Defining Topic Windows

After the Help file is built, topics appear in the Help window. In addition to the main Help window, you can also display Help topics in other kinds of windows: secondary windows, pop-up windows, and embedded windows. By using different windows for the Help information, you can create topics that communicate information effectively.

This chapter discusses how to define Help windows for your topic.

About Help Windows

There are four kinds of Help windows.

Window type	Description
Main window	Main Help application window, which contains the menu bar and button bar and that most Help files use to display topics. The main Help window can include a nonscrolling region.
Secondary window	Independent windows similar to the main Help window, except secondary windows lack a menu bar and button bar. Secondary windows can include a nonscrolling region.
Pop-up window	Temporary windows that appear in a fixed size and location over the main Help window or a secondary window.
Embedded window	Rectangular windows that appear in a fixed size and location within another Help window and use a dynamic-link library (DLL) to display information, such as a bitmap.

Help files can display topics in the main window, one secondary window, and

one pop-up window at the same time. Although you can display only one secondary window and one pop-up window at a time, you can use them throughout your Help file. Figure 9.1 shows a Help file that uses a main window, a secondary window, and a pop-up window.

Graphic

Inside topics, Help also displays *embedded windows*, which are rectangular regions that display graphics, multimedia data, and other specialized information. Embedded windows are formatted with the topic text, and they move when the topic is scrolled or when the window is sized.

Figure 9.2 shows a topic in the main Help window that includes an animation displayed in an embedded window.

Graphic

To display topics in windows, you use *jumps*. When you create a jump hot spot, you choose a topic to display and a window in which to display the topic. Jumps

The Main Help Window

are discussed in Chapter 8, “Creating Links and Hot Spots.”

Although it is possible to create a Help file that does not use the main Help window, most Help files do have a main window. Help defines the basic characteristics of the default main Help window, but you can change these attributes if you want. You can define a total of eight attributes for the main window. Main window attributes include:

- The title that appears in the main window’s title bar.
- The position of the window’s upper left corner.
- The window’s size.
- The window’s state (maximized or normal).
- The background color of the scrolling and nonscrolling regions.

The main window attributes are determined by the entry in the [WINDOWS] section of the Help project file and are used when the Help file is displayed. After

the file is built, the main window attributes cannot be changed while the Help file is open. The window attributes do not apply to any other Help file that might be opened during a user's session.

A main window definition has the following form:

main="caption", **(***x-coord*, *y-coord*, *width*, *height***)**, *window-state*, **(***scrolling-RGB***), (***nonscrolling-RGB***)**

The following example shows a main window definition:

```
[WINDOWS]
main="Sample Help", (50, 50, 800, 900), 0, , (192,192,192)
```

This definition defines a main window that displays Sample Help in the title bar. The window is positioned in the upper-left corner of the screen and is 800-by-900 (in Help coordinates). The window appears in its normal state with a light grey background in the nonscrolling region.

The following sections describe each attribute in a main window definition.

Note

The information here is a summary of the window attributes. For a complete description of each attribute, refer to the [WINDOWS] section in Chapter 16, "The Help Project File."

Window Name

The *window-name* must be "main" for the main Help window and cannot be changed.

Caption

The *caption* is the text that appears in the main window's title bar and in the icon label when the window is minimized. In the sample definition above, the window caption is Sample Help.

X-Coord and Y-Coord

The *x-coord* and *y-coord* attributes are numbers that define where the window appears on the user's computer screen. The numbers represent the horizontal and vertical location of the upper-left corner of the window as defined in terms of Help's 1024-by-1024 coordinate system. For example, the coordinates for the

sample main window are (50, 50), or the upper-left corner of the screen.

When Help first displays the window, it uses the position values you've specified. Users can move the window after it has been displayed.

Width and Height

The *width* and *height* attributes are numbers that define the normal width and height of the window, again in terms of the Windows Help coordinate system. In the example definition, the window is 800-by-900.

When Help first displays the window, it uses the size you've specified. Users can change the window size after it has been displayed.

Window State

The *window-state* is a number (0 or 1) that indicates whether the window is displayed at normal size or maximized when Help first opens it. In the example, the main window is displayed at normal size.

The following table shows the effect of the window-state attribute.

State	Initial window position
Normal	Window appears in the location and size defined by your position and size values.
Maximized	Window fills the entire screen. If the user restores the window to its normal state, it occupies the part of the screen defined by your position and size values.

