAMPLE + 1
#define MYCAPTION  "hello"
```

**See Also**
#define

## #elif example

This example causes #define A to have the value 2.

```
#if (1 == 0)
#define A 1
#elif (1 == 1)
#define A 2
#endif
```

**See Also**
#elif

## #else example

This example causes #define A to have the value 3.

```
#if (1 == 0)
#define A 1
#elif (1 == 3)
#define A 2
#else
#define A 3
#endif
```

**See Also**

#else

## #endif example

#endif in this example marks the end of the scope of conditional compilation.

```
#if (1==0)
#define A 1
#elif (1==1)
#define A 2
#endif
```

**See Also**
#endif

## #if example

After you compile this example, neither A nor B will be defined.

```
#if (1 == 0)
#define A 1
#endif
#if defined A
#define B 2
#endif
```

**See Also**

#if

## #ifdef example

The #define NOATOM will be defined after you compile this example.

```
#define RC_INVOKED
#ifdef  RC_INVOKED
#define NOATOM
#endif
```

**See Also**
#ifdef

## #ifndef example

After you compile this example, the file WINDOWS.H will not be included by Resource Workshop.

```
#ifndef  WORKSHOP_INVOKED
#include <windows.h>
#endif
```

**See Also**
#ifndef

## #include examples

These examples show you how to #include C identifiers:

```
#include "w3demo.h"
#include <project.h>
```

These examples show you how to #include Pascal identifiers:

```
#include "MYCONSTS.INC"
#include <PROJECT.PAS>
```

**See Also**
#include

## ACCELERATOR examples

This example shows one way to define Ctrl+A (in hexadecimal), Ctrl+P, and Shift+Q as accelerators.

```
myaccel ACCELERATORS
BEGIN
  0x41,1, CONTROL, NOINVERT
  "^P",2
  VK_Q,3, VIRTKEY, SHIFT
END
```

If you want to use identifiers rather than numbers to identify your accelerators, use a C-style header file (with an .H extension) or a Pascal include file (with an .INC extension). For Pascal, you can also keep your identifiers in a unit.

This example uses #defines in the file KEYNAMES.H.

```
#define SelectAll  1
#define PrintNow   2
#define Quit       3
```

Then your script statements in the .RC file can use identifiers rather than numbers to uniquely identify each accelerator:

```
#include "keynames.h"
myaccel ACCELERATORS
BEGIN
  "^a", SelectAll, NOINVERT
  0x50, PrintNow, CONTROL
  "Q",  Quit
END
```

**See Also**
ACCELERATOR

## BITMAP examples

The bitmap "image" is in IMAGEF.BMP:

```
image BITMAP imagef.bmp
```

The bitmap 200 is in FRAME.BMP. It is loaded when the application starts:

```
200 BITMAP PRELOAD frame.bmp
```

The bitmap "bigstar" is in STAR.BMP. It cannot be moved around in memory:

```
bigstar BITMAP FIXED star.bmp
```

**See Also**
BITMAP

## CHECKBOX example

This example creates a check box with the text label "Backup files." The control-ID is 113 and the coordinates for the upper left corner of the check box are 13, 30. The control is 60 dialog units wide and 12 units high.

It uses the default styles, WS_TABSTOP and BS_CHECKBOX. WS_TABSTOP indicates that when the user presses Tab in a preceding control, the input focus changes to this control. It uses the default button style BS_CHECKBOX, indicating that the button is a check box.

```
CHECKBOX "Backup files", 113, 13, 30, 60, 12
```

**See Also**
CHECKBOX

## COMBOBOX example

This example creates a combo box with the ID 107. The coordinates for the upper-left corner of the combo box are 10, 9. The control is 49 dialog units wide and 33 units high.

It uses the window style constants WS_VSCROLL and WS_TABSTOP. WS_VSCROLL places a vertical scroll bar along the left side of the control. WS_TABSTOP indicates that when the user presses Tab in a preceding control, the input focus changes to this control. It uses the default combo box style, CBS_SIMPLE, to indicate that the list box is always displayed.

```
COMBOBOX 107, 10, 9, 49, 33, WS_VSCROLL | WS_TABSTOP
```

**See Also**
COMBOBOX

## CONTROL example

This example creates a control called "Forward." Its control ID is 105. The control's class is BUTTON, indicating that the control is a button. The button style is radio button (BS_RADIOBUTTON) and it starts a group (WS_GROUP).

The coordinates for the upper-left corner of the radio button are 68, 49 and the button is 49 dialog units wide and 12 high.

```
CONTROL "Forward", 105, "BUTTON", BS_RADIOBUTTON | WS_GROUP, 68, 49, 45, 12
```

**See Also**
CONTROL

## CTEXT example

This example centers static text in a rectangle of a specified size:

```
CTEXT "Copyright 1991 Acme Software", -1, 36, 22, 73, 8
```

**See Also**

CTEXT

## CURSOR examples

This example names the cursor resource 20 and tells Resource Workshop that the resource is found in the file HAND.CUR.

```
20 CURSOR hand.cur
```

This example names the resource "paintcan" and tells Resource Workshop it can be found in the file PAINT.CUR. The resource is loaded into memory when the application begins. It must remain in the same place in memory while the application runs. The application can't remove it from memory before the application ends.

```
paintcan CURSOR PRELOAD FIXED NONDISCARDABLE paint.cur
```

**See Also**
CURSOR

## DEFPUSHBUTTON example

This example creates a default push button control labeled "OK." Its control ID is 104. The coordinates of the upper-left corner of the dialog box are 11, 71. The button is 24 dialog units wide and 14 high.

The button class style constant BS_DEFPUSHBUTTON indicates that the button is a default push button. WS_TABSTOP indicates that when the user presses Tab in a preceding control, the input focus changes to this control.

```
DEFPUSHBUTTON "OK", 104, 11, 71, 24, 14, BS_DEFPUSHBUTTON | WS_TABSTOP
```

**See Also**
DEFPUSHBUTTON

## DIALOG example

This example shows how to use the DIALOG statement to display a simple dialog box that asks the user to select an option using radio buttons:

```
direction DIALOG 10, 10, 320, 130
STYLE WS_POPUP | WS_BORDER
CAPTION "Search Direction"
BEGIN
  CTEXT "Choose a direction:", 1, 15, 15, 250, 12
  RADIOBUTTON "&Forward",  2, 80, 35, 60, 12
  RADIOBUTTON "&Backward", 3, 80, 55, 60, 12
END
```

**See Also**
DIALOG

## EDITTEXT example

This is how you specify an EDITTEXT control with coordinates of 15, 15 for the control's upper-left corner. The control is 100 dialog units wide and 10 high.

The control uses the default styles, WS_BORDER, WS_TABSTOP, and ES_LEFT. WS_BORDER creates a thin border (1 pixel wide) around the control. WS_TABSTOP indicates that when the user presses Tab in a preceding control, the input focus changes to this control. ES_LEFT aligns the edit text flush left.

```
EDITTEXT 5, 15, 15, 100, 10
```

**See Also**
EDITTEXT

## FONT example

This example identifies a font as 2 and tells Resource Workshop the font is in the file VETICA.FON.

```
2 FONT vetica.fon
```

This example identifies a font as 27 and tells the Resource Workshop the font is in the file BOOK.FON. The font resource is loaded when the application starts and remains in memory until the application ends:

```
27 FONT PRELOAD NONDISCARDABLE book.fon
```

**See Also**
FONT

## GROUPBOX example

This example creates a group box in a dialog window. The group box is titled "Command set" in the upper left corner. The upper left corner of the group box control begins at coordinates 30, 15, and is 30 dialog units wide and 60 high.

```
GROUPBOX "Editor options", 30, 15, 30, 60
```

**See Also**
GROUPBOX

## ICON examples

This Type 1 example names an icon "myicon" and tells Resource Workshop to look for it in the file MYICON.ICO:

```
myicon ICON myicon.ico
```

This Type 2 example creates an icon named "myicon." The icon will be defined elsewhere in the resource file with a Type 1 icon statement. The icon's control ID is 120, and the upper-left corner coordinates are 30, 40. Notice that there are no width and height arguments.

```
ICON "myicon" 120, 20, 20
```

**See Also**
ICON

## LISTBOX example

This example creates a list box that is:

- uniquely identified as 200
- has coordinates for the upper-left corner at 50, 60
- is 60 dialog units wide and 52 high

```
LISTBOX 200, 50, 60, 60, 52
```

**See Also**
LISTBOX

## LTEXT example

This example creates a disabled control 93 dialog units wide and 8 high. The left corner of the rectangular control begins at coordinates 4, 60. The text "Version 3.10" is flush left within the rectangle.

```
LTEXT "Version 3.10", 104, 4, 60, 93, 8, WS_DISABLED
```

**See Also**
LTEXT

## MENU example

This example specifies the main menu of an application. The main menu contains the File menu and the Help menu. File is a pop-up menu (commonly known as a drop-down menu) with several options, each defined with a MENUITEM substatement.

The ampersand (&) underlines the letter that immediately follows it. This lets the user choose the command from the menu by typing that letter.

```
mainmenu MENU PRELOAD
BEGIN
  POPUP      "&File"
  BEGIN
    MENUITEM "&New", 100
    MENUITEM "&Open", 101
    MENUITEM "&Close", 102
    MENUITEM "&Save", 103
    MENUITEM "Save &As", 104
    MENUITEM SEPARATOR
    MENUITEM "&Print", 105
    MENUITEM SEPARATOR
    MENUITEM "E&xit", 106
  END
  MENUITEM "&Help", 200
END
```

**See Also**
MENU

## MENUITEM example

The MENUITEM statements in this example specify the options that will appear on the menu. Note the use of SEPARATOR to separate the file options from the printing options and to separate the printing options from the exit option.

The ampersand (&) underlines the letter that immediately follows it. This lets the user choose the command from the menu by typing that letter.

```
mainmenu MENU PRELOAD
BEGIN
  POPUP      "&File"
  BEGIN
    MENUITEM "&New", 100
    MENUITEM "&Open", 101
    MENUITEM "&Close", 102, GRAYED
    MENUITEM "&Save",103
    MENUITEM "Save &As", 104
    MENUITEM SEPARATOR
    MENUITEM "&Print", 105
    MENUITEM "&Draft Printing", 107, CHECKED
    MENUITEM SEPARATOR
    MENUITEM "E&xit", 106
  END
  MENUITEM "&Help", 200
END
```

**See Also**
MENUITEM

## POPUP example

In this example, when the user selects the File menu, the menu "drops-down" (because of the POPUP statement) and displays all the options on the menu.

The ampersand (&) underlines the letter that immediately follows it. This lets the user choose the command from the menu by typing that letter.

```
mainmenu MENU PRELOAD
BEGIN
  POPUP      "&File"
  BEGIN
    MENUITEM "&New", 100
    MENUITEM "&Open", 101
    MENUITEM "&Close", 102, GRAYED
    MENUITEM "&Save", 103
    MENUITEM "Save &As", 104
    MENUITEM SEPARATOR
    MENUITEM "&Print", 105
    MENUITEM "&Draft Printing", 107, CHECKED
    MENUITEM SEPARATOR
    MENUITEM "E&xit", 106
  END
  MENUITEM "&Help", 200
END
```

**See Also**
POPUP

## PUSHBUTTON example

This example defines an "OK" push button. Its control ID is 108. The coordinates of the upper-left corner of the button are 15, 55. The button is 24 dialog units wide and 14  long.

It uses the defauult window style WS_TABSTOP, indicating that when the user presses Tab in a preceding control, the input focus changes to this control. It uses the default button style BS_PUSHBUTTON, indicating that the button is a push button.

```
PUSHBUTTON "OK", 108, 15, 55, 24, 14
```

**See Also**
PUSHBUTTON

## RADIOBUTTON example

This example defines a radio button called "Left." Its control ID is 101. The coordinates of the upper-left corner of the button are 10, 60. The button is 28 dialog units wide and 12 long.

It uses the default window style WS_TABSTOP, indicating that when the user presses Tab in a preceding control, the input focus changes to this control. It uses the default button style BS_RADIOBUTTON, indicating that the button is a radio button.

```
RADIOBUTTON "LEFT", 101, 10, 60, 28, 12
```

**See Also**
RADIOBUTTON

## RCDATA examples

These examples show you the variety of data you can include in a raw data resource:

```
// a single string (not null-terminated)
mystring RCDATA PRELOAD
BEGIN
  "Now is the time."
END

// a bunch of data
mydata RCDATA
BEGIN
  0x1000,
  255,
  "Null-terminated string\0",
  0777,
END
```

**See Also**

RCDATA

## RCINCLUDE example

This example opens the source file SEARCH.DLG and processes the directives and resource definitions in the file.

```
RCINCLUDE SEARCH.DLG
```

**See Also**

RCINCLUDE

## RTEXT example

This example creates a disabled control 93 dialog units wide and 8 high. The left corner of the rectangular control begins at coordinates 4, 60. The text "Version 3.10" is flush right within the rectangle.

```
RTEXT "Version 3.10", 104, 4, 60, 93, 8, WS_DISABLED
```

**See Also**
RTEXT

## SCROLLBAR example

This example defines a vertial scroll bar, whose control ID is 102. The coordinates of the upper-left corner are 13, 15. The scroll bar is 17 dialog units wide and 64 high. SBS_TOPALIGN creates a scroll bar that is aligned with the top side of the control's rectangle.

```
SCROLLBAR 102, 13, 15, 17, 64, SBS_HORZ | SBS_TOPALIGN
```

**See Also**
SCROLLBAR

## STRINGTABLE example

This example lists strings an application might use for displaying messages to the user:

```
STRINGTABLE
BEGIN
  1,  "Press any key to continue..."
  2,  "Click mouse to continue..."
  3,  "All rights reserved"
  4,  "Disk Full"
END
```

To use the #define, set up the .H file with the proper #define statements. This example calls the file STRNAMES.H.

```
#define PressKey    1
#define PressMouse  2
#define AllRights   3
#define DiskFull    4
```

Then in the .RC file:

```
#include strnames.h
/* a single table with defines */
STRINGTABLE
BEGIN
  PressKey,   "Press any key to continue..."
  PressMouse, "Click mouse to continue..."
  AllRights,  "All rights reserved"
  DiskFull,   "Disk Full"
END
```

**See Also**
STRINGTABLE

## User-defined resources example

In the first example, the new resource type is DEFAULTS and is named "defdata". The new resource is in the file DFLT.RES.

```
defdata DEFAULTS dflt.res
```

In this example, the resource type is 256 and is named "printertype". The resource is in the file PRTYP.RES. The resource can be moved around in memory and then removed from memory when the application no longer needs it.

```
printertype 256 MOVEABLE DISCARDABLE prtyp.res
```

**See Also**

User-defined resources

## Alphabetical listing of errors

This is an alphabetical listing of Resource Workshop errors. For a functional listing of errors, go to the Functional listing of errors Help screen.

Create a new identifier: <identifier name>?
Custom control library entry points missing from this library
#define is used. Delete anyway?
#define text too long
Delete Resource: <resource name>
Device dependent bitmap does not match current display. Cannot convert
Division by zero is not allowed
Duplicate command value found
Duplicate IDs not allowed
Duplicate key value found
'#else' before '#if'
'#endif' before '#if'
Entry too large
#error directive encountered: <error message>
Expecting ')'
Expecting #define identifier
Expecting a number or '('
Expecting BEGIN
Expecting caption: quoted string or unsigned integer
Expecting class name or ID
Expecting constant expression
Expecting control window style
Expecting END
Expecting filename
Expecting filename in quotes
Expecting filename or BEGIN
Expecting HEXSTRING (hex digits surrounded by single quotes)
Expecting identifier
Expecting interface keyword
Expecting menu text (quoted string) or SEPARATOR
Expecting quoted string
Expecting resource ID or name
Expecting resource type ID or name
Expecting resource name or resource type name
Expecting semicolon
Expecting unit keyword
Expecting unsigned long integer
Expecting unsigned short integer
Expecting signed long integer
Expecting signed short integer
Expecting window rectangle (4 signed integers)
Expression stack overflow
External leading must be from 0 to 65535
Field too large
<filename> does not exist
<filename> does not exist. Create?
<filename> exists. Overwrite?
<filename> exists. Replace resources?
<filename> has changed. Save?
<filename> is not a valid file name
File <filename> does not exist
File create
File extension: <.EXT> does not match the standard: <EXT> Continue anyway?
File IO error
File is not a dynamic link library

File read failed
File seek failed
File write failed
Fill area too complex. Some portions were not filled.
Font family must be from 0 to 15
Font is too large to save in Version 2 format! (>64K)
Font weight must be from 1 to 1000
Fonts must have numeric resource IDs
Height must be from 1 to 32767
HEXSTRING over 255 bytes long
Horizontal resolution must be from 1 to 65535
Identifier already exists
Identifier is used as a literal
Image colors exceed device capabilities. Changes will not be saved.
Incomplete expression
Incorrect resource compiler version
Info function returned a bad Handle
Info function return a NULL Handle
Input reset failed
Input source stack overflow
Internal leading must be less than character height
Internal software error
Invalid accelerator key value
Invalid accelerator option
Invalid bitmap format
Invalid cursor format
Invalid escape sequence
Invalid file format
Invalid font format
Invalid font specification
Invalid icon format
Invalid menu option
Invalid number of rows or columns
Invalid preprocessor directive
Invalid stretch parameters
Invalid value in HEXSTRING
Macro parameter substitution is not supported
Maximum height is 32767
Maximum of 255 controls
Maximum width is 32767
Memory allocation error during decompile
Memory lock failed
Memory unlock failed
Menu too large to edit
Must be first token on a line
Must be within character range
New field instance failed
New symbol failed
No duplicate command values found
No duplicate key values found
No resources in this file
Not a positive short integer
Not a valid identifier
Not a valid identifier name
Not a valid resource name

Not enough memory to edit this bitmap
Not implemented
Number of undo levels must be > 1 and < 100
Parser stack overflow
PASCAL string over 255 bytes
Pascal string too long
Pascal syntax error (unrecognized token)
Please close a window
Please enter a name
Please enter a valid filename
Please enter a valid number
Points must be from 1 to 65535
Preprocessor directives not allowed
Remove contents of <filename> from this project?
Resource already exists
Resource binary too large
Resource editor initialization error
Resource has changed. Compile? WARNING: Changes will be discarded if you answer no!
Resource is too large to edit as text
Resource of that name/ID, and type already exists
Resource type already exists
Software error!
Source input stack overflow (too many nested includes?)
String ID is already used
Symbol already defined
Syntax error
System resources low. Please close some files or applications.
Table too large to edit
Token is too large for scanner (unbalanced quotes?)
Too many controls
Too many digits in a number
Too many items to paste
Too many items to view.
Too much data for 1 field
#undef is not supported
Unexpected end of file
Unexpected const keyword
Unexpected file format
Unexpected NULL pointer encountered
Unexpected operator
Unknown bitmap format
Unreleased version format
Value is out of range
VBuff allocation error
VBuff lock error
Vertical resolution must be from 1 to 65535
Virtual buffer allocation
Virtual buffer lock
Virtual table allocation failed
Virtual table create
Virtual table get
Virtual table lock
Virtual table put
Virtual table read
Virtual table write

VTMgr allocation error
VTMgr lock error
WARNING! This DLL may or may not be a valid custom control library. If it is not, a system crash is likely! Do you wish to proceed?
Width must be from 1 to 32767
You cannot undo this action! Select cancel to stop
You cannot use this identifier. It is a keyword, resource type name, or resource name
You have opened a Windows version 2 file.

## Functional listing of errors

This is a functional listing of Resource Workshop errors. For an alphabetical listing of errors, go to the Alphabetical listing of errors Help screen.

**Accelerator editor errors**

An ACCELERATORS table must have at least one item
Could not allocate undo for duplicate
Duplicate key value found
Height must be from 1 to 32767
No duplicate key values found
Please close a window
You cannot undo this action! Select cancel to stop
Value is out of range
Width must be from 1 to 32767

**Compiler errors**

#define text too long
'#else' before '#if'
'#endif' before '#if'
#undef is not supported
A compile is in progress in another instance
Bad character in source input
Binary too large
Cannot open file: <filename>
Compile initialization failed
Division by is zero not allowed
#error directive encountered: <error message>
Expecting ')'
Expecting #define identifier
Expecting a number or '('
Expecting BEGIN
Expecting caption: quoted string or unsigned integer
Expecting class name or ID
Expecting constant expression
Expecting control window style
Expecting END
Expecting filename
Expecting filename in quotes
Expecting filename or BEGIN
Expecting HEXSTRING (hex digits surrounded by single quotes)
Expecting identifier
Expecting interface keyword
Expecting menu text (quoted string) or SEPARATOR
Expecting quoted string
Expecting resource ID or name
Expecting resource type ID or name
Expecting resource name or resource type name
Expecting semicolon
Expecting signed long integer
Expecting signed short integer
Expecting unit keyword
Expecting unsigned long integer
Expecting unsigned short integer
Expecting window rectangle (4 signed integers)
Expression stack overflow

Field too large
File IO error
HEXSTRING over 255 bytes long
Incomplete expression
Incorrect resource compiler version
Input reset failed
Invalid accelerator key value
Invalid accelerator option
Invalid bitmap format
Invalid cursor format
Invalid escape sequence
Invalid font format
Invalid font specification
Invalid icon format
Invalid menu option
Invalid preprocessor directive
Invalid value in HEXSTRING
Macro parameter substitution is not supported
Must be first token on a line
No resources in this file
Not a positive short integer
Parser stack overflow
PASCAL string over 255 bytes
Pascal string too long
Pascal syntax error (unrecognized token)
Please enter a valid filename
Source input stack overflow (too many nested includes?)
Symbol already defined
Syntax error
Token is too large for scanner (unbalanced quotes?)
Too many controls
Too many digits in a number
Too much data for 1 field
Unexpected const keyword
Unexpected end of file
Unexpected operator
You have opened a Windows version 2 file.

**Dialog editor errors**

Could not allocate undo for duplicate
Custom control library entry points missing from this library
File is not a dynamic link library
Height must be from 1 to 32767
Info function returned a bad Handle
Info function return a NULL Handle
Invalid number of rows or columns
Maximum of 255 controls
Please close a window
Too many items to paste
Value is out of range
WARNING! This DLL may or may not be a valid custom control library. If it is not, a system crash is likely! Do you wish to proceed?
You cannot undo this action! Select cancel to stop
Width must be from 1 to 32767

**File management errors**

Cannot add a new RES or program file to an RC project
Cannot add <filename> to project twice
Cannot determine format from extension
Cannot save to a running program
Change reference in <filename> from <reference> to <new reference>?
Command line parameter error
Could not open file
<filename> does not exist
<filename> does not exist. Create?
<filename> exists. Overwrite?
<filename> exists. Replace resources?
<filename> has changed. Save?
<filename> is not a valid file name or path
File <filename> does not exist
File create
File extension: <.EXT> does not match the standard: <EXT> Continue anyway?
File read failed
File seek failed
File write failed
Invalid file format
Remove contents of <filename> from this project?
Unexpected file format
Unreleased version format
Virtual table read

**Identifier errors**

#define is used. Delete anyway?
Constant is used. Delete anyway?
Could not create identifier
Create a new identifier: <identifier name>?
Identifier already exists
Identifier is used as a literal
Input source stack overflow
New symbol failed
Not a valid identifier
Not a valid identifier name
You cannot use this identifier. It is a keyword, resource type name, or resource name

**Internal errors**

Internal software error
Not implemented
Software error!
Unexpected NULL pointer encountered
Virtual buffer lock
Virtual table create
Virtual table get
Virtual table lock
Virtual table put
Virtual table write

**Memory errors**

Allocate failed
Could not allocate memory
Could not allocate undo for delete
Could not allocate undo for memory options
Could not allocate undo for new

Could not allocate undo for rename
Memory allocation error during decompile
Memory lock failed
Memory unlock failed
New field instance failed
System resources low. Please close some files or applications.
Too many items to view.
VBuff allocation error
VBuff lock error
Virtual buffer allocation
Virtual table allocation failed
VTMgr allocation error
VTMgr lock error

**Menu editor errors**

A MENU or POPUP must have at least one item
Cannot delete last item or pop-up
Could not allocate undo for duplicate
Duplicate command value found
Height must be from 1 to 32767
Menu too large to edit
No duplicate command values found
Please close a window
You cannot undo this action! Select cancel to stop
Value is out of range
Width must be from 1 to 32767

**Paint editor errors**

An image of this size and color attributes exists. Continue anyway?
Ascent must be less than character height
Cannot edit vector fonts
Character set must be from 0 to 255
Character width must be from 1 to maximum width
Could not allocate undo for duplicate
Could not create bitmap
Could not create icon image
External leading must be from 0 to 65535
Fill area too complex. Some portions were not filled.
Font family must be from 0 to 15
Font is too large to save in Version 2 format!  (>64K)
Font weight must be from 1 to 1000
Fonts must have numeric resource IDs
Horizontal resolution must be from 1 to 65535
Image colors exceed device capabilities. Changes will not be saved.
Internal leading must be less than character height
Invalid stretch parameters
Maximum height is 32767
Maximum width is 32767
Must be within character range
Not enough memory to edit this bitmap
Please close a window
Please enter a valid number
Points must be from 1 to 65535
Unknown bitmap format
Vertical resolution must be from 1 to 65535
You cannot undo this action! Select cancel to stop

Value is out of range

**Resource errors**

A resource of that name already exists
Cannot convert Windows 2 image: file is read-only
Cannot find resource
Conflicting memory options
Could not create resource
Delete Resource: <resource name>
Device dependent bitmap does not match current display. Cannot convert
Not a valid resource name
Please enter a name
Preprocessor directives not allowed
Resource already exists
Resource binary too large
Resource editor initialization error
Resource has changed. Compile? WARNING: Changes will be discarded if you answer no!
Resource is too large to edit as text
Resource of that name/ID, and type already exists
Resource type already exists

**String editor errors**

Cannot duplicate a stringtable
Cannot rename a stringtable
Could not allocate undo for duplicate
Duplicate IDs not allowed
Entry too large
Height must be from 1 to 32767
Please close a window
String ID is already used
Table too large to edit
You cannot undo this action! Select cancel to stop
Value is out of range
Width must be from 1 to 32767

## Identifier already exists

The specified identifier already exists. Use a unique name.

Note that C #defines are case-sensitive, but Pascal constants are not.

## Memory lock failed

Resource Workshop could not lock memory. Exit Resource Workshop immediately, without saving files. Start Windows again.

## Fonts must have numeric resource IDs

Font resources cannot have alphanumeric literal names. The FONT resource requires a positive short (16-bit) integer font ID. Enter an integer ID value.

## Please close a window

Resource Workshop allows 10 editor windows to be open at the same time. To open another window, you need to close a currently open window. To close the current editor window, press Ctrl+F4.

## Duplicate key value found

Two entries in an accelerator table have the same key value. Each entry must be unique.

**See Also**
Accelerators

## No duplicate key values found

No entries in an accelerator table have the same key value.

**See Also**
Accelerators

## Invalid number of rows or columns

The number of rows and columns in the array dialog must be sufficient to hold the number of selected controls.

**See Also**
Dialog boxes

## Too many items to paste

You cannot paste these controls in your dialog box, because it would cause the number of controls in the dialog to exceed limits. A dialog box can have a maximum of 255 controls.

**See Also**
Dialog boxes

## Maximum of 255 controls

This dialog box already contains 255 controls. You cannot add a new control without deleting another control. A dialog box can have a maximum of 255 controls.

**See Also**
Dialog boxes

## File is not a dynamic link library

You selected a file that is not a dynamic link library. When you install a custom control library, the file must be a dynamic link library.

**See Also**
Dialog boxes

## Custom control errors

The custom control errors are:

- Custom control library entry points missing from this library
- Info function returned a NULL handle
- Info function returned a bad handle

These errors mean that the specified dynamic link library does not look like a valid custom control library.

**See Also**
Dialog boxes

## Could not create icon image

There is not enough memory to create an icon image. Close one or more applications to free up memory.

To find out how much memory is available, switch to the Program Manager and choose Help|About. The dialog box displays the available system memory.

**See Also**
Icons

## Please close a window

Resource Workshop allows 10 editor windows to be open at the same time. To open another window, you need to close a currently open window. To close the current editor window, press Ctrl+F4.

**See Also**
Icons

## An image of this size and color attributes exists. Continue anyway?

You're trying to create an image that has the same size and color attributes as an existing icon.

Windows will only choose one image when displaying an icon. If there are two of the same size and color attributes, one will never be used.

**See Also**
Icons

## Not enough memory to edit this bitmap

There is not enough memory to edit this bitmap. Close one or more applications to free up memory.

If that doesn't free enough memory, try opening the bitmap as a bitmap project. If there is still not enough memory to edit the bitmap, you might have to obtain more memory to edit the bitmap.

To find out how much memory is available, switch to the Program Manager and choose Help|About. The dialog box displays the available system memory.

**See Also**
Bitmaps

## Unknown bitmap format

Resource Workshop does not understand this bitmap format. Resource Workshop edits only Windows 3 and OS/2 1.x device-independent bitmaps.

Read the bitmap into the program that produced it and use the clipboard to transfer the image to a new bitmap.

**See Also**
Bitmaps

## Could not create bitmap

There is not enough memory to create the requested bitmap. Close one or more applications to free up memory.

If that doesn't free enough memory, try opening the bitmap as a bitmap project. If there is still not enough memory to create the bitmap, you might have to obtain more memory to create the bitmap.

**See Also**
Bitmaps

## Character width must be from 1 to maximum width

You specified a character width that is greater than the maximum width for this font. Either increase the value in the Maximum Width input box (one of the Sizes input boxes) in the Font Size Information dialog box or enter a smaller character width.

**See Also**
Fonts

## Must be within character range

The default and break characters must be within the values entered in the First and the Last input boxes.

Either change the character range values or change the values in the Default and the Break input boxes.

**See Also**
Character input boxes
Fonts

## Cannot edit vector fonts

Windows 3.x supports two types of fonts, raster and vector. Resource Workshop's font editor can only edit raster fonts. (Some examples of vector fonts are Roman, Modern, and Script.)

**See Also**
Fonts

## Image colors exceed device capabilities. Changes will not be saved.

You are trying to edit an image with more colors than your device driver supports.

Any changes you make to this image will not be saved by Resource Workshop. To change this image, create another image and use the clipboard to copy this image to the new one.

**See Also**
Using the Paint editor

## Please enter a valid number

The number you entered is out of the range permitted for this field. The permitted range is 1 to 255.

**See Also**
Using the Paint editor

## Fill area too complex. Some portions were not filled.

The paint can tool was not able to complete the requested operation. It ran out of memory.

**See Also**
Using the Paint editor

## Invalid stretch parameters

A value entered in the Stretch Selection dialog box was out of range for an integer. Please enter a correct value (-32768 to 32767).

**See Also**
Using the Paint editor

## No duplicate command values found

The command values for this menu and all its pop-ups contain no duplicate values.

**See Also**
Menus

## Menu too large to edit

The Outline pane for this menu required more than 64K. You must use an external text editor to edit a menu this large. Try working in the Integrated Development Environment.

**See Also**
Menus
Outline pane

## Duplicate command value found

The item highlighted in the Outline pane contains a command value that duplicates a value in a menu item earlier in the menu list.

If you don't want duplicate values, change this value.

**See Also**
Menus
Outline pane

## Cannot delete last item or pop-up

A pop-up menu or menu item must contain at least one entry.

**See Also**
Menus

## Table too large to edit

The table window for this string table requires more than 64K, which exceeds its capacity.

Use an external text editor, such as the Integrated Development Environment, to split the string table or edit the table.

**See Also**
String tables

## Entry too large

The maximum length of a string in a string table is 255 characters.

**See Also**
String tables

## Duplicate IDs not allowed

String table IDs must be unique per project. If the duplicate value is not in the table you are editing, it is probably in another table.

**See Also**
String tables

## Could not allocate memory

Resource Workshop could not obtain memory for an operation. Exit Resource Workshop immediately, without saving files. To free memory for use by Resource Workshop, exit other applications, or run Windows in Enhanced mode.

## Memory lock failed

Resource Workshop could not lock memory. Exit Resource Workshop immediately, without saving files. Start Windows again.

## Memory unlock failed

Resource Workshop could not unlock global memory. Exit Resource Workshop immediately, without saving files. Start Windows again.

## File create

Resource Workshop could not create a file.

Verify that the specified file does not already exist and that there is sufficient directory or disk space for the file. Retry the operation that caused the error.

## Could not open file

Resource Workshop could not open the specified file.

Verify that the file exists. Retry the operation that caused the error.

## File seek failed

Resource Workshop failed in seeking to a location in a file.

The file may be corrupted. Retry the operation that caused the error. Try running CHKDSK on the disk.

## File read failed

Resource Workshop could not read the specified file.

Verify that the file exists and is readable. Retry the operation that caused the error.

## File write failed

Resource Workshop could not write to the specified file.

Verify that the file exists and can be written to. Retry the operation that caused the error.

## Virtual table allocation failed

Resource Workshop could not obtain memory for an operation. Exit Resource Workshop immediately, without saving files.

To free up more memory for use by Resource Workshop, try exiting other applications, or running Windows in Enhanced mode.

## Virtual table put

Resource Workshop's internal database is probably corrupt. Exit Resource Workshop immediately, without saving files. You should also exit Windows.

## Virtual table read

Resource Workshop could not read the specified file.

Verify that the file exists and is readable. Retry the operation that caused the error.

## Virtual table get

Resource Workshop's internal database is probably corrupt. Exit Resource Workshop immediately, without saving files. You should also exit Windows.

## Virtual table create

Resource Workshop's internal database is probably corrupt. Exit Resource Workshop immediately, without saving files. You should also exit Windows.

## Virtual table write

Resource Workshop could not write to the specified file.

Verify that the file exists and can be written to. Retry the operation that caused the error.

## Virtual table lock

Resource Workshop's internal database is probably corrupt. Exit Resource Workshop immediately, without saving files. You should also exit Windows.

## Virtual buffer allocation

Resource Workshop could not obtain memory for an operation.

Exit Resource Workshop immediately, without saving files. To free memory for Resource Workshop, exit other applications or run Windows in Enhanced mode.