Scrolling and Nonscrolling Region Background Colors

The *scrolling-RGB* and *nonscrolling-RGB* attributes are number sequences that define the background color of the window's scrolling and nonscrolling regions. The numbers represent the red, green, and blue (RGB) components of the color. If these numbers are not given, the scrolling and nonscrolling areas use the user's default colors as defined by the Control Panel Colors setting. Help supports any

solid or dithered colors available from the Windows system colors. In the example, the main window uses light gray [(192,192,192)] as the background color for the nonscrolling region.

Use the Color setting in Control Panel to see the colors resulting from different RGB values. To change the color of text in the window, use your word processor. Format the text in the color that you want it to appear. Just remember not to create conflicts between the text color and any custom background color you might have specified.

The Windows Help Coordinate System

When defining windows, you use the Help coordinate system to specify window position and size values. Help uses its own device-independent coordinate system because the screen resolution, or number of pixels, on Windows displays can vary. This device-independent system divides the screen into 1024 vertical units and 1024 horizontal units. When Help displays a Help file, it converts the size and position values specified by the Help file into values appropriate to the resolution of the display device. This allows Help to display Help windows using a consistent size and location despite variations in screen resolution.

The Help coordinate system is mapped to the horizontal and vertical resolutions of the video card, so you'll have to do some calculations to determine the exact pixel location. For example, if the resolution of your video card is *horiz* by *vert* pixels, and the horizontal and vertical locations you specify are at Windows Help coordinates *x* and *y*, then the x-coordinate of the upper-left corner is at the pixel given by the following formula:

x * (*horiz*/1024)

The y-coordinate of the upper-left corner is at the pixel given by the following formula:

y * (*vert*/1024)

The window's width and height are also specified in Help's coordinate system. For example, a window may specify a width of 511 and a height of 511. A standard VGA card has a resolution of 640-by-480 pixels, so the width is actually the following number of pixels:

511 * (640/1024) = 319

The height is actually the following number:

511 * (480/1024) = 240

In other words, the window would take up approximately one quarter of the area of the Windows Help screen (half the width multiplied by half the height).

To convert from pixels to Windows Help coordinates, you invert the ratio between Windows Help's resolution and the video resolution. Again, assuming the resolution of your video card is *horiz* by *vert* pixels, and the horizontal and vertical locations (or dimensions) you want are *x* and *y* pixels, the x-coordinate (or width), in Windows Help coordinates, is as follows:

x * (1024/*horiz*)

The y-coordinate (or height), in Windows Help coordinates, is:

y * (1024/*vert*)

Because the Windows Help coordinate system ranges from 0 through 1023 in both directions, the horizontal position plus the width must be less than or equal to 1023. Similarly, the vertical position plus the height must be less than or equal

Secondary Windows

to 1023.

You can create *secondary windows* for your Help file. A secondary window is an independent Help window that complements and supports the main Help window and displays similar information. Secondary windows have many uses but are very effective when you want to display new information without having users lose information that is already displayed in the main Help window. In other words, secondary windows let you have two Help windows open at the same time. Here are just a few possibilities.

You can use secondary windows to:

- Display the Help Contents or alphabetic index.

Placing the contents or index topics in a secondary window that has jumps to the main Help window lets users browse through the information without losing the list of choices.

GRAPHIC

- Create a separate content window for a specifc type of Help information.

For example, you can create a secondary window that displays step-by-step procedures in it. That way users can close the main window and just use the secondary window with their application.

GRAPHIC

- Create a window that displays pictures, examples, and other information related to the main topic.

Because secondary windows function independently of the main Help window, you can use them to show users extra material at the same time that they are looking at the main topic. For example, in Windows version 3.1, the Help files include a master glossary that is displayed in a secondary window.

- Gain complete control over the Help interface.

You cannot remove the standard menus and buttons from the main Help window. However, if you want to create a Help system with custom controls (menus only or buttons only, for example), you can display all the Help topics in a secondary window. To provide functionality to users, you can add a nonscrolling region and create custom controls using Help macros that simulate menus and buttons.

Whether you use secondary windows and how you use them depends entirely on how you have designed the information model for your Help system.

How Secondary Windows Work

A secondary window has the same characteristics as a standard window. It includes a Control menu, a title bar, minimize and maximize buttons, and window borders. Users can move, resize, minimize, or maximize secondary windows. If necessary, secondary windows can display horizontal and vertical scroll bars that users can use to view information displayed in the secondary window. Secondary windows do not include a menu bar or button bar or any other Help controls. Any custom controls or functionality in secondary windows must be defined by individual Help authors for each Help file.