## Virtual buffer lock

Resource Workshop's internal database is probably corrupt. Exit Resource Workshop immediately, without saving files. You should also exit Windows.

## Binary too large

A binary data item (resource or field) that is too large for Resource Workshop could not be compiled.

## Unexpected NULL pointer encountered

Resource Workshop encountered an unexpected NULL pointer.

## Input source stack overflow

Resource Workshop could not open an include or rcinclude file or expand a #define. Too many files are open or too many #defines are nested.

**See Also**
Identifiers and identifier files

## Cannot find resource

Resource Workshop could not find the selected resource. Exit Resource Workshop immediately, without saving files.

## Unexpected file format

This error can occur when Resource Workshop:

- decompiles a binary resource. In this case, the error means that Resource Workshop could not match the binary data with the resource type definition. Resource Workshop skips the resource.
- saves a program or dynamic link library. The error means that the file is non-standard. This error most often occurs when you try to save Microsoft applications, such as Word for Windows, that use a non-standard executable file format.

## Unreleased version format

The version of the file you are opening is greater than Resource Workshop supports.

## Software error!

Resource Workshop encountered unexpected data. Exit Resource Workshop immediately, without saving files. You should also exit Windows.

## Not implemented

The selected function is not implemented in this release of Resource Workshop.

## You cannot use this identifier. It is a keyword, resource type name or resource name

You are trying to create an identifier whose name conflicts with a keyword, resource type name, or resource name. Choose a unique name.

## Cannot determine format from extension

You started Resource Workshop with a file name argument in the command line. Resource Workshop cannot determine the file format from the extension.

Use the File|Open Project command to open the project.

## File <filename> does not exist

The specified file does not exist in the given path. Check the path and the spelling of the file name.

## Command line parameter error

You started Resource Workshop with a command line containing an illegal argument.

## Cannot save to a running program

Windows does not support dynamic update of running programs. You must exit the program before saving, or use File|Save File As to save to a different file.

## <filename> exists. Overwrite?

You are saving to an existing file. Choose Yes to replace the contents of that file. Choose Cancel to abort the process and return to Resource Workshop.

## <filename> exists. Replace resources?

You are saving resources into an existing program or dynamic link library. Choose Yes to replace the resources in that file. Choose Cancel to abort the process and return to Resource Workshop.

## <filename> has changed. Save?

The contents of the file or project have changed. Choose Yes to save the changes. Choose Cancel to abort the process and return to Resource Workshop.

## Remove contents of <filename> from this project?

If you answer Yes, all resources and/or identifiers in the named file and in any file referenced by the file will be removed from this project.

Because you cannot undo the action, Resource Workshop displays a Warning dialog box when you answer Yes.

The warning message reads:

> You cannot undo this action! Select cancel to stop

## You cannot undo this action! Select cancel to stop

There is no undo for the action you are about to perform. This is your last chance to change your mind.

## Change reference in <filename> from <reference> to <new reference>?

You used the File|Save File As or Resource|Save Resource As command to save a file or resource.

Resource Workshop wants to know whether to change the reference in the current project to that file or resource to the new name just specified. Choose Yes to change the reference.

## <filename> is not a valid file name or path

The specified file name is invalid. Correct the name and retry the operation that caused the error.

## <filename> does not exist

The specified program or dynamic link library file does not exist. Resource Workshop will not create program files, but will only add resources to those files.

## Cannot add <filename> to project twice

The same file cannot be added to the same project two times.

If the file contains a binary resource, select the resource in the Project window. Then use Edit|Duplicate to duplicate the resource. Use Resource|Save Resource As to save the resource to a new file.

## Cannot add a new RES or program file to an RC project

When you use the Add File to Project dialog box, you can't create a new file. You need to specify a file that already exists.

## Cannot duplicate a stringtable

Stringtable resources cannot be duplicated, because duplication would result in duplicate IDs.

## Could not allocate undo for duplicate

There is not enough memory available to undo a duplicate resource action. Close some applications to free memory.

## Identifier already exists

The specified identifier already exists. Use a unique name.

Note that C/C++ #defines are case-sensitive, but Pascal constants are not.

**See Also**
Identifiers and identifier files

## Could not create identifier

The specified identifier could not be created.

## A resource of that name already exists

The selected name is already assigned to another resource. Choose another name.

## Too many items to view.

There is not enough memory available to view the requested items. Close some applications to free memory.

## Bad character in source input

The specified source file contains an unrecognizable character.

## #define text too long

The definition for the specified #define is too long for Resource Workshop to store. A #define definition must be less than 2000 characters.

## Invalid preprocessor directive

Resource Workshop has encountered a # (pound sign) character that is not followed by a valid preprocessor directive name.

## Symbol already defined

Resource Workshop encountered a #define whose name is a keyword, or whose definition is not the same as a previous definition.

Although duplicate definitions of the same #define are ignored, two different definitions for the same #define are not allowed.

## Expecting #define identifier

Resource Workshop encountered a #define followed by an illegal name. #define names must begin with a letter and contain only letters, digits, and underscores.

## '#else' before '#if'

Resource Workshop encountered an #else or an #elif directive without a corresponding #if directive. Use a text editor to correct this syntax error.

## '#endif' before '#if'

Resource Workshop encountered an #endif directive without a corresponding #if directive. Use a text editor to correct this syntax error.

## Unexpected end of file

An end of file was encountered when processing a compiler directive (#if, #ifdef, etc.).

## Expecting resource name or resource type name

Resource Workshop encountered an undefined identifier or integer expression, which it classified as a resource name or ID. It expects the next token to be a resource type name or ID, but encountered something else.

## Expecting ')'

A numeric expression contains unbalanced parentheses.

## Expecting identifier

Resource Workshop encountered an illegal token in an #ifdef or #if preprocessor directive.

## Expecting constant expression

Resource Workshop could not evaluate an integer expression.

## Expecting filename

A quoted or unquoted filename is expected.

## Expecting filename in quotes

An #include statement was not followed by a filename surrounded by quotes or angle brackets.

## Expecting a number or '('

Resource Workshop encountered an unexpected token when attempting to parse an integer expression. This error is frequently caused by an error in an identifier name.

## Expecting BEGIN

Resource Workshop encountered an unexpected token when searching for the BEGIN keyword. This error is frequently caused by a typo in an identifier name.

## Expecting END

Resource Workshop encountered an unexpected token when searching for the END keyword. This error is frequently caused by a typo in an identifier name.

## Not a positive short integer

Resource and resource type IDs must be positive short integers.

## HEXSTRING over 255 bytes long

A hexstring data item was over 255 bytes long. This syntax extension in Resource Workshop is not supported by the Microsoft Resource Compiler.

## Invalid value in HEXSTRING

A hexstring data type (a Resource Workshop syntax extension not supported by the Microsoft Resource Compiler) is a series of hex digits and white space surrounded by single quotes.

Resource Workshop encountered a character within the single quotes that cannot be interpreted as a hex digit.

## Field too large

Resource Workshop encountered a data field larger than 32K.

## PASCAL string over 255 bytes

A Pascal format string must be less than 256 bytes in length.

## Cannot open file:<filename>

Resource Workshop could not open the specified file. You may have insufficient rights to the file.

## Conflicting memory options

A resource definition with conflicting memory options has been encountered.

## String ID is already used

ID values in stringtable resources must be unique within a single project.

## Resource of that name/ID, and type already exists

Resource names or IDs must be unique within type.

## Incomplete expression

Resource Workshop could not completely evaluate an expression.

## A MENU or POPUP must have at least one item

A MENU and POPUP definition must contain at least one menu item or menu separator.

## An ACCELERATORS table must have at least one item

You cannot define an empty accelerator table.

## Fonts must have numeric resource IDs

Font resources cannot have alphanumeric literal names. The FONT resource requires a positive short (16-bit) integer font ID.

## Invalid font format

Resource Workshop could not compile a font resource due to an invalid format.

This error typically occurs when you try to examine ROMAN.FON, SCRIPT.FON, or MODERN.FON. These vector fonts are shipped with Windows.

## Invalid icon format

Resource Workshop could not compile an icon resource due to an invalid file format.

## Invalid cursor format

Resource Workshop could not compile a cursor resource due to an invalid format.

## Invalid bitmap format

Resource Workshop could not compile a bitmap resource due to an invalid format.

## Expecting filename or BEGIN

Resource Workshop could not compile a file or resource. It encountered a token that was not a file name, curly brace, or a BEGIN keyword.

## #undef is not supported

Resource Workshop does not support #undef. If the #undef define is required in this .H file in order to satisfy C requirements, surround the statement with the #ifndef directive.

## Macro parameter substitution is not supported

Resource Workshop does not support complex macros with parameters in this release.

## Expecting signed short integer

Resource Workshop's incremental compiler expects to see a signed short (16-bit) integer or integer expression in this field. This error is usually caused by an undefined identifier.

## Expecting unsigned short integer

Resource Workshop's incremental compiler expects to see an unsigned short (16-bit) integer or integer expression in this field. This error is usually caused by an undefined identifier.

## Expecting signed long integer

Resource Workshop's incremental compiler expects to see a signed long (32-bit) integer or integer expression in this field. This error is usually caused by an undefined identifier.

## Expecting unsigned long integer

Resource Workshop's incremental compiler expects to see an unsigned long (32-bit) integer or integer expression in this field. This error is usually caused by an undefined identifier.

## Expecting quoted string

Resource Workshop's incremental compiler expects to see a quoted string in this field.

## Expecting resource ID or name

Resource Workshop expects a positive short integer or an unquoted alphanumeric literal for a resource name or ID.

## Expecting resource type ID or name

Resource Workshop expects a positive short integer or an unquoted alphanumeric literal for a resource type name or ID.

## Expecting window rectangle (4 signed integers)

A dialog control requires 4 unsigned integers to specify its location and size.

## Expecting class name or ID

A dialog control requires a quoted name or unsigned character with a value > 0x7F. The unsigned character syntax is reserved for use by standard windows controls.

## Expecting HEXSTRING (hex digits surrounded by single quotes)

The CTLDATA keyword for dialog controls must be followed by a HEXSTRING data field.

## Invalid menu option

The valid options for POPUP or MENUITEM statements are:

- CHECKED
- MENUBREAK
- MENUBARBREAK
- INACTIVE
- GRAYED
- HELP

## Invalid accelerator option

The valid accelerator options are: ASCII, VIRTKEY, SHIFT, ALT, CONTROL, and NOINVERT.

## Invalid accelerator key value

Accelerator key values must contain:

- unsigned characters
- a quoted string that includes an unsigned character and a ^ to indicate control keys

## Invalid font specification

This dialog definition contains the FONT keyword that is not followed by a point size/face name pair.

## Expecting caption: quoted string or unsigned integer

A dialog control caption must be either a quoted string or an unsigned short (16-bit) integer.

## Pascal string too long

A string that is stored in Pascal format (like those in string tables) can be no longer than 255 bytes.

## Expecting control window style

The control specification on this line is not an unsigned long (32-bit) integer or expression. This error is usually caused by an undefined style identifier.

## Expecting menu text (quoted string) or SEPARATOR

The MENUITEM keyword must be followed either by a quoted string (the item text) or the keyword SEPARATOR.

## Compile initialization failed

The compiler could not complete initialization. Exit and restart both Windows and Resource Workshop.

## Input reset failed

The compiler could not complete initialization. Exit and restart both Windows Resource Workshop.

## Source input stack overflow (too many nested includes?)

The sum of your nested includes or #defines exceeds 63.

## Parser stack overflow

The nesting level of a recursive resource definition exceeds a Resource Workshop limitation. For example, this error can occur if the number of nested pop-ups exceeds 63.

## Expression stack overflow

An integer expression is too complex for Resource Workshop to evaluate. The maximum nesting depth is 32. Try simplifying the expression by removing parentheses.

## File IO error

A read error occurred on a file. Check to see if the file is corrupt.

## New symbol failed

Resource Workshop could not create a #define or constant. This error is usually caused by a lack of memory.

To find out how much system memory is available, switch to the Program Manager and choose Help| About. The dialog box displays the available system memory.

## New field instance failed

In the process of creating a field record, Resource Workshop could not obtain memory for an operation. Exit Resource Workshop immediately, without saving files.

To free up more memory for use by Resource Workshop, exit other applications, or run Windows in Enhanced mode.

## Allocate failed

Resource Workshop could not obtain memory for an operation. Exit Resource Workshop immediately, without saving files.

To free up more memory for use by Resource Workshop, exit other applications, or run Windows in Enhanced mode.

## Memory lock failed

Resource Workshop could not lock memory. Exit Resource Workshop immediately, without saving files. You should also exit Windows.

## VTMgr allocation error

Resource Workshop could not obtain memory for an operation. Exit Resource Workshop immediately, without saving files.

To free up more memory for use by Resource Workshop, exit other applications, or run Windows in Enhanced mode.

## VTMgr lock error

Resource Workshop could not lock memory. Exit Resource Workshop immediately, without saving files. You should also exit Windows.

## VBuff allocation error

Resource Workshop could not obtain memory for an operation. Exit Resource Workshop immediately, without saving files.

To free up more memory for use by Resource Workshop, exit other applications, or run Windows in Enhanced mode.

## VBuff lock error

Resource Workshop could not lock memory. Exit Resource Workshop immediately, without saving files. You should also exit Windows.

## Internal software error

Resource Workshop encountered unexpected data. Exit Resource Workshop immediately, without saving files. You should also exit Windows.

## Device dependent bitmap does not match current display. Cannot convert

You tried to open a Windows 2.0 resource file containing a device-dependent bitmap in a format that does not match the current display device. Use another tool to convert the bitmap.

## Preprocessor directives not allowed

Preprocessor directives are not allowed when editing a resource as text.

## Too much data for 1 field

You entered more data in this field than the incremental compiler allows. This error may also be caused by an invalid character that was parsed as an extra token. Delete the extra data.

## A compile is in progress in another instance

You tried to start a compile while a compile is still in progress in another copy of Resource Workshop. Wait until the compile has completed in the other copy before retrying this operation.

## Cannot convert Windows 2 image: file is read-only

You opened a resource file containing a Windows 2.0 format bitmap resource. However, the file is read-only. Change the file's attributes to examine the file in Resource Workshop.

## Too many controls

A dialog resource contains more than 255 controls. Only 255 controls are allowed per dialog template.

## Invalid escape sequence

A backslash character in a string was not followed by a valid escape code.

## Too many digits in a number

The number contains too many digits to be represented by an unsigned 32-bit integer.

## Expecting unit keyword

Resource Workshop encountered an unrecognized token when it was expecting a unit keyword. Make sure that the include or unit file is syntactically correct by compiling it.

**See Also**
Identifiers and identifier files

## Expecting semicolon

Resource Workshop encountered an unrecognized token when it was expecting a semicolon. Make sure that the include or unit file is syntactically correct by compiling it.

**See Also**
Identifiers and identifier files

## Expecting interface keyword

Resource Workshop encountered an unrecognized token when it was expecting an interface keyword. Make sure that the include or unit file is syntactically correct by compiling it.

**See Also**
Identifiers and identifier files

## Must be first token on a line

A preprocessor directive or constant name must be the first token on a source line. Use an external text editor to correct the source.

## Pascal syntax error (unrecognized token)

Resource Workshop encountered an unrecognized token. Make sure that the include or unit file is syntactically correct by compiling it.

**See Also**
Identifiers and identifier files

## Unexpected const keyword

Resource Workshop encountered an unexpected const keyword. Make sure that the include or unit file is syntactically correct by compiling it.

**See Also**
Identifiers and identifier files

## Unexpected operator

Resource Workshop encountered an unexpected operator. Make sure that the include or unit file is syntactically correct by compiling it.

**See Also**
Identifiers and identifier files

## Please enter a valid filename

You specified a text editor in the Compile Error dialog box that does not exist. Enter the complete filename, including the file extension.

## Identifier already exists

The name you selected for this identifier is already in use. Use a unique name.

Note that C/C++ defines are case-sensitive, but Pascal constants are not.

**See Also**
Identifiers and identifier files

## Please enter a name

You tried to create a new resource without specifying a name. Enter a name and choose OK.

## Resource already exists

A resource of the type and name specified already exists in this project. Resource names and IDs must be unique by type and in a project. Use a name or ID that is not used for a resource of this type.

## Could not create resource

Resource Workshop could not create the new resource. If the resource name is valid, try the operation again.

If the error persists, check to see how much memory is available. If memory is low, close another application or run Windows in enhanced mode.

## Please close a window

Resource Workshop allows 10 editor windows to be open at the same time. To open another window, you need to close a currently open window. To close the current editor window, press Ctrl+F4.

## Resource editor initialization error

An unspecified error occurred while trying to create a resource editor window. This type of error should not occur during normal operation of the program.

It is possible portions of your disk may be corrupted. Run CHKDSK to test the disk. Otherwise your resource file may be damaged.

## Cannot rename a stringtable

Because stringtable resources are not named you cannot rename one.

The name Resource Workshop displays for a stringtable resource is merely the ID or identifier of the first string in the table.

## Fonts must have numeric resource IDs

Font resources cannot have alphanumeric literal names. The FONT resource requires a positive short (16-bit) integer font ID.

## Could not allocate undo for rename

There was not enough memory available to create an undo record for the rename operation. Save the current project, and exit Resource Workshop. To free memory, exit other applications or run Windows in enhanced mode.

## Could not allocate undo for delete

There was not enough memory available to create an undo record for the delete operation. Save the current project, and exit Resource Workshop. To free memory, exit other applications or run Windows in enhanced mode.

## Could not allocate undo for new

There was not enough memory available to create an undo record for the new operation. Save the current project, and exit Resource Workshop. To free memory, exit other applications or run Windows in enhanced mode.

## Could not allocate undo for memory options

There was not enough memory available to create an undo record for the specified operation. Save the current project, and exit Resource Workshop. To free memory, exit other applications or run Windows in enhanced mode.

## Not a valid identifier

The name you selected for a #define or constant is not syntactically correct. Identifier names must start with a letter and contain only letters, digits, and underscores.

## Resource type already exists

You are trying to create a new resource type that already exists. Use a unique name or ID for the resource type.

Note that user-defined resource type IDs should be greater than 256, since the numbers 1 to 256 are reserved for use by the operating system vendor.

## Identifier is used as a literal

The name you have chosen for this identifier is used as a literal in this project (either a resource name, resource type name, or keyword). Choose a unique name for the identifier.

## Resource is too large to edit as text

Resource Workshop uses a standard Windows edit window for its text editor. The maximum capacity of this window is approximately 32K. In order to edit this resource as text, you must save it into a separate .RC file, and use a larger capacity text editor, such as the Integrated Development Environment.

## #define/constant is used. Delete anyway?

This #define or constant is used in one or more resources. If you delete it, Resource Workshop will substitute its value for its name in every place it is used.

## Create a new identifier: <identifier name>?

You entered an alphanumeric value in a numeric field. In order to use that alphanumeric value in this field, you must assign a number to it by creating an identifier.

Click Yes to create a new identifier. The identifier will be placed in one of the project's header, unit, or include files.

## Font is too large to save in Version 2 format! (>64K)

You cannot save this font in Windows Version 2.0 format. It is too large.

**See Also**
Fonts

## Width must be from 1 to 32767

You entered an incorrect width value. Enter a value from 1 to 32767.

## Height must be from 1 to 32767

You entered an incorrect height value. Enter a value from 1 to 32767.

## Maximum width is 32767

You entered an incorrect width value. Enter a value from 1 to 32767.

## Maximum height is 32767

You entered an incorrect height value. Enter a value from 1 to 32767.

## Font weight must be from 1 to 1000

The value in the Weight input box is incorrect. Enter a value from 1 to 1000.

**See Also**
Attributes boxes (Font Header Information dialog box)
Fonts

## Font family must be from 0 to 15

The value in the Family input box is incorrect. Enter a value from 0 to 15.

**See Also**
Fonts

## Character set must be from 0 to 255

The value you entered in the Character Set input box is incorrect. Enter a value from 0 to 255.

**See Also**
Fonts

## Horizontal resolution must be from 1 to 65535

The value you entered in the Horizontal Resolution input box is incorrect. Enter a value from 1 to 65535.

**See Also**
Fonts
Sizes input boxes (Font Header Information dialog box) de

## Vertical resolution must be from 1 to 65535

The value you entered in the Vertical Resolution input box is incorrect. Enter a value from 1 to 65535.

**See Also**
Fonts
Sizes input boxes (Font Header Information dialog box) de

## Points must be from 1 to 65535

The value you entered in the Points input box (one of the Sizes input boxes) is invalid. Enter a value from 1 to 65535.

**See Also**
Fonts

## Internal leading must be less than character height

The value you entered in the Internal Leading input box (one of the Sizes input boxes in the Font Header Information dialog box) is incorrect.

It must be less than the value in the Height input box (one of the Sizes input boxes in the Font Size Information dialog box).

**See Also**
Fonts

## External leading must be from 0 to 65535

The value you entered in the External Leading input box (one of the Sizes input boxes in the Font Header Information dialog box) is incorrect. Enter a value from 0 to 65535.

**See Also**
Fonts

## Ascent must be less than character height

The value in the Ascent input box (one of the Sizes input boxes in the Font Header Information dialog box) is invalid.

It must be less than the value in the Height input box (one of the Sizes input boxes in the Font Size Information dialog box).

**See Also**
Fonts

## Division by zero is not allowed

You specified a constant expression or a remainder function that contains a divide by zero. This is not allowed - you must change the expression or function.

## Delete Resource: <resource name>

The selected resource will be deleted. Click Yes if you're sure you want to delete it. Click No if you don't want to delete it.

## #error directive encountered: <error message>

The Resource Workshop compiler encountered an #error directive. The text of the user-defined message is displayed.

**See Also**
#error

## Please enter a valid number

You must enter an integer value between 0 and 255 in the RGB input boxes of the Edit Color dialog box or the Set Transparent Color dialog box. Enter the value again.

**See Also**
Using the Paint editor

## File extension: <.EXT> does not match the standard for this type: <.EXT> Continue anyway?

You are saving a file to an extension that's not supported for that resource type. For example, you're trying to save a cursor resource to a .BMP file type.

Click Yes to save the file to the unsupported file type. Click No to cancel the file save.

Supported file extensions include:

| File extensions | File type |
|---|---|
| .BMP | bitmap |
| .CUR | cursor image |
| .DLG | dialog box resource script |
| .DLL | dynamic link library |
| .DRV | Windows device driver |
| .EXE | executable |
| .FNT | font |
| .FON | font library |
| .ICO | icon image |
| .RC | resource script |
| .RES | binary resource |

## WARNING! This DLL may or may not be a valid custom control library. If it is not, a system crash is likely! Do you wish to proceed?

You are trying to install a dynamic link library file that might not be valid custom control library. If the file is invalid, your system could crash. You might want to save all files before proceeding.

If you are developing your own custom controls, be sure that:

- your **ClassInfo, ClassStyle,** and **ClassFlags** functions have the correct ordinal values
- your **ListClasses** function is correctly exported.

## <filename> does not exist. Create?

The specified file does not exist. Click Yes if you want to create it.

## Incorrect resource compiler version

You tried to open an .EXE file whose resource version mark is not supported by Resource Workshop.

## Invalid file format

You tried to open a file that is in an unrecognizable format. It is either invalid or has been damaged. Try running CHKDSK on your hard disk to check for data validity.

## Memory allocation error during decompile

There was not enough available memory to open the specified .RES or .EXE file. Close some applications or windows and try again.

To find out how much memory is available, go to the Program Manager and choose Help|About. The dialog box displays the available system memory and resources.

## No resources in this file

You tried to open a file that contains no resources.

## Not a valid identifier name

The name you selected for a #define or constant is not syntactically correct. Identifier names must start with a letter and contain only letters, digits, and underscores.

## Not a valid resource name

The name you selected for a resource is not syntactically correct. Resource names must start with a letter and contain only letters, digits, and underscores.

## Number of undo levels must be > 1 and < 100

The number of undo levels must be between 1 and 100. Enter a valid value.

## Resource binary too large

The binary resource file you're trying to read into Resource Workshop is too large.

## Resource has changed. Compile? WARNING: Changes will be discarded if you answer no!

The resource you're working on has changed. If you don't compile the changes, they will be discarded.

Choose Cancel to return to Resource Workshop without losing any changes or saving any files.

## Syntax error

A syntax error has occurred. Check the resource script syntax and recompile.

## Token is too large for scanner (unbalanced quotes?)

The scanner tried to create a token for a string or hexstring, but couldn't find the closing single or double quote. Check the resource script for unmatched quotes.

## Value is out of range

The value you entered is not valid. Enter the correct value.

## System resources low. Please close some files or applications.

Resource Workshop has detected a low memory condition in USER.EXE's heap, used to store windows and menus. Close some windows and/or applications.

To find out how many system resources are available, go to the Program Manager and choose Help| About. The dialog box displays the available system memory and resources.

## <filename> exists. Overwrite?

## <filename> exists. Replace resources?

The first message is displayed if you are saving to an existing file. Choose Yes to replace the contents of that file.

The second message is displayed if you are saving resources into an existing program or dynamic link library. Choose Yes to replace the resources in that file.

## You have opened a Windows version 2 file.

The file you're trying to open contains resources that are in Windows version 2 format. Resource Workshop will convert all Windows version 2 resources to version 3 format.

Click OK to continue the process. Click Cancel to cancel the conversion.

## Functional listing of Resource Workshop menu commands

Here's a functional listing of the menus in Resource Workshop. For an alphabetical listing, see the Alphabetical listing of Resource Workshop menus Help screen.

**Primary menus**
Compile
Control
Edit
File
Help
Resource
View
Window

**Accelerator editor menu**
Accelerator

**Menu editor menus**
Menu
View

**Stringtable editor menu**
Stringtable

**Dialog editor menus**
Align
Control
Options

**Paint editor menus**
Bitmap
Cursor
Font
Icon
Images
Options
View
Text

## Alphabetical listing of Resource Workshop menus commands

Here's an alphabetical listing of the menus in Resource Workshop. For a functional listing, see the Functional listing of Resource Workshop menus Help screen.

Accelerator
Align
Bitmap
Compile
Control
Cursor
Edit
File
Font
Help
Icon
Images
Menu
Options
Resource
Stringtable
Text
View
Window

## Control menus

There are two Control menus in Resource Workshop:

- the Control menu that affects the Resource Workshop desktop
- the Control menu that displays when you create a dialog box

## Options menus

There are two Options menus in Resource Workshop:

- the Options menu that displays when you edit a dialog box
- the Options menu that displays when you edit a bitmapped resource

## View menus

There are three View menus in Resource Workshop:

- the View menu that displays when you open a project or resource
- the View menu that displays when you edit a bitmapped resource
- the View menu that displays when you edit a menu

## File menu

The File menu, along with the Help menu, is displayed when you first open Resource Workshop. You use this menu to create, open, close, and save projects; add resource files to and remove resource files from projects; close files; set configuration options; and exit Resource Workshop.

Here are the File menu commands:
New Project
Open Project
Preferences
Exit

Once you've created or opened a project, several more File commands are available:
Save Project
Save File As
Close All
Add to Project
Remove from Project

## File | New Project

File|New Project brings up the New Project dialog box, where you create your project file and choose the type of file on which to base your project.

## New Project dialog box

The New Project dialog box is where you choose the type of file on which to base your project. Display this dialog box with File|New Project.

Use the Project File Type radio buttons to select the project file type. Here are the types of files on which you can base your project:

**.RC**
Turn on the .RC radio button to base your project on a resource script file. Use this option to add in any type of resource.

**.RES**
Turn on the .RES radio button to base your project on a binary resource file. This type of file contains one or more compiled resources.

**.CUR**
Turn on the .CUR radio button to base your project on a cursor (.CUR) file. A cursor file contains a single cursor image.

**.ICO**
Turn on the .ICO radio button to base your project on an icon (.ICO) file. An icon file contains one or more images of an icon.

**.BMP**
Turn on the .BMP radio button to base your project on a bitmap (.BMP) file. A bitmap file contains a single bitmap image.

**.FNT**
Turn on the .FNT radio button to base your project on a font (.FNT) file. A font file contains one or more customized font images.

**See Also**
Bitmap files
Creating a project
Cursor files
Font files
Icon files
Resource compiler files
Resource files

## File | Open Project

File|Open Project opens an existing project. An existing project is one that you created with Resource Workshop or an .RC file you created with other resource development software.

The File|Open Project command brings up the Open Project dialog box.

## Open Project dialog box

The Open Project dialog box is where you open an existing project. Display this dialog box with File| Open Project.

**File Name input box**
The File Name input box is where you enter the name of the project to open.

**File Type list**
The File Type list selects the type of file to open.

**Path display box**
The Path display box displays the current path.

**Files list box**
The Files list box displays the files in the current directory.

**Directories list box**
The Directories list box displays the directories on your computer.

**See Also**
Opening an existing project

## File Name input box

Enter the file name in the File Name input box. This input box has a different meaning based on the dialog box you're working with:

- If you're using the Add File to Project dialog box, enter the name of the file you're adding to the current project.
- If you're using the Open Project dialog box, enter the name of the project you want to open.
- If you're using the Save File As dialog box, enter the name of the file you want to save.
- If you're using the Save Resource As dialog box, enter the new file name for the resource you're working with.

If you enter a file extension that represents a standard resource file type, Resource Workshop assigns the proper type to the file. However, if you use a nonstandard extension, be sure to pick the correct file type from the File Type list before exiting the dialog box.

## File Type list

Select the type of file from the File Type list. If you enter a file extension representing a standard resource file type in the File Name input box, Resource Workshop assigns the proper file type to the file.

However, if you use a nonstandard extension, click the arrow in the File Type list box and scroll down to the correct file type. Double-click the correct file type.

## Path display box

The Path display box shows the current path.

- If you're using the Add File to Project dialog box, the Path display box shows where the file will be read from if you enter only the file name.
- If you're using the Open Project dialog box, the Path display box shows where the file will be read from if you enter only the file name.
- If you're using the Save File As dialog box, the Path display box shows where the file will be stored if you enter only the file name.
- If you're using the Save Resource As dialog box, the Path display box shows where the file will be stored if you enter only the file name.

## Files list box

To use the Files list box, enter a file name with a wildcard in the File Name input box and press Enter. You see a list of all matching files appear in the File list box. Scroll down and select a file.

## Directories list box

You use the Directories list box to choose a directory. You can double-click a directory name or icon to switch to that directory. To move up one level, double-click (..).

The directory name then appears in the File Name input box. When you press Enter, the directory name is displayed in the Path display box.

## File | Save Project

The File|Save Project command saves everything in your current project. If you save a new project that hasn't been named yet, Resource Workshop displays the Save File As dialog box, where you specify the file name and directory.

## File | Save File As

Use the File|Save File As command if you want to rename the current project or resource file. Resource Workshop brings up the Save File As dialog box, where you specify the file's new name and directory.

## Save File As/Save Resource As dialog box

The Save File As dialog box is where you rename a file when you save it.

The Save Resource As dialog box is where you put a resource in a separate file for use with other projects.

**See Also**
Saving a project
Saving resources and resource files

## Save File As dialog box

The Save File As dialog box is where you rename a project file or resource file when you save it. Display this dialog box with File|Save File As.

**File Name input box**
The File Name input box is where you enter the new file name.

**File Type list**
The File Type list selects the type of file to open.

**File Selected display box**
The File Selected display box shows file's current path and name.

**Path display box**
The Path display box displays the current path.

**Files list box**
The Files list box displays the files in the current directory.

**Directories list box**
The Directories list box displays the directories on your computer.

**See Also**
Saving a project

## File | Close All

The File|Close All command closes all open files. If you've made changes and haven't saved them, Resource Workshop asks you if you want to save the changes. Click Yes.

When it closes files, Resource Workshop compiles the resources that have changed since the last compile and saves them into the project file. Any changed resource in an external file and linked to the project file will be updated in the external file.

## File | Add To Project

The File|Add To Project command adds a resource stored in an external file to the current project.

This command brings up the Add File to Project dialog box where you specify the file name.

This command is not available for the .CUR, .ICO, .BMP, and .FNT resource types.

**See Also**
Bitmap (.BMP) files
Cursor (.CUR) files
Font (.FNT) files
Icon (.ICO) files

## Add File to Project/New File Resource dialog box

The Add File To Project dialog box adds a resource stored in an external file to the current project.

The New File Resource dialog box creates a file for a resource you want to store in an external file and specifies the project in which it will be referenced.

## Add File to Project dialog box

The Add File to Project dialog box is where you specify the name of an external file to add to your project. Display this dialog box with File|Add File to Project.

**File Name input box**
The File Name input box is where you enter the name of the external file.

**File Type list**
The File Type list selects the type of file to open.

**RCINCLUDE will be Placed In drop-down box**
The RCINCLUDE drop-down box selects the file where the RCINCLUDE resource script statement will be placed.

**Path display box**
The Path display box displays the current path.

**Files list box**
The Files list box displays the files in the current directory.

**Directories list box**
The Directories list box displays the directories on your computer.

**See Also**
Adding a linked resource to a project
Working with binary files

## RCINCLUDE will be Placed In drop down box

The RCINCLUDE drop down box selects the file where the RCINCLUDE statement will be placed. This is usually your current project file.

If your project contains more than one .RC file, and you want to put the reference elsewhere, scroll down the list to find the name of the file in which you want to place the reference.

**See Also**
RCINCLUDE

## New File Resource dialog box

The New File Resource dialog box is where you

- create a file for a resource you want to store in an external file and
- specify the project in which it will be referenced.

**File Name input box**
The File Name input box is where you enter the name of the external file.

**File Type list**
The File Type list selects the type of file to open.

**A Reference to the Resource in this File Will Be Placed In drop-down box**
This drop down box selects the name of the project in which you want to reference the new resource file.

**Path display box**
The Path display box displays the current path.

**Files list box**
The Files list box displays the files in the current directory.

**Directories list box**
The Directories list box displays the directories on your computer.

**See Also**
Working with binary files

## File | Remove From Project

The File|Remove From Project command removes the selected file from the current project. This command is enabled only when the View|By File command is active.

## File | Preferences

The File|Preferences command brings up the Preferences dialog box, where you set configuration options for Resource Workshop.

## Preferences dialog box

The Preferences dialog box is where you set Resource Workshop configuration options. Display this dialog box with File|Preferences.

**Undo Levels input box**
The Undo Levels input box is where you enter the number of changes you want to trace when you undo or redo actions.

**Text Editor input box**
The Text Editor input box is where you specify the name of the text editor that Resource Workshop uses.

**Include Path input box**
The Include Path input box lets you enter the path where Resource Workshop searches for include files.

**Multi-Save check boxes**
The Multi-Save check boxes determine how a project is saved.

**Backups check box**
The Backups check box creates backup files each time you save a project.

**Windows target version check boxes**
The Windows target version check boxes choose the Windows version you're working with.

**See Also**
Working with binary files

## Undo Levels input box

The Undo Levels input box is where you enter the number of changes you want to trace when you undo or redo actions with the Edit|Undo or Edit|Redo commands.

Depending on the amount of memory in your computer, you can undo or redo up to 99 actions. The default number is 10.

## Text Editor input box

The Text Editor input box is where you specify the name of the text editor that Resource Workshop uses. The default text editor is the Windows Notepad editor, NOTEPAD.EXE.

You use the text editor to edit a resource's source script.

## Include Path input box

Enter the path that you want Resource Workshop to search for identifier files in the Include Path input box.

Set this option when all projects are closed. When you choose this option, Resource Workshop saves it in WIN.INI as the default include path.

## Multi-Save check boxes

The Multi-Save check boxes control how a project is to be saved when you select File|Save Project. These preferences are enabled only when a project is open because they apply to a specific project.

**.RES check box**
The .RES check box saves the project's resources in a resource file (.RES) in binary format.

**Executable check box**
The Executable check box saves the project's resources and binds them to the specified executable file. The executable file can be an .EXE file or a .DLL file. If you work with a decompiled binary file that you've saved as an .RC file, you can save both the .RC and the binary file by using this option.

**See Also**
Executable files
Resource files

## Make Backups When Saving Files check box

Turn on the Make Backups When Saving Files check box to create a set of backup files each time you save a project. Backup files have a tilde as the first character in the file extension, for example, .~RC.

## Windows target version check boxes

The Windows target version check boxes choose the Windows version you're working with.

.RES and .EXE files targeted for Windows 3.1 are not backward-compatible with Windows 3.0. If the .RES file is targeted to Windows 3.1, you cannot used the Windows 3.0 version of RC.EXE to bind your resources to the application. You must use Borland tools or version 3.1 of the Microsoft Resource Compiler.

**Windows 3.0 check box**
Turn this check box on if you're creating resources for a Windows 3.0 application.

**Windows 3.1 check box**
Turn this check box on if you're creating resources for a Windows 3.1 application.

## File | Exit

The File|Exit command closes all files and exits Resource Workshop.

If you've made changes and haven't saved them, Resource Workshop asks you if you want to save the changes.

## Edit menu

The Edit menu provides commands to undo, redo, cut, copy, paste, delete, and select areas of your image.

Here are the Edit menu commands:
Undo
Redo
Cut
Copy
Paste
Delete
Duplicate
Select All

## Edit | Undo (Alt+Backspace)

Edit|Undo "undoes" your most recent action. It inserts any characters you deleted, deletes any characters you inserted, replaces any characters you overwrote, and moves your cursor back to its prior position.

If you undo a block operation on a resource, the resource will appear as it was before you executed the block operation.

If you continue to press Undo, it continues to undo changes until it reaches the number specified in the Undo Levels input box in the Preferences dialog box.

The Undo command will not change an option setting that affects more than one window.

## Edit | Redo (Shift+Alt+Backspace)

Edit|Redo reverses the effect of the most recent Undo command. It is effective only immediately after you use Edit|Undo or Edit|Redo.

A series of Redo commands reverses the effects of a series of Undo commands.

## Edit | Cut (Shift+Del)

Edit|Cut puts the selected area of your image, text block, or resource in the Windows clipboard. You can then choose Edit|Paste to paste the image somewhere else in the same project or in another project running in another copy of Resource Workshop.

## Edit | Copy (Ctrl+Ins)

Edit|Copy copies the selected area of your image, text block, or resource. To paste the copied image somewhere else in the same project or in another project running in another copy of Resource Workshop, choose Edit|Paste.

## Edit | Paste (Shift+Ins)

Edit|Paste inserts the selected area of your image, text block, or resource at the cursor position. This command is active only if you select the area with the Edit|Cut or Edit|Copy commands.

If you're in the Project window, this command brings up the Paste Resource dialog box, where you paste resources.

## Paste Resource dialog box

The Paste Resource dialog box is where you paste resources and associated identifiers into a new file.

**Paste Resource Into drop-down box**
The Paste Resource Into drop-down box is where you choose the file into which you're going to paste the resource.

**Paste Identifiers Into drop-down box**
The Paste Identifiers Into drop-down box is where you choose the file into which you're going to paste the associated identifiers.

## Edit | Delete (Del)

Edit|Delete removes the selected area of your image or the selected resource. You can't paste the area as you could if you had chosen Edit|Cut or Edit|Copy.

If you're in the String editor, this command is the same as the Stringtable|Delete Item command.

## Edit | Duplicate

Edit|Duplicate duplicates an area of an image or a resource. It duplicates an area of the image by copying the area you select and placing the copy in the upper left corner of the image. As long as the area is still selected, you can use the mouse or arrow keys to move the copied area anywhere in the image.

If you're in the Project window, this command brings up the Duplicate Resource dialog box. If you're in the Dialog editor, this command brings up the Duplicate Control dialog box.

## Duplicate Resource dialog box

The Duplicate Resource dialog box is where you duplicate a resource.

**Duplicate Resource Will Be Placed In drop-down box**
The Duplicate Resource Will Be Placed In drop-down box is where you choose the file into which you're going to place the duplicate resource.

## Duplicate Control dialog box

The Duplicate Control dialog box is where you place multiple copies of a control in rows and columns.

**Rows input box**
The Rows input box is where you specify the number of rows you want.

**Columns input box**
The Columns input box is where you specify the number of columns you want.

**Row Spacing input box**
The Row Spacing input box is where you specify the spacing between rows.

**Column Spacing input box**
The Column Spacing input box is where you specify the spacing between columns.

**See Also**
Adding multiple copies of controls

## Edit | Select All

Edit|Select All selects the entire source script, image, or resource. You can then cut the script, image, or resource, and copy, paste, delete, or duplicate it.

**See Also**
Edit|Cut
Edit|Copy
Edit|Paste
Edit|Delete
Edit|Duplicate

## Resource menu

You use the Resource menu to create, edit, save, and rename resources; define a resource's memory options; and add, edit, and delete identifiers.

Here are the Resource menu commands:
New
Edit
Edit As Text
View
Save Resource As
Rename
Memory Options
Identifiers

## Resource | New

The Resource|New command brings up the New Resource dialog box where you create a new resource.

This command is not available when the project file is of the .CUR, .ICO, .BMP, or .FNT type.

**See Also**
Bitmap (.BMP) files
Cursor (.CUR) files
Font (.FNT) files
Icon (.ICO) files

## New Resource dialog box

The New Resource dialog box is where you specify the type of resource to create. Display this dialog box with Resource|New.

**Resource Type list box**
The Resource Type list box selects the type of resource to create.

**Place Resource In combo box**
The Place Resource In combo box selects where to put the new resource.

**New Type button**
The New Type button brings up the New Resource Type dialog box, where you create your own resource type.

**See Also**
Adding an embedded resource to a project

## Resource Type list box

Select the type of resource you want to create from the Resource Type list box.

To create your own resource type, press the New Type button to bring up the New Resource Type dialog box. If you've already created a resource type, it will be listed in the Resource Type list.

**See Also**
Creating a user-defined resource type

## Place Resource In combo box

The Place Resource In combo box allows you to specify the project in which the new resource will be placed.

If your project contains more than one .RC file and you want to put the new resource in a project that is different from the one in the combo box, select the file name from the drop-down list.

**See Also**
Saving a dialog box in a resource script file

## New Resource Type/Custom Resource dialog box

The New Resource Type dialog box is where you create your own resource type.

The Custom Resource dialog box is where you choose a user-defined resource type.

**See Also**
Creating a user-defined resource type

## New Resource Type dialog box

The New Resource Type dialog box is where you create your own resource type. Display this dialog box by choosing the New Type button in the New Resource dialog box.

**Resource Type input box**
The Resource Type input box is where you enter the name of the new resource type.

**Resource Type list**
The Resource Type list shows the resources that are already defined.

**See Also**
Creating a user-defined resource type
User-defined resources

## Custom Resource dialog box

You see the Custom Resource dialog box when you use the Add File to Project dialog box to read in an existing file that's not a standard resource type.

You can click OK, Cancel, or press the New Type button to display the New Resource Type dialog box.

**See Also**
User defined resources

## Resource | Edit

The Resource|Edit command brings up the selected resource in the appropriate editor, ready for editing.

**See Also**
Editing a bitmap
Editing a cursor
Editing a dialog box
Editing a font resource
Editing a menu
Editing a string table
Editing a user-defined resource type
Editing an accelerator table
Editing an icon
Using a resource editor

## Resource | Edit As Text

The Resource|Edit As Text command brings up the selected resource in the text editor, where you edit the resource's source script.

**See Also**

Editing a user-defined resource
Editing an accelerator table's resource script
Editing an icon resource as text
Editing a bitmap resource as text
Editing a cursor resource as text
Editing a dialog box resource as text
Editing a font resource as text
Editing a menu resource as text
Editing a string table resource as text
Resource script language
Using a text editor

## Resource | View

Resource|View shows you what the selected resource looks like. This is a display only mode - you can't edit the resource.

To edit the resource, choose Resource|Edit or Resource|Edit As Text.

## Resource | Save Resource As

The Resource|Save Resource As command puts a resource in a separate file for use with other projects.

Resource Workshop brings up the Save Resource As dialog box, where you specify the file's new name and directory.

## Save Resource As dialog box

The Save Resource As dialog box is where you rename a resource when you save it. Display this dialog box with Resource|Save Resource As.

**File Name input box**
The File Name input box is where you enter the resource's new file name.

**File Type list**
The File Type list selects the type of file in which to save the resource.

**Resource display box**
The Resource display box shows the name currently assigned to the resource.

**Path display box**
The Path display box displays the current path.

**Files list box**
The Files list box displays the files in the current directory.

**Directories list box**
The Directories list box displays the directories on your computer.

**See Also**
Saving a user-defined resource as a file
Saving the accelerator table in a resource script file
Saving the bitmap resource as a file
Saving the cursor resource as a file
Saving the dialog box in a resource script file
Saving the font resource as a file
Saving the icon resource as a file
Saving the menu resource in a resource script file
Saving resources and resource files

## Resource | Rename

The Resource|Rename command brings up the Rename Resource dialog box, where you rename a resource.

## Rename Resource dialog box

The Rename Resource dialog box is where you rename a resource. Display this dialog box with Resource|Rename.

**Old Name display box**
The Old Name display box shows the resource's current name. You can't change the name in this box.

**New Name input box**
The New Name input box is where you enter the resource's new name.

**See Also**
Renaming a resource

## Resource | Memory Options

The Resource|Memory Options command brings up the Resource Memory Options dialog box, where you specify memory options for your resources.

## Resource Memory Options dialog box

The Resource Memory Options dialog box is where you specify how each resource in your project should be managed in memory. Display this dialog box with Resource|Memory Options.

**Resource Name display box**
The Resource Name display box displays the name of the resource. You can't change the name in this box.

**Memory Options check boxes**
The Memory Options check boxes specify a resource's load and memory options.

**See Also**
Specifying resource memory options

## Memory Options check boxes

The Memory Options check boxes specify how a resource is loaded and managed in memory. The name of the selected resource is displayed at the top of the dialog box.

**Note:** If you set memory options for an icon resource, those options apply to all the images in the resource.

**Load On Call check box**
The Load On Call check box loads the resource into memory only when it's needed. Choosing this option can reduce the amount of time required to load your program.

If you turn this option off, you activate Preload, which means that Windows will load the resource into memory when it first loads the program. Preload a resource only if you know Windows needs it as soon as the application begins to execute.

**Moveable check box**
The Moveable check box lets Windows move the resource segment in memory. If you turn this option off, the resource occupies a fixed location in memory.

**Discardable check box**
The Discardable check box lets Windows discard the resource segment from memory when it's no longer needed. Windows can load the resource into memory again when necessary.

If you turn this option off, you activate Nondiscardable, which means that Windows won't be able to remove the resource from memory while the application is running. If the Pure check box is also off, you can modify the resource from within your application.

**Pure check box**
The Pure check box prevents the resource segment in memory from being modified. Usually, you leave this option on.

## Resource | Identifiers

The Resource|Identifiers command brings up the Identifiers dialog box, where you add, edit, and delete identifiers.

## Identifiers dialog box

The Identifiers dialog box is where you add, edit, delete, or list identifiers. Display this dialog box with Resource|Identifiers. You can leave this dialog box open as you work in an editor.

**Name display box**
The Name display box displays the name of the identifier that's highlighted in the Identifiers list. You can't enter this name; you need to pick it from the Identifiers list.

**Value display box**
The Value display box displays the ID value associated with the selected identifier.

**Identifiers list**
The Identifiers list shows identifiers.

**Usage box**
The Usage box shows the type and name of the resource whose identifier is highlighted in the Identifiers list. If the highlighted identifier is not associated with a resource, the Usage box says (unused).

**View radio buttons**
The View radio buttons control which identifiers Resource Workshop displays.

**Show Predefined check box**
The Show Predefined check box displays the predefined #defines that are included with Windows, in addition to the identifiers that are defined in the current project.

**New button**
The New button displays the New Identifier dialog box, where you add a new identifier.

**Change button**
The Change button brings up the Change Identifier Value dialog box, where you change an identifier's value

**Delete button**
The Delete button deletes the selected identifier.

**See Also**
Adding identifiers
Deleting identifiers
Editing identifiers
Identifiers and identifier files
Listing identifiers

## View radio buttons

The View radio buttons determine which identifier names are displayed in the Identifiers list.

**All**
Turn on the All radio button if you want to display all identifiers in all files in the current project.

**Single File**
Turn on the Single File radio button if you want to display identifiers that are in a specific file (project file and header or include files) associated with the current project. Choose the file name from the drop-down box to the right of the radio button.

## New Identifier dialog box

The New Identifier dialog box is where you add a new identifier. This dialog box pops up whenever you create a new identifier in a resource editor.

**Name input box**
The Name input box is where you enter the name of the new identifier.

**Value input box**
The Value input box is where you enter the resource ID you want to assign to the identifier.

**File drop-down box**
The File drop-down box is where you choose the file in which to place the identifier. You'll usually pick the identifier file for the current project.

**See Also**
Adding identifiers
Identifiers and identifier files
Renaming a resource

## Change Identifier Value dialog box

The Change Identifier Value dialog box is where you change an identifier's value. Display this dialog box by choosing the Change button in the Identifiers dialog box.

**Name display box**
The Name display box shows the identifier's name. You can't change the name in this box.

**Old Value display box**
The Old Value display value shows the identifier's current value. You can't change the name in this box.

**New Value input box**
The New Value input box is where you enter the new value for the identifier.

**See Also**
Editing identifiers
Identifiers and identifier files

## View menu

The View menu is displayed when you open a project or resource. Use this command to determine how the Project window displays information. (This is different from the View menu that's displayed when you're editing a menu or the View menu that's displayed when you're using the Paint editor.)

Here are the View menu commands:
By Type
By File
Show Identifiers
Show Resources
Show Items
Show Unused Types

## View | By File

Use the View|By File command to group resources according to the file they're in. This command displays all resources and file names in the Project window in the order in which they appear in the source files.

Resource Workshop indicates whether resources are saved in the project file or in an external file.

## View | By Type

The View|By Type menu command groups resources according to type, such as icon, menu, and dialog. Choose this command to see all resources listed in the Project window according to type instead of file name. This command does not indicate if resources are in an external file.

Resources are listed in this order:
Bitmaps
Menus
Dialogs
String tables
Accelerators
RCDATA
Cursors
Icons
User-defined resources

After the list of resources, any identifiers in the project are listed if you've turned on the View|Show Identifiers command.

## View | Show Identifiers

The View|Show Identifiers command displays any identifiers in the project. If you don't want to see the identifiers displayed in the Project window, turn this command off.

**See Also**
Identifiers and identifier files

## View | Show Resources

The View|Show Resources command lists the individual names of resources. Turn this command off to see just file names in the Project window without an explanation of the resource types contained in those files.

**See Also**
Resources overview

## View | Show Items

The View|Show Items command shows another level of detail in the Project window. When this command is on, you can see items within individual resources (for example, POPUPs and MENUITEMs defined in a menu resource).

## View | Show Unused Types

The View|Show Unused Types command displays all possible types of resources in the Project window, even if some of them are not included in the project.

Turn on View|By Type for this option to be useful.

## Window menu

The Window menu contains commands for manipulating and opening windows.

Most of the windows you open from this menu have all the standard window elements, such as scroll bars, Minimize and Maximize buttons, and a Control menu box.

At the bottom of the Window menu is a list of open windows. If there is more than one, you can switch to another window and make it active by selecting it from the list.

These are the Window menu commands:
Tile
Cascade

## Window | Tile

Choose Window|Tile to tile your open windows.

This option arranges the windows so they cover the entire desktop without overlapping one another.

## Window | Cascade

Choose Window|Cascade to stack all open edit windows.

This option overlaps open windows so each is the same size as all others and only part of each underlying window is visible.

## List of open windows

At the bottom of the Window menu is a list of open windows.

If more than one window is open, you can switch to another window and make it active by choosing it (either click it in the list, or type the number associated with the window you want).

## Help menu

The Help menu provides access to online Help, which comes up in a special Help window.
The Help system provides information on virtually all aspects of Resource Workshop.

(Also, don't forget that one-line menu and dialog-box hints appear on the status line whenever you select a menu command or dialog box item.)

These are the Help menu commands:
Contents
Using Help
Getting Started
Projects
Resources
Using Menus
Error Messages
Resource Script
About Resource Workshop

## Help | Contents (F1)

Choose Help|Contents to see Resource Workshop's online Help Contents screen. This screen summarizes the organization and contents of the Help system.

To get into the Help system from the Contents screen, choose one of the icons, highlighted words, or highlighted phrases: click it, or Tab to it and then press Enter.

If you don't know how to use a Help system under Windows, choose the Help|Using Help menu command.

**See Also**
Help contents

## Help | Using Help

The Help|Using Help command displays information on how to use Resource Workshop's Help system (or any other Help system under Windows).

## Help | Getting Started

Help|Getting Started explains what to do if you're a first time Resource Workshop user.

**See Also**
How to start using Resource Workshop

## Help | Projects

Help|Projects displays information on Resource Workshop projects. You learn what a project is and how to create one.

**See Also**
Projects overview

## Help | Resources

Help|Resources displays information on Resource Workshop resources. You learn how to create new resources and edit existing ones.

**See Also**
Resources overview

## Help | Using Menus

Choose Help|Using Menus to see an alphabetical listing of Resource Workshop menus. You can go to the appropriate Help screen to find out what each menu is used for.

**See Also**
Menu commands alphabetical listing

## Help | Error Messages

Choose Help|Error Messages to see an alphabetical listing of all Resource Workshop error messages.

**See Also**
Alphabetical listing of errors

## Help | Resource Script

Choose Help|Resource Script to see an overview explanation of the Resource Script Language.

**See Also**
Resource Script overview

## Help | About Resource Workshop

Choose Help|About Resource Workshop to see copyright and version information.

Click OK or press Esc, Alt+Spacebar+C, or Alt+F4 to close the box.

## Control menu

The Control menu box is on the far left of the desktop and window title bar. Click it once or press Alt+Spacebar to display the menu.

The commands that appear on this menu affect the Resource Workshop desktop or the current window. The Control menu commands are:
Restore
Move
Size
Minimize
Maximize
Close
Next
Switch To

## Control | Restore

Use the Control|Restore command to return the desktop or window to its previous size.

This command is available only if the desktop or window is maximized or minimized.

## Control | Move

Use the Control|Move command to move the desktop or window. Use the arrow cursor keys to move it, and press Enter when you are done. You can also move the desktop or window by dragging the title bar.

You can't use this command if the desktop or window is maximized.

## Control | Size

Use Control|Size to change the size of the desktop or window with the keyboard. Use the arrow cursor keys to move the borders. Press Enter when you are satisfied with the size.

You can't use this command if the desktop or window is maximized.

## Control | Minimize

**Desktop**
Use Control|Minimize command to turn the desktop into the Resource Workshop icon. This command is available only if the desktop is not already minimized.

**Window**
Use Control|Minimize command to minimize the window and return to the Project window. This command is available only if the window is not already minimized.

## Control | Maximize

Use Control|Maximize to make the desktop or window fill the entire screen. This command is available only if the desktop or window is not already maximized.

## Control | Close

**Desktop**
Use the Control|Close command to close the desktop and unload Resource Workshop from memory. If you've modified a project, a dialog box appears asking you if you want to save the file before closing.

**Window**
Use the Control|Close command to close the window. If you've modified the resource, Resource Workshop automatically saves changes.

If you're editing the resource as text, you see a dialog box with the message: "Resource has changed. Compile?" (If you click No, you lose your changes.) You can also use the Compile menu to compile your resource before you exit the text editor.

## Control | Next

Control|Next is displayed only in the window Control menu. This command displays the next Resource Workshop window.

## Control | Switch To

Control|Switch To is displayed only in the desktop Control menu. This command displays the Task List dialog box, where you can switch from one application to another and rearrange application windows.

## Task List dialog box

The Task List dialog box lists all open applications.

If the application you want is in the list box, double-click on it to choose it.

From the keyboard:

- Press the Up or Down arrow keys to reach the application you want.
- Press Enter to choose it.

**See Also**
Action buttons

## Action buttons

Action buttons are used in the Task List dialog box to switch to the selected application and rearrange application windows.

- With the mouse, click the action button you want.
- If you're using the keyboard, press Tab to go to the action button, then press Enter.

**Switch To**
The Switch To button switches to another application in the list box.

**End Task**
The End Task button closes the selected application in the list box.

**Cascade**
The Cascade button arranges the application windows in an overlap style.

**Tile**
The Tile button arranges the application windows in a tile fashion (side-by-side).

**Arrange Icons**
The Arrange Icons button evenly spaces the icons across the bottom of the desktop.

## Compile menu

The Compile menu is displayed when you edit a resource's source script with the text editor.

The Compile menu command is Compile|Compile Now.

## Compile | Compile Now (Alt+F9)

Use the Compile|Compile Now command to compile the resource's source script when you edit it.

If there's a compilation error, Resource Workshop displays an error message. Read the error message and click OK when you're done. Resource Workshop puts you back in the text editor so you can fix the error.

## Menu menu

The Menu menu is displayed when you create a menu. You use this menu to add pop-up commands, menu items, and menu separators to your menus. You can also insert a standard File, Edit, or Help pop-up menu and test your menu for duplicate IDs.

Here are the commands:
New Popup
New Menu Item
New Separator
New File Popup
New Edit Popup
New Help Popup
Check Duplicates

## Menu | New Popup (Ctrl+P)

The Menu|New Popup command inserts a new pop-up menu below the line that's currently highlighted in the Outline pane.

## Menu | New Menu Item (Ins)

The Menu|New Menu Item command inserts a new menu item below the line that's currently highlighted in the Outline pane.

## Menu | New Separator (Ctrl+S)

The Menu|New Separator command inserts a new separator below the line that's currently highlighted in the Outline pane.

## Menu | New File Popup

The Menu|New File Popup command inserts a standard file pop-up menu below the line that's currently highlighted in the Outline pane.

The menu items on the File pop-up are:

- New
- Open
- Save
- Save as
- Print
- Page Setup
- Printer Setup
- Exit

## Menu | New Edit Popup

The Menu|New Edit Popup command inserts a standard edit pop-up menu below the line that's currently highlighted in the Outline pane.

The menu items on the Edit pop-up are:

- Cut
- Copy
- Paste

## Menu | New Help Popup

The Menu|New Help Popup command inserts a standard help pop-up menu below the line that's currently highlighted in the Outline pane.

The menu items on the Help pop-up are:

- Index
- Keyboard
- Commands
- Procedures
- Using Help
- About

## Menu | Check Duplicates

Use the Menu|Check Duplicates command to test your menu. The test menu is updated as you make changes to your menu, so you can test changes as often as you want.

If there are duplicates, the Menu editor displays a message box with the message "Duplicate command value found." When you close this box, the Menu editor highlights the statement in the Outline pane that contains the duplicate value. You can type a new value in the Item ID input box.

## View menu (Menu editor)

The View menu is displayed when you create a menu. Use this command to change the view of the Menu editor. (This is different from the View menu that's displayed when you open a project or resource or the View menu that's displayed when you use the Paint editor.)

Here are the View menu commands:
View as Popup
First Graphic
Second Graphic

**See Also**
Using the Menu editor

## View | View as Popup

The View|View as Popup command controls whether the test menu is displayed on the menu bar or in a pop-up menu. This option is off by default, and the pop-up commands in the Test Menu pane are displayed across the menu bar.

Leave this option off if your menu resource contains the application's entire menu structure and you want it displayed as it would appear to the user.

If you're working on a floating menu, check this option to display the test menu as it would actually appear. The command Pop-up displays on the menu bar, and you choose Pop-up to display the menu itself.

**See Also**
Test Menu pane

## View | First Graphic

The View|First Graphic command represents the default configuration of the three panes.

It puts the Attribute pane on the left, the Test Menu pane on the upper right, and the Outline pane on the lower right.

**See Also**
Attribute pane
Outline pane
Test Menu pane

## View | Second Graphic

The View|Second Graphic command is the alternative configuration of the three panes.

It puts the Test Menu pane across the top of the Menu editor, like a normal menu bar. The Attribute and Outline panes are underneath.

**See Also**
Attribute pane
Outline pane
Test Menu pane

## Accelerator menu

The Accelerator menu is displayed when you create or edit an accelerator. Here are the Accelerator menu commands:
New Item
Key Value
Check Dup Keys

## Accelerator | New Item (Ins)

The Accelerator|New Item command inserts a new line in the Outline pane below the line that's currently highlighted.

You'll see two zeros in the outline and in the Command and Key input boxes in the Attribute pane.

**See Also**
Customizing an accelerator table

## Accelerator | Key Value

The Accelerator|Key Value command is where you specify the key combination for users to press to invoke the accelerator. This command puts you in Key Value mode.

When you select this command, the Outline pane is replaced by instructions. The instructions tell you to enter the accelerator keys on the keyboard and click the mouse when you're done.

## Accelerator | Check Dup Keys

The Accelerator|Check Dup Keys command debugs your accelerator table by searching for duplicate key combinations.

If two accelerators use the same key combination, the Accelerator editor displays a message and highlights the second accelerator.

Make your changes and continue debugging your accelerator table with Accelerator|Check Dup Keys until you see an "OK" message.

## Stringtable menu

The Stringtable menu is displayed when you edit strings and string tables. You use this menu to add and delete string table items.

Here are the Stringtable menu options:
New Item
Delete Item
Change Item
Restore Item

**See Also**
Using the String editor

## Stringtable | New Item (Ins)

The Stringtable|New Item command inserts a new line below the line that's currently highlighted.

## Stringtable | Delete Item

The Stringtable|Delete Item command deletes the line that's currently highlighted.

## Stringtable | Change Item (Enter)

Use the Stringtable|Change Item command to edit the line that's currently highlighted. You can change the values in the ID Source and String input boxes.

## Stringtable | Restore Item (Esc)

The Stringtable|Restore Item command undoes your most recent changes.

## Control menu (Dialog editor)

The Control menu is displayed when you create a dialog box. (This is different from the Control menu box that's displayed on the left side of the menu bar.)

You use the Control menu to put controls in your dialog box. Here are the Control menu commands:
Style
Push Button
Radio Button
Horizontal Scroll Bar
Vertical Scroll Bar
List Box
Check Box
Group Box
Combo Box
Edit Text
Static Text
Icon
Black Frame
Black Rectangle
Custom

## Control | Style

The Control|Style command opens one of the Style dialog boxes, where you customize controls or the dialog box you're creating. This command is available after you select a control or the dialog box.

**See Also**
Button Style dialog box
Combo Box Style dialog box
Edit Text Style dialog box
List Box Style dialog box
Scroll Bar Style dialog box
Static Style dialog box
Window Style dialog box

## Control | Push Button

The Control|Push Button command turns the cursor into the Push Button tool. The Push Button tool puts a rectangular button in your dialog box. The user "presses" the button to select an action. Push buttons always contain text, so you need to specify a caption for each one.

Click in the dialog box to place the button. Double-click in the button to bring up the Button Style dialog box, where you customize the button.

**See Also**
Button controls

## Control | Radio Button

The Control|Radio Button command turns the cursor into the Radio Button tool. The Radio Button tool puts a circular button in your dialog box with text on its left or right side. When the button is turned on, a solid dot fills the circle. Radio buttons are used in groups to represent related but mutually exclusive options.

Click in the dialog box to place the button. Double-click in the button to bring up the Button Style dialog box, where you customize the button.

**See Also**
Button controls

## Control | Horizontal Scroll Bar

The Control|Horizontal Scroll Bar command turns the cursor into the Horizontal Scroll Bar tool. The Horizontal Scroll Bar tool puts a horizontal rectangle in your dialog box with a direction arrow on each end.

Click in the dialog box to place the scroll bar. Double-click in the scroll bar to bring up the Scroll Bar Style dialog box, where you customize the scroll bar.

**See Also**
Scroll bar controls

## Control | Vertical Scroll Bar

The Control|Vertical Scroll Bar command turns the cursor into the Vertical Scroll Bar tool. The Vertical Scroll Bar tool puts a vertical rectangle in your dialog box with a direction arrow on each end.

Click in the dialog box to place the scroll bar. Double-click in the scroll bar to bring up the Scroll Bar Style dialog box, where you customize the scroll bar.

**See Also**
Scroll bar controls

## Control | List Box

The Control|List Box command turns the cursor into the List Box tool. The List Box tool puts a rectangle in your dialog box. The rectangle usually includes a list of strings. It can also contain a visual representation of the list.

Usually, a user can browse through the list box, then select one or more items.

Click in the dialog box to place the list box. Double-click in the list box to bring up the List Box Style dialog box, where you customize the list box.

**See Also**
List Box controls

## Control | Check Box

The Control|Check Box command turns the cursor into the Check Box tool. The Check Box tool puts a rectangular button in your dialog box with text to its left or right.

Click in the dialog box to place the check box. Double-click in the check box to bring up the Button Style dialog box, where you customize the check box.

**See Also**
Button controls

## Control | Group Box

The Control|Group Box command turns the cursor into the Group Box tool. The Group Box tool puts a rectangular box around a group of controls in your dialog box. It's used to visually group controls together. You can include a caption in the upper left corner of the group box.

Click in the dialog box to place the group box. Double-click in the group box to bring up the Button Style dialog box, where you customize the group box.

**See Also**
Button controls

## Control | Combo Box

The Control|Combo Box command turns the cursor into the Combo Box tool. The Combo Box tool puts a box in your dialog box that's a combination of a list box and a static control or an edit text control.

Click in the dialog box to place the combo box. Double-click in the combo box to bring up the Combo Box Style dialog box, where you customize the combo box.

**See Also**
Combo Box controls

## Control | Edit Text

The Control|Edit Text command turns the cursor into the Edit Text tool. The Edit Text tool puts a rectangle in your dialog box. The user can enter text from the keyboard into this box.

Place the edit text in your dialog box by moving the cursor. Double-click to bring up the Edit Text Style dialog box, where you customize the text.

**See Also**
Edit Text controls

## Control | Static Text

The Control|Static Text command turns the cursor into the Text Static tool. The Text Static tool puts text in your dialog box.

Click in the dialog box to place the static text. Double-click in the static text to bring up the Static Style dialog box, where you customize the static text.

**See Also**
Static controls

## Control | Icon

The Control|Icon command turns the cursor into the Icon tool. The Icon tool puts an icon in your dialog box.

Click in the dialog box to place the icon. Double-click in the icon to bring up the Static Style dialog box, where you customize the icon.

**See Also**
Static controls

## Control | Black Frame

The Control|Black Frame command turns the cursor into the Black Frame tool. The Black Frame tool puts a rectangular empty frame in your dialog box. The frame is the color of the current window frame.

Click in the dialog box to place the frame. Double-click in the group box to bring up the Static Style dialog box, where you customize the frame.

**See Also**
Static controls

## Control | Black Rectangle

The Control|Black Rectangle command turns the cursor into the Black Rectangle tool. The Black Rectangle tool puts a rectangle in your dialog box that's the same color as the current window frame.

Click in the dialog box to place the rectangle. Double-click in the rectangle to bring up the Static Style dialog box, where you customize the rectangle.

**See Also**
Static controls

## Control | Custom

The Control|Custom command displays the New Custom Control dialog box. You use this dialog box to select a control that doesn't fit into any of the predefined Window types.

**See Also**
Adding a custom control
Installing a custom control library

## Align menu

The Align menu is displayed when you create a dialog box. It lets you align the controls in your dialog box. Here are the Align menu commands:

Align
Size
Array
Grid

## Align | Align

The Align|Align commands brings up the Align Controls dialog box where you align controls. This command is available only if you select two or more controls.

## Align Controls dialog box

The Align Controls dialog box is where you align controls. Display this dialog box with the Align|Align command.

You can use radio buttons or the keyboard to align controls. To use the keyboard, select the controls in a selection frame. Then press Ctrl + the shortcut key. To use the shortcut for the Space Equally (Horizontal or Vertical) radio button, press Ctrl + the left mouse button and drag a corner of the selection frame.

**Horizontal Alignment radio buttons**
The Horizontal Alignment radio buttons align the selected controls horizontally.

**Vertical Alignment radio buttons**
The Vertical Alignment radio buttons align the selected controls vertically.

**See Also**
Selecting multiple controls

## Vertical Alignment radio buttons

Turn on one of the Vertical Alignment radio buttons to move the selected controls vertically. Most options are enabled only if you select two or more controls first.

**No Change (Ctrl)**
Turn on the No Change radio button if you don't want the controls to move vertically.

**Tops (T)**
Turn on the Tops radio button to align the controls so their tops are at the top of the selection frame.

**Centers (V)**
Turn on the Centers radio button to align the controls so their vertical centers are at the center of the selection frame.

**Bottoms (B)**
Turn on the Bottoms radio button to align the controls so their bottoms are at the bottom of the selection frame.

**Space Equally (Strch)**
Turn on the Space Equally radio button to move controls vertically so they are spaced evenly within the selection frame.

**Center in Dialog (C)**
Turn on the Controls radio button to move the selection frame vertically so that it's centered in the dialog box. The relative positions of the individual controls within the selection frame is unchanged.

## Horizontal Alignment radio buttons

Turn on one or more of the Horizontal Alignment radio buttons to move the selected controls horizontally. Most options are enabled only if you select two or more controls first.

**No Change (Ctrl)**
Turn on the No Change radio button if you don't want the controls to move horizontally.

**Left Sides (L)**
Turn on the Left Sides radio button to align controls so their left sides are on the left side of the selection frame.

**Centers (H)**
Turn on the Centers radio button to align controls so their horizontal centers are in the center of the selection frame.

**Right Sides (R)**
Turn on the Right Sides radio button to align controls so their right sides are on the right side of the selection frame.

**Space Equally (Strch)**
Turn on the Space Equally radio button to move controls horizontally so they are spaced evenly within the selection frame.

**Center in Dialog (D)**
Turn on the Center in Dialog radio button to move the selection frame horizontally so that it's centered in the dialog box. The relative positions of the individual controls within the selection frame is unchanged.

## Align | Size

Depending on whether you select controls or the entire dialog box, the Align|Size command brings up the Size Controls dialog box or the Size Dialog dialog box where you resize groups of controls or the dialog box itself.

This command is available only if you select one or more controls or the dialog box. Select the dialog box by clicking the dialog box title bar.

## Size Controls dialog box

The Size Controls dialog box is where you specify the size of controls in the dialog box. To display this dialog box, select a group of controls and choose Align|Size.

You can enter values only if you select one control and then turn on the Enter Values radio buttons. Values are entered in dialog units (the units used to specify a dialog window).

Dialog units are calculated from the height and width of the dialog's font.

- Horizontal dialog units are 1/4 the width of a character in the dialog's font.
- Vertical dialog units are 1/8 the height of a character in the dialog's font.

**Horizontal Size radio buttons**
The Horizontal radio buttons size the width of the selected controls.

**Vertical Size radio buttons**
The Vertical radio buttons size the height of the selected controls.

**X input box**
The X input box is where you enter the distance from the upper left corner of the control to the upper left corner of the dialog box (directly below the title bar).

**CX input box**
The CX input box is where you enter the width of the control.

**Y input box**
The Y input box is where you enter the distance from the upper left corner of the control to the upper left corner of the dialog box (directly below the title bar).

**CY input box**
The CY input box is where you enter the height of the control.

**See Also**
Selecting multiple controls

## Horizontal Size radio buttons

Turn on one of the Horizontal Size radio buttons to size the width of the selected controls. Most options are enabled only if you select two or more controls first.

**No Change**
Turn on the No Change radio button if you don't want the controls to change size horizontally.

**Shrink to Smallest**
Turn on the Shrink to Smallest radio button to reduce the width of controls to match the smallest control in the selected group.

**Grow to Largest**
Turn on the Grow to Largest radio button to increase the width of controls to match the largest control in the selected group.

**Width of Size Box**
Turn on the Width of Size Box radio button to resize controls so that they're as wide as the selection frame.

**Width of Dialog**
Turn on the Width of Dialog radio button to resize controls so that they're as wide as the dialog box.

**Enter Values**
Turn on the Enter Values radio button to specify a control's X and CX values, the control's width. This radio button is active only if you select a single control.

## Vertical Size radio buttons

Turn on one of the Vertical Size radio buttons to size the height of the selected controls. Most options are enabled only if you select two or more controls first.

**No Change**
Turn on the No Change radio button if you don't want the controls to change size vertically.

**Shrink to Smallest**
Turn on the Shrink to Smallest radio button to reduce the height of controls to match the smallest control in the selected group.

**Grow to Largest**
Turn on the Grow to Largest radio button to increase the height of controls to match the largest control in the selected group.

**Height of Size Box**
Turn on the Height of Size Box radio button to resize controls so that they're as tall as the selection frame.

**Height of Dialog**
Turn on the Height of Dialog radio button to resize controls so that they're as tall as the dialog box.

**Enter Value**
Turn on the Enter Value radio button to specify a control's X and CX values, the control's height. This radio button is active only if you select a single control.

## Size Dialog dialog box

The Size Dialog dialog box is where you specify the starting point, the height, and the width of the dialog box. To display this dialog box, select the dialog box and choose Align|Size.

Values are entered in dialog units (the units used to specify a dialog window). Dialog units are calculated from the height and width of the dialog's font.

- Horizontal dialog units are 1/4 the width of a character in the dialog's font.
- Vertical dialog units are 1/8 the height of a character in the dialog's font.

**Horizontal Size radio buttons**
The Horizontal radio buttons size the width of the dialog box.

**Vertical Size radio buttons**
The Vertical radio buttons size the height of the dialog box.

**X input box**
The X input box is where you enter the upper left corner of the dialog box.

**CX input box**
The CX input box is where you enter the width of the dialog box.

**Y input box**
The Y input box is where you enter the upper left corner of the dialog box.

**CY input box**
The CY input box is where you enter the height of the dialog box.

## Horizontal Size radio buttons

Turn on one of the Horizontal Size radio buttons to size the width of the dialog box.

**No Change**
Turn on the No Change radio button if you don't want the dialog box to change size horizontally.

**Set by Windows**
The Set by Windows radio button lets Windows position your dialog box. This radio button is active only when you select the dialog box. When you turn this radio button on, and the X input box is disabled. This option is generally used for dialog boxes that are main windows.

**Enter Values**
Turn on the Enter Values radio button to specify the dialog box's X and CX values.

## Vertical Size radio buttons

Turn on one of the Vertical Size radio buttons to size the height of the dialog box.

**No Change**
Turn on the No Change radio button if you don't want the dialog box to change size vertically.

**Enter Values**
Turn on the Enter Value radio button to specify a dialog box's Y and CY coordinates.

## Align | Array

The Align|Array command brings up the Form Controls Into An Array dialog box where you format controls. This command also numbers each control as a member of the array group.

Align|Array is only available if you select two or more controls.

## Form Controls Into An Array dialog box

The Form Controls Into An Array dialog box is where you form groups of controls into arrays of rows and columns. To display this dialog box, select two or more controls and choose Align|Array.

**Array Layout input boxes**
The Array Layout input boxes are where you arrange a group of controls.

**Rows input box**
Enter the number of rows in which you want to arrange the controls in the Rows input box.

**Columns input box**
Enter the number of columns in which you want to arrange the controls in the Column input box.

**Order radio buttons**
The Order radio buttons order the controls in the group.

**Left to Right**
Turn on the Left to Right radio button to order the controls from left to right.

**Top to Bottom**
Turn on the Top to Bottom radio button to order the controls from top to bottom.

**See Also**
Selecting multiple controls

## Align | Grid

The Align|Grid command brings up the Set Grid Attributes dialog box, where you define a grid to help align controls.

## Set Grid Attributes dialog box

The Set Grid Attributes dialog box is where you specify a grid. Display this dialog box with the Align| Grid command.

**Width input box**
The Width input box specifies the width of each cell in the grid.

**Height input box**
The Height input box specifies the height of each cell in the grid.

**Grid Type radio buttons**
The Grid Type radio buttons determine how selected controls move relative to the grid.

> **Absolute**
> Turn on the Absolute radio button if you want the controls to snap to the nearest grid line.
>
> **Relative**
> Turn on the Relative radio button to move the control in increments equal to the current Width and Height values.

**Show Grid check box**
The Show Grid check box displays the grid.

## Options menu (Dialog editor)

The Options menu lets you choose options for the Dialog editor. (This is different from the Options menu that's displayed when you edit a bitmapped resource.)

Here are the Options menus commands:
Hide Tools
Hide Alignment
Hide Caption
Modify Controls
Set Tabs
Set Groups
Set Order
Test Dialog
Preferences
Install Control Library
Redraw Now

## Options | Hide Tools

The Options|Hide Tools command hides the Tools palette. This command is a toggle - when the Tools palette is hidden, the command changes to Show Tools.

## Options | Hide Alignment

The Options|Hide Alignment command hides the Alignment palette. This command is a toggle - when the Alignment palette is hidden, the command changes to Show Align.

## Options | Hide Caption

The Options|Hide Caption command hides the Caption window. This command is a toggle - when the Caption window is hidden, the command changes to Show Caption.

## Options | Modify Controls

The Options|Modify Controls command puts you in edit mode, where you modify the controls in your dialog box.

**See Also**
Working with controls

## Options | Set Tabs

The Options|Set Tabs command assigns a tab stop to a control. Users can press Tab to move to that control.

When you choose Options|Set Tabs, the cursor turns into the Tab Set tool. Resource Workshop surrounds any controls currently set as tab stops with a gray box.

To change the Tab Stop attribute for a control, put the cross hair of the Tab Set tool in the control and click. The Set Tabs option works as a toggle, turning the Tab Stop check box on or off.

When you finish setting tab stops, choose Options|Modify Controls to return to edit mode.

## Options | Set Groups

The Options|Set Groups command defines groups of controls in which users can move around using the arrow keys. You usually assign the first control in the group to a tab stop with the Options|Set Tabs command.

Here's how to define groups of controls:

1. If necessary, move the controls so that they're together.

2. Choose Options|Set Groups. The cursor turns into the Set Groups tool.

3. For each group you want to define, click the first member so it's surrounded by a gray box.

- If any control that you don't want to define as the first member of a group is surrounded by a gray box, click the gray box to remove it from the control.
- You don't have to identify the last member of a group. By clicking the first member of each group, you also identify the last member of the previous group.

When you finish identifying the first member in each group, choose Options|Modify Controls to return to edit mode.

## Options | Set Order

The Options|Set Order command specifies the order of controls in your dialog box. Here's how you use it:

1. Choose the controls whose order you want to change. You can click and drag to select a group of controls. You don't need to select any controls if you want to specify the order for all controls in the dialog box.

2. Choose Options|Set Order. The cursor turns into the Set Order tool.

- Each control is numbered to show its current place in the overall order. If you chose just some of the controls in step 1, you see the order numbers only for those controls.
- The Next Item prompt at the bottom of the Dialog editor tells you the order number that Resource Workshop assigns to the next control you click.

3. Click the items you want to assign new order numbers to. The Dialog editor displays a gray box around all the controls you've already picked. Once you've clicked a control to assign it an order number, you can't pick the same control again.

- If you make a mistake and want to go back to the original order, just click on all the controls. Then choose Edit|Undo (or press Alt+Backspace).

When you finish assigning new order numbers, choose Options|Modify Controls to return to edit mode.

## Options | Test Dialog

Use the Options|Test Dialog command to test how the controls in your dialog box work. You can test how the Tab and arrow keys move you around, or how text you type is scrolled in an edit text control. Choose Options|Test Dialog to leave test mode.

When you test a dialog box, the status line at the bottom of the Dialog editor says Test.

**See Also**
Testing a dialog box

## Options | Preferences

The Options|Preferences command displays the Preferences dialog box where you set preferences for the way the Dialog editor works.

## Preferences dialog box

The Preferences dialog box is where you customize the Dialog editor. Display this dialog box with the Options|Preferences command.

**Status Line Units radio buttons**
The Status Line Units radio buttons determine the unit of measurement the status line uses to display information.

**Selection Border radio buttons**
The Selection Border radio buttons choose how the frame around the selected control is displayed.

**Drawing Type radio buttons**
The Drawing Type radio buttons determine how elements of your dialog box are displayed in the Dialog editor.

**Selection Options check boxes**
The Selection Options check boxes choose how the selection feature behaves when you select controls to customize.

**Generate CONTROL Statements Only check box**
The Generate CONTROL Statements Only check box generates CONTROL statements instead of specialized dialog control statements, to define a control. For example, if you're adding a list box to your dialog box and you choose this option, Resource Workshop generates a CONTROL statement instead of a LISTBOX statement.

**Draw Custom Controls As Frames check box**
The Draw Custom Controls as Frames check box is available only when the Normal radio button (one of the Drawing Type radio buttons) is on. When Draw Custom Controls As Frames is on, custom controls are drawn as empty rectangular outlines. When the option is off, custom controls are drawn as gray rectangles with their text in a white rectangle in the center. Drawing custom controls as frames can speed up drawing of your dialog boxes on the screen.

## Status Line Units radio buttons

The Status Line Units radio buttons determine the unit of measurement the status line uses to display information. You can display measurements using screen units, but the Dialog editor always uses dialog unit measurements.

**Dialog**
Turn on the Dialog radio button if you want the status line units to display in dialog units.

**Screen**
Turn on the Screen radio button if you want the status line units to display in pixels.

## Selection Border radio buttons

The Selection Border radio buttons choose how the frame around the selected control is displayed. You can resize and move objects no matter which option you choose.

**Thick Frame**
Turn on the Thick Frame radio button to display a thick frame around the selected control.

**Handle**
Turn on the Handle radio button to display sizing handles at each corner and at the midpoint of each side on the selected control.

## Drawing Type radio buttons

The Drawing Type radio buttons determine how elements of your dialog box are displayed in the Dialog editor.

**Draft**
Turn the Draft radio button on to draw each control as a rectangle with its control ID in the center. This option also lets you see how much space is occupied by the control's selection frame.

**Normal**
Turn the Normal radio button on to draw standard Windows controls as they will appear at run time.

**WYSIWYG**
Turn on the WYSIWYG radio button to have Resource Workshop create the dialog and control child windows. The controls, including custom controls, draw themselves. This option is slower, but the most accurate. It's the default.

## Selection Options check boxes

The Selection Options check boxes "set the rules" for selecting controls. When you work with closely spaced controls, you might want to turn these options on for greater precision.

**Select Near Border check box**
If you turn the Select Near Border check box on, you must click on the control's border to select the control. If this check box is off, you can click anywhere inside the control's border.

**Selection Rectangle Surrounds check box**
Turn the Selection Rectangle Surrounds check box on if you want to entirely surround a control (or controls) with the selection frame. If you turn this option off, the selection rectangle only needs to touch the control (or controls).

## Options | Install Control Library

The Options|Install Control Library command displays the Install a New Control Library dialog box, where you can install a new control library.

## Install a New Control Library dialog box

The Install a New Control Library dialog box is where you install a dynamic link library file. A .DLL file contains a custom control library. Display this dialog box with Options|Install Control Library.

**File input box**
Enter the name of the dynamic link library file in the File input box. The current directory path is displayed below the input box.

**Files drop-down box**
Choose dynamic link library (.DLL) from the Files drop-down box. Just select the file name and double-click it.

**Directories drop-down box**
Choose the directory from the Directories drop-down box. You can change the directory path by choosing a new directory in the Directories drop-down box. Just select the directory name and double-click it. The new path is displayed below the File input box.

When the .DLL file is installed, the custom controls in that .DLL will be available just like any standard Windows control.

**See Also**
Installing a custom control library

## Options | Redraw Now

The Options|Redraw Now command redraws the dialog box.

## View menu (Paint editor)

The View menu is displayed when you use the Paint editor. (This is different from the View menu that's displayed when you open a project or resource or the View menu that's displayed you're editing a menu.)

Use this menu to choose how your image is displayed. Here are the View menu commands:
Zoom In
Zoom Out
Actual Size
CGA Resolution [32 x 16]
Split Horizontal
Split Vertical

## View | Zoom In (Ctrl+Z)

The View|Zoom In command zooms the entire image in the currently selected window. Resource Workshop increases magnification to the next level: 400%, 800%, or 1600%.

The Paint editor uses the center of the image as a reference when zooming the entire image.

**See Also**
Using the zoom accelerators
Zooming images

## View | Zoom Out (Ctrl+O)

The View|Zoom Out command zooms out the entire image in the currently selected window. Resource Workshop decreases magnification to the next level: 800%, 400%, or 100%.

The Paint editor uses the center of the image as a reference when zooming the entire image.

**See Also**
Using the zoom accelerators
Zooming images

## View | Actual Size (Ctrl+A)

The View|Actual Size command returns a zoomed image to its actual size.

**See Also**
Using the zoom accelerators
Zooming images

## View | CGA Resolution [32 x 16]

When you're editing cursors and icons, the View|CGA Resolution [32 x 16] command is displayed. (It's not available when you edit fonts and bitmaps.)

This command changes, based on the current resolution:

- If you're creating a resource in 32 x 32 resolution, the command View|CGA Resolution [32 x 16] lets you see how it would look on a CGA screen.
- If you're creating a resource in 32 x 16 (CGA) resolution, the command View|EGA/VGA Resolution [32 x 32] lets you see how it would look on an EGA or VGA screen.
- If you're creating a resource in 64 x 64 resolution, both commands are available. You can see how it would look on a CGA screen or on an EGA or VGA screen.

To get out of any of these view modes, choose View|Actual Size.

All you change with this command is the view of the icon or cursor - if it was a 32 x 32 pixel icon or cursor, it still is. You don't change its size.

## View | Split Horizontal

Use View|Split Horizontal to split the Paint editor screen horizontally. This displays one window pane on top of the other.

This command is available only if the screen is split vertically.

**See Also**
View|Split Vertical

## View | Split Vertical

Use View|Split Vertical to split the Paint editor screen vertically. This displays the two window panes side-by-side.

This command is available only if the screen is split horizontally.

**See Also**
View|Split Horizontal

## Text menu

The Text menu is displayed when you use the Paint editor. You use this menu to choose how the text in your resource is displayed.

Here are the Text menu commands:
Align Left
Align Center
Align Right
Font

## Text | Align Left

The Text|Align Left command aligns text to the left of where you initially clicked before typing the text.

For example, if you choose Text|Align Left and then use the Text tool to type characters on your image, the characters move towards the right side of your image. They're aligned with the original click on the left.

Use the Text|Align Left command before you type text or immediately after you finish typing. After you click to make another selection, you can't change the alignment of the text you've just typed.

**See Also**
Aligning text

## Text | Align Center

The Text|Center command centers text. The text is centered in relationship to where you initially clicked before typing the text.

For example, if you choose Text|Center and then use the Text tool to type characters on your image, the characters move towards the center of your image. They are centered in the image in relationship to your original click position.

Use the Text|Center command before you type text or immediately after you finish typing. After you click to make another selection, you can't change the alignment of the text you've just typed.

**See Also**
Aligning text

## Text | Align Right

The Text|Align Right command aligns text to the right of where you initially clicked before typing the text.

For example, if you choose Text|Align Right and then use the Text tool to type characters on your image, the characters move towards the left side of your image. They're aligned with the original click on the right.

Use the Text|Align Right command before you type text or immediately after you finish typing. After you click to make another selection, you can't change the alignment of the text you've just typed.

**See Also**
Aligning text

## Text | Font

The Text|Font command brings up the Select Font dialog box, where you choose the typeface, size, and style of the text.

Use this command either before you type text or immediately after you finish typing. After you click to make another selection, you can't change the font of the text you've just typed.

## Options menu (Paint editor)

The Options menu is displayed when you use the Paint editor. (This is different from the Options menu that's displayed when you edit a dialog box.)

Use this menu to choose options for the Paint editor.

Here are the Options menu commands:
Align
Size
Pattern
Brush Shape
Airbrush Shape
Pen Style
Editor Options

## Options | Align

The Options|Align command brings up the Align Selection dialog box, where you align your image.

This command is only available once you've selected an area of the image with the Pick Rectangle tool or the Scissors tool.

**See Also**
Pick Rectangle tool overview
Scissors tool overview

## Align Selection dialog box

The Align Selection dialog box is where you align the area of the image you've selected. Display this dialog box with Options|Align.

**Horizontal radio buttons**
The Horizontal radio buttons align the image horizontally.

**Vertical radio buttons**
The Vertical radio buttons align the image vertically.

Before you use this dialog box, you should zoom the image. Then, display a grid on the zoomed image. Using a grid makes it easy to align the selected area. The grid represents screen pixel coordinates.

**See Also**
Aligning a selected area
Zooming images

## Vertical radio buttons

The Vertical radio buttons align the selected area vertically within the image. After you've changed the area's alignment, it will be in the same column of the grid as its current location; only its row alignment will change.

**No Change**
Turn on the No Change radio button if you don't want to change the selected area's vertical alignment. This is the default.

**Top**
Turn on the Top radio button if you want to move the selected area to the top of the image.

**Center**
Turn on the Center radio button if you want to move the selected area to the center of the image.

**Bottom**
Turn on the Bottom radio button if you want to move the selected area to the bottom of the image.

## Horizontal radio buttons

The Horizontal radio buttons align the selected area horizontally within the image. After you've changed the area's horizontal alignment, it will be in the same row of the grid as its current location; only its column alignment will change.

**No Change**
Turn on the No Change radio button if you don't want to change the selected area's horizontal alignment. This is the default.

**Left Side**
Turn on the Left Side radio button if you want to move the selected area to the left side of the image.

**Center**
Turn on the Center radio button if you want to move the selected area to the center of the image.

**Right Side**
Turn on the Right Side radio button if you want to move the selected area to the right side of the image.

## Options | Size

The Options|Size command brings up the Stretch Selection dialog box where you size the image.

This command is only available once you've selected an area of the image with the Pick Rectangle tool or the Scissors tool.

**See Also**
Pick Rectangle tool overview
Scissors tool overview

## Stretch Selection dialog box

The Stretch Selection dialog box is where you stretch the area of the image you've selected. Display this dialog box with Options|Size.

**Old Position/Size display boxes**
The current position (Top, Left) and size (Width, Height) of the selected area.

**New Position/Size input boxes**
The New Position/Size input boxes are where you enter the new position and size of the selected area.

Before you use this dialog box, you should zoom the image. Then, display a grid on the zoomed image. Using a grid makes it easy to resize the selected area.

**See Also**
Resizing a selected area
Zooming images

## New Position/Size input boxes

The New Position/Size input boxes are where you enter the new position and size of the selected area. The values represent pixel coordinates of the grid displayed on the zoomed image.

First position the image by entering pixel coordinates in the Left and Top input boxes. Then, size the image with the Width and Height input boxes.

For example, a value of 10 in the Left input box places the selected area in column 10 (the 10th column from the left side of the image). A value of 5 in the Top input box positions the selected area in Row 5 (the 5th row down from the top of the image).

## Left input box

In the Left input box, enter the number of the column in which you want to position the left side of the selected area. This value does not stretch the area, but simply places it in the new column.

## Top input box

In the Top input box, enter the number of the row in which you want to position the top of the selected area. This value does not stretch the area, but simply places it in the new row.

## Width input box

Enter the new value for the width of the selected area in the Width input box. This is the number of the column in which you want to place the right side of the area. By entering a value that's greater than the selected area's width, you can stretch the area.

## Height input box

Enter the new value for the height of the selected area in the Height input box. This is the number of the row where you want to place the bottom of the area. By entering a value that's greater than the selected area's height, you can stretch the area.

## Options | Pattern

The Options|Pattern command brings up the Set Pattern dialog box, where you choose a paint pattern for Paint editor tools.

## Set Pattern dialog box

The Set Pattern dialog box is where you choose a pattern for all tools that paint a pattern on your image. Display this dialog box with Options|Pattern or by clicking the Pattern selection at the bottom of the Tools palette.

The tools that paint a pattern are:
Airbrush
Filled Ellipse
Filled Rectangle
Filled Rounded Rectangle
Paintbrush

The current pattern is displayed at the top of the dialog box. To choose a new pattern, just click on the pattern and click OK.

The next time you use a tool capable of filling with a pattern, Resource Workshop uses the pattern you specify. This pattern stays the same until you specify a new one.

**See Also**
Choosing a paint pattern

## Options | Brush Shape

The Options|Brush Shape command brings up the Brush Shape dialog box, where you choose a brush shape for the Paintbrush tool.

## Brush Shape dialog box

The Brush Shape dialog box is where you choose a brush shape for the Paintbrush tool. Display this dialog box with Options|Brush Shape or by clicking the Paintbrush shape at the bottom of the Tools palette.

The current shape is displayed at the top of the dialog box. To choose a new pattern, just click the shape you want and click OK.

The next time you use the Paintbrush tool, Resource Workshop uses the shape you specify. This shape stays the same until you specify a new one.

**See Also**
Choosing a brush shape

## Options | Airbrush Shape

The Options|Airbrush Shape command brings up the Airbrush Shape dialog box, where you choose a brush shape for the Airbrush tool.

## Airbrush Shape dialog box

The Airbrush Shape dialog box is where you choose a shape for the Airbrush tool. Display this dialog box with Options|Airbrush Shape or by clicking the Airbrush shape at the bottom of the Tools palette.

The current shape is displayed at the top of the dialog box. To choose a new shape, just click the shape you want and click OK.

The next time you use the Airbrush tool, Resource Workshop uses the shape you specify. This shape stays the same until you specify a new one.

**See Also**
Choosing a brush shape

## Options | Pen Style

The Options|Pen Style command brings up the Set Pen Style dialog box, where you choose the line style for Paint editor tools.

## Set Pen Style dialog box

The Set Pen Style dialog box is where you choose the line style for tools that draw a line. Display this dialog box with Options|Pen Style or by clicking the Pen style at the bottom of the Tools palette.

The tools that draw lines are:
Ellipse
Line
Pen
Rectangle
Rounded Rectangle

The current line style is displayed at the top of the dialog box. To choose a new style, just click the style you want and click OK.

Notice the null choice for a pen style. You can use null when you want to paint filled-in frames that don't have a border.

The next time you use a tool that draws a line, Resource Workshop uses the style you specify. This style stays the same until you specify a new one.

**See Also**
Choosing a line style

## Options | Editor Options

The Options|Editor Options command brings up the Set Paint Editor Options dialog box, where you customize the Paint editor.

## Set Paint Editor Options dialog box

The Set Paint Editor Options dialog box is where you customize the Paint editor. Display this dialog box with the Options|Editor Options menu command.

**Draw On Both Images check box**
The Draw On Both Images check box updates the image in both window panes.

**Grid On Zoomed Windows check box**
The Grid On Zoomed Windows check box displays a grid on a zoomed window.

**Save With Default Device Colors check box**
The Save With Default Device Colors check box saves the default Colors palette.

## Draw On Both Images check box

Turn on the Draw On Both Images check box if you want the Paint editor to update the image in both window panes as you draw.

If you turn this check box off, the Paint editor updates the other image only after you finish drawing an element.

Turn this option off if you find that performance is sluggish while you're drawing.

## Grid On Zoomed Windows check box

Turn on the Grid On Zoomed Windows check box if you want to display a grid on a zoomed image. The grid shows you how the image is painted on a pixel-by-pixel basis.

The grid makes it easy to use the Options|Align and Options|Size commands to align and size the selected area of the image.

**See Also**
Zooming images

## Save With Default Device Colors check box

Turn the Save With Default Device Colors check box off if you want to save a custom Colors palette you've created. When this check box is on, any colors you've customized in the color palette will revert to the default color when you close the Paint editor. This option applies only to a 256-color palette.

**See Also**
Customizing colors

## Images menu

The Images menu is displayed when you open an icon image. Use this menu to create a new image or edit an existing image.

Here are the Images menu commands:
New Image
Edit Image

## Images | New Image

The Images|New Icon Image command brings up the New Icon Image dialog box, where you specify the size and color form for a new icon image.

## New Icon Image dialog box

The New Icon Image dialog box is where you choose the size and color format for a new image. Display this dialog box with Images|New Image.

**Size radio buttons**
The Size radio buttons determine the image's pixel size.

**Colors radio buttons**
The Colors radio buttons determine the number of colors in the image.

## Size radio buttons

The Size radio buttons let you choose the image's size.

**32 x 32**
Turn the 32 x 32 radio button on if you want the icon to be in a 32 x 32 pixel format.

**32 x 16**
Turn the 32 x 16 radio button on if you want the icon to be in a 32 x 16 pixel format.

**64 x 64**
Turn on the 64 x 64 radio button if you want the icon to be in a 64 x 64 pixel format.

## Colors radio buttons

The Colors radio buttons let you choose the image's color format.

**2 Colors**
Turn on the 2 Colors radio button if you want the icon to be a black and white image.

**8 colors**
Turn on the 8 Colors radio button if you want the icon to be an 8-color image.

**16 colors**
Turn on the 16 Colors radio button if you want the icon to be a 16-color image.

**256 colors**
Turn on the 256 Colors radio button if you want the icon to be a 256-color image. This button is available only if your display hardware supports 256 colors. Your program must supply its own support for 256 colors.

## Display Device Information dialog box

The Display Device Information displays information about your display hardware, including:

- Number of bits per pixel
- Number of color planes
- Number of colors supported
- Whether it is a palette device

If the device supports logical color palettes, you also see:

- The number of entries in the system palette
- The number of reserved entries in the system palette
- The color resolution of the device in bits per pixel

Click OK when you've finished reviewing the information.

## Images | Edit Image

The Images|Edit Image command puts you in the Paint editor where you can edit the selected icon image.

## Icon menu

The Icon menu is displayed when you work with icons. Here are the Icon menu commands:
Hide Palette
Hide Toolbox
Size and Attributes
Test
Edit Foreground Color
Edit Background Color

## Icon | Hide Palette

The Icon|Hide Palette command hides the Paint editor Colors palette. This command is available when the Colors palette is displayed.

If you hide the Colors palette, the command switches to Show Palette.

## Hide Toolbox

The Hide Toolbox command hides the Paint editor Tools palette. This command is available when the Tools palette is displayed.

If you hide the Tools palette, the command switches to Show Toolbox.

## Icon | Size and Attributes

The Icon|Size and Attributes command brings up the Icon Image Attributes dialog box, where you change an icon's resolution and color format and view information about your display hardware.

## Icon Image Attributes dialog box

The Icon Image Attributes dialog box is where you change an icon's attributes. Display this dialog box with Icon|Size and Attributes.

**Size radio buttons**
The Size radio buttons determine the image's pixel size.

**Colors radio buttons**
The Colors radio buttons determine the number of colors in the image.

**Device Info Button**
The Device Info button brings up the Display Device Information dialog box that displays information about your display hardware.

**See Also**
Changing an icon's attributes

## Icon | Test

Use the Icon|Test command to move your icon around and test its transparent and inverted areas against various backgrounds.

**See Also**
Testing an icon

## Icon | Edit Foreground Color

If you select a color from the Colors palette for the foreground color, the Icon|Edit Foreground Color command brings up the Edit Color dialog box.

If you select Transparent or Inverted for the foreground color, this command brings up the Set Transparent Color dialog box.

**See Also**
Foreground color overview
Transparent and inverted areas

## Icon | Edit Background Color

If you select a color from the Colors palette for the background color, the Icon|Edit Background Color command brings up the Edit Color dialog box.

If you select Transparent or Inverted for the background color, this command brings up the Set Transparent Color dialog box.

**See Also**
Background color overview
Transparent and inverted areas

## Cursor menu

The Cursor menu is displayed when you work with cursors. Here are the Cursor menu commands:
Hide Palette
Hide Toolbox
Set Hot Spot
Test
Set Transparent Color

## Cursor | Hide Palette

The Cursor|Hide Palette command hides the Paint editor Colors palette. This command is available when the Colors palette is displayed.

If you hide the palette, the command switches to Show Palette.

## Cursor | Set Hot Spot

The Cursor|Set Hot Spot command brings up the Set Hot Spot dialog box, where you identify the cursor's active area.

## Set Hot Spot dialog box

The Set Hot Spot dialog box is where you enter the pixel coordinates of the cursor's hot spot (active area). Display this dialog box with Cursor|Set Hot Spot.

**Horizontal input box**
The Horizontal input box is where you enter the horizontal coordinate of the cursor's hot spot.

**Vertical input box**
The Vertical input box is where you enter the vertical coordinate of the cursor's hot spot.

**See Also**
Setting the cursor's hot spot

## Cursor | Test

Use the Cursor|Test command to move your cursor around and see how it looks on different color backgrounds.

Here's how you test the cursor's hot spot,

1. Use the View|Zoom In command to zoom in on the cursor image.
2. Display a grid on the zoomed image.
3. Select the Paint Can tool.
4. Choose Cursor|Test.
5. Move the hot spot to a particular pixel on the zoomed image and click the mouse. The test cursor disappears and is replaced by the Paint Can tool. If you correctly set the hot spot, the paint tool points to the same pixel your test cursor pointed to.

**See Also**
Testing a cursor
Zooming images

## Cursor | Set Transparent Color

The Cursor|Set Transparent Color command brings up the Set Transparent Color dialog box, where you can change the color of the cursor's transparent and inverted areas.

## Bitmap menu

The Bitmap menu is displayed when you work with bitmaps.

Here are the Bitmap menu commands:
Hide Palette
Hide Toolbox
Size and Attributes
Edit Foreground Color
Edit Background Color

## Bitmap | Hide Palette

The Bitmap|Hide Palette command hides the Paint editor Colors palette. This command is available when the Colors palette is displayed.

If you hide the Colors palette, the command switches to Show Palette.

## Bitmap | Size and Attributes

The Bitmap|Size and Attributes command brings up the Set Bitmap Attributes dialog box, where you change a bitmap's size and attributes.

## Set Bitmap Attributes dialog box

The Set Bitmap Attributes dialog box is where you change a bitmap's size and attributes. Display this dialog box with the Bitmap|Size and Attributes command.

**Size input boxes**
The Size input boxes are where you choose the size of your bitmap.

**Resize Current Bitmap check box**
The Resize Current Bitmap check box resizes your bitmap. Turn this check box on if you want an image that's already drawn to stretch or shrink when you resize your bitmap.

**Compression radio buttons**
The Compression radio buttons compress your bitmap.

**Colors radio buttons**
The Colors radio buttons determine the number of colors in the image.

**Format radio buttons**
The Format radio buttons choose how to store your bitmap.

**Device Info Button**
The Device Info button brings up the Display Device Information dialog box that displays information about your display hardware.

**See Also**
Changing a bitmap's attributes

## Size input boxes

The Size input boxes let you choose the bitmap's size. The maximum size you can work with is limited by the amount of available memory on your computer. Even though you could type 9999 for the width or height of your image, the maximum size for bitmaps on your computer is probably far less.

**Width in Pixels input box**
The Width in Pixels input box is where you enter the width of the total bitmap image. Enter the value in pixels.

**Height in Pixels input box**
The Height in Pixels input box is where you enter the height of the total bitmap image. Enter the value in pixels.

## Compression radio buttons

The Compression radio buttons let you compress your bitmap images.

**None**
Turn on the None radio button if you're creating a 2-color bitmap.

**RLE4**
Turn on the RLE4 radio button if you're creating a 16-color bitmap.

**RLE8**
Turn on the RLE8 radio button if you're creating a 256-color bitmap.

RLE stands for "run length encoded."

You may want to experiment with both compressed and non-compressed color images because sometimes the compressed image takes up more space than the uncompressed image.

## Colors radio buttons

The Colors radio buttons let you choose the image's color format.

**2 Colors**
Turn on the 2 Colors radio button if you want the bitmap to be a black and white image.

**16 colors**
Turn on the 16 Colors radio button if you want the bitmap to be a 16-color image.

**256 colors**
Turn on the 256 Colors radio button if you want the bitmap to be a 256-color image. This button is available only if your display hardware supports 256 colors. Your program must supply its own support for 256 colors.

## Format radio buttons

The Format radio buttons let you choose how to store your bitmap.

**Windows**
Turn on the Windows radio button to store your bitmap in Windows format. The image can be compressed or in a format compatible with Windows.

**OS/2**
Turn on the OS/2 radio button to store your bitmap in OS/2 format. The image can be compressed or in a format compatible with OS/2.

## Bitmap | Edit Foreground Color

The Bitmap|Edit Foreground Color command brings up the Edit Color dialog box, where you edit the bitmap's foreground color.

**See Also**
Foreground color overview

## Bitmap | Edit Background Color

The Bitmap|Edit Background Color command brings up the Edit Color dialog box, where you edit the bitmap's background color.

**See Also**
Background color overview

## Font menu

The Font menu is displayed when you work with fonts. Here are the Font menu commands:
Hide Palette
Hide Toolbox
Header
Font Size
Character Width

## Font | Hide Palette

The Font|Hide Palette command hides the Paint editor Colors palette. This command is available when the Colors palette is displayed.

If you hide the Colors palette, the command switches to Show Palette.

## Font | Header

The Font|Header command brings up the Font Header Information dialog box, where you create header information for your font resource, including:

- name
- copyright
- character set
- family
- size and weight

## Font Header Information dialog box

The Font Header Information dialog box is where you create header information for your font resource. Display this dialog box with the Font|Header command.

**Attributes boxes**
The Attributes boxes choose your font's attributes.

**Copyright input box**
The Copyright input box is where you enter copyright information.

**Device input box**
The Device input box is where you enter a device name for your font. The device name informs your programs that this font can only be used on a particular device.

**Face Name input box**
The Face Name input box is where you assign a name to your font. The face name is displayed on the status line at the top of the image.

**Font Version radio buttons**
The Font Version radio buttons select the font's version. For reliability, always choose 2.00.

**Sizes input boxes**
The Sizes input boxes define the size of characters in your font resource.

**Type display box**
The Type display box displays the type of the font. Resource Workshop creates and edits only raster fonts.

**See Also**
Defining a header for a font resource

## Attributes boxes

The Attributes boxes tell Windows and your applications the attributes the font uses.

**Character Set input box**
The Character Set input box is where you select the font's character set.

**Family input box**
The Family input box describes the font family.

**Italic check box**
The Italic check box implements italicized characters.

**Strikeout check box**
The Strikeout check box implements strikeout characters.

**Underline check box**
The Underline check box implements underlined characters.

**Variable Pitch check box**
The Variable Pitch check box creates characters of varying pitch.

**Weight input box**
The Weight input box is where you choose the weight of characters. Enter 400 if the font is of normal weight. Enter 700 to indicate a boldfaced font.

## Family input box

Enter the font's family in the Family input box. Enter one of the following values:

| Value | Family Name | Description |
|---|---|---|
| 1 | Roman | Proportionally-spaced fonts with serifs. |
| 2 | Swiss | Proportionally-spaced fonts without serifs. |
| 3 | Modern | Fixed-pitch fonts. |
| 4 | Script | Cursive or script fonts. |
| 5 | Decorative | Novelty fonts. |

Enter 0 (zero) if you're creating an image, rather than a character set.

## Character Set input box

Enter the font's character set in the Character Set input box. Examples of character sets are Courier, Elite, and Pica.

The value can be from 0-255. However, only 0, 2, and 255 have predefined meanings:

| Value | Description |
|---|---|
| 0 | ANSI, the default Windows character set |
| 2 | Symbol, used for math and scientific formulas |
| 255 | OEM, a machine specific character set |

## Sizes input boxes

The Sizes input boxes let you choose the size of characters in your font resource.

**Horizontal Resolution input box**
The Horizontal Resolution input box is where you choose your font's horizontal resolution.
Enter the resolution as pixels per logical inch on your video display.

**Vertical Resolution input box**
The Vertical Resolution input box is where you choose your font's vertical resolution. Enter the resolution as pixels per logical inch on your video display.

**Points input box**
The Points input box is where you choose your font's type size. The size is based on a point, which is 1/72 of an inch. Characters are measured from the top of the ascender to the bottom of the descender. Do not include the space reserved for diacritics. Enter that value in the Internal Leading input box.

**Internal Leading input box**
The Internal Leading input box is where you choose the space reserved for diacritics. Enter the value in pixels.

**External Leading input box**
The External Leading input box is where you choose the number of spaces between lines.
(the vertical distance from the bottom of one character cell to the top of the character cell below it).
Enter the value in pixels.

**Ascent input box**
The Ascent input box is where you choose the height of the character above the baseline. Enter the value in pixels.

## Font | Font Size

The Font|Font Size command brings up the Font Size Information dialog box, where you choose the size for your font images and the number of images to include in your font resource.

## Font Size Information dialog box

The Font Size Information dialog box is where you choose the size for your font images and the number of images to include in your font resource. The character you're editing is displayed on the status line at the top of the image.

Display this dialog box with the Font|Font Size command.

**Sizes input boxes**
The Sizes input boxes are where you choose the font's size.

**Stretch Current Characters check box**
The Stretch Current Characters check box resizes the font, so that existing images stretch or shrink, based on height and width values you change in the Font Size Information dialog box.

**Character input boxes**
The Character input boxes are where you choose how many images to include in your font.

**See Also**
Defining and adding characters for a font

## Size input boxes

The Size input boxes let you choose the font's width, height, average width, and maximum width.

**Width input box**
If all images in your font resource are the same width (fixed-width), enter the width in pixels. If the widths vary (variable-width), enter a 0 (zero) here and enter a maximum width in the Maximum Width input box.

**Height input box**
The Height input box is where you enter the font's height. Enter the value in pixels.

**Average Width display box**
If you enter 0 (zero) in the Width input box to indicate a variable-width font, Resource Workshop calculates an average width for your font images and displays it in the Average Width display box.

If you don't use a variable-width font, the value in the Average Width display box is the same as the value in the Width input box. The Average Width value is calculated when you open the Font Size Information dialog box. You won't see this value change as you type other changes into the dialog box.

**Maximum Width input box**
If you enter 0 (zero) in the Width input box to indicate a variable-width font, enter the maximum width for font images, in pixels. This option is available only if you enter 0 in the Width input box.

**See Also**
Creating variable-width fonts

## Character input boxes

The Character input boxes options let you choose how many images to include in your font resource.

You use these input boxes to map your font images to the ANSI character set. For example, to map a font image to the character a, specify the decimal code 97. (The image itself does not need to be the character a, unless you want it to be.) The second font image would then be mapped to the character b, decimal code 98.

The ANSI values to which you map a set of images are arbitrary, but must be in the range 0 to 255. The ANSI values become important when you load the font in your program and display the images, because you use the ANSI value that corresponds to an image to display it.

**First input box**
The First input box is where you enter an ANSI decimal code that defines the first image in your font. For example, if you want the first image to correspond to a, type 97.

**Last input box**
The Last input box is where you enter an ANSI decimal code that defines the last image in your font. For example, if you want the last image to correspond to z, type 122.

**Default input box**
The Default input box is where you enter an ANSI decimal code for the default image that will be displayed when you edit this font resource. The default value must be within the character range defined by the values entered in the First and Last input boxes.

**Break input box**
The Break input box is where you enter an ANSI decimal code for a break character for your font resource. (A break character, typically the space character, is used to pad justified lines.) The Break value must be within the character range defined by the values entered in the First and Last input boxes.

## Font | Character Width

The Font|Character Width command brings up the Character Width dialog box, where you define a variable-width font.

This command is available only if you've entered 0 (zero) in the Width input box in the Font Size Information dialog box.

## Character Width dialog box

The Character Width dialog box is where you define variable-width fonts. Display this dialog box with the Font|Character Width command.

**Maximum Width display box**
The Maximum Width display box shows the value you entered in the Maximum Width input box in the Font Size Information dialog box.

**Width input box**
The Width input box is where you choose a width for your font resource. Enter a value, in pixels, that's less than or equal to the value in the Maximum Width display box.

**Stretch Current Characters check box**
The Stretch Current Characters check box resizes the font so that the existing images stretch or shrink, based on new width value.

**See Also**
Creating variable-width fonts