A secondary window is completely independent of the main Help window It can be open even if the main Help window is closed. When open, a secondary

window's title appears in the Windows Task List. If a user minimizes a secondary window, it displays its own icon. Help uses the standard Windows Help question-mark icon unless the Help author specifies a custom icon using the **ICON** option in the [OPTIONS] section of the Help project file. (See Chapter 16, "The Help Project File," for details.)

Any topic that you can display in the main Help window you can also display in a secondary window. Secondary windows can include any standard Windows Help feature, including text, graphics, jump hot spots, pop-up windows, macro hot spots, nonscrolling areas, or embedded windows.

Secondary windows appear on demand, usually when the user chooses a menu item, Help button, or hot spot. A macro in the Help project file or a **WinHelp** API call can also be used to display a secondary window. For information about using Help macros to display a secondary window, see Chapter 14, "Help Macros." For information about how an application can send an API call to Help in order to display a secondary window, see Chapter 19, "The WinHelp API."

Secondary window attributes are specified when a Help file is opened and remain in effect until the Help file is closed.

Limitations

Despite their usefulness, secondary windows have the following limitations:

- Windows Help can display only one secondary window at a time. If the user completes an action that requires Help to display a secondary window while one is already open, the new secondary window replaces the current one.
- A single Help file can define only up to five secondary window types, which limits the number of different ways you can use secondary windows. For example, you cannot create a design that customizes the secondary window for every topic displayed in it; instead, you should design secondary windows for categories of information.
- You cannot use all of the Help macros in secondary windows. Some macros are ignored when executed from a secondary window, and others are not recommended because they may create conflicts if executed from a secondary window. To find out whether a macro can be executed from a secondary window, check the macro's Comments

section in Chapter 15, “Help Macro Reference.”

- Secondary windows remain open if the user closes the main Windows Help window using the Control menu. The user must then close the secondary window using the secondary window’s Control menu or a custom control that you provide.
- Secondary window topics don’t appear in the history list nor are they recorded in the back list. So, if you place important information in a secondary window, the user will not be able to use the History or Back button to return to a previously viewed topic.
- The keyword search feature in Windows Help displays all topics in the main Help window, even if that topic is normally displayed in a secondary window. You can assign keywords to secondary window topics, but when accessed from the Search dialog box, they will not appear in the secondary window.
- You cannot use Help macros to change the on-top state of a secondary window without affecting the main Help window.

Creating a Secondary Window

Help creates a secondary window only if one is defined in the [WINDOWS] section of the Help project file for that Help file. Adding a secondary window to your Help file requires the following steps:

- Creating the topic that will appear in the secondary window
- Defining the attributes of the secondary window in the [WINDOWS] section of the Help project file
- Creating any jumps in the main topic file to the topic displayed in the secondary window

Defining Secondary Window Attributes

You can define a total of ten attributes for each secondary window that you create. The attributes are the same as those for the main window, except that a secondary window includes a unique name and may be authored to stay on top of

other windows.

Secondary window attributes include:

- The name of the secondary window.
- The title that appears in the secondary window's title bar.
- The position of the window's upper left corner.
- The window's size.
- The window's state (maximized or normal).
- The background color of the scrolling and nonscrolling regions.
- The window's on-top state (on top or normal).

You can define as many as five secondary windows for a single Help file. The window attributes are determined by the entry in the [WINDOWS] section of the Help project file and are set when the Help file is opened. Once, the window attributes cannot be changed while the Help file is open. The window attributes do not apply to any other Help file that might be opened during a user's session.

A secondary window definition has the following form:

window-name="*caption*", (*x-coord*, *y-coord*, *width*, *height*), *window-state*, (*scrolling-RGB*), (*nonscrolling-RGB*), *ontop-state*

The following example shows a sample secondary window definition:

```
[WINDOWS]
graph="Charts", (0,0,600,723), 0, , (192,192,192), 0
```

This definition creates a secondary window named "graph" that displays Charts in the title bar. The window is positioned in the extreme upper-left corner of the screen and is 600-by-723 (in Help coordinates). The window appears in its normal state with a light grey background in the nonscrolling region. The window does not stay on top of other application windows.

Note

The following sections describe each attribute in a secondary window definition. If you have already read about Help window attributes in “The Main Window” section, you may want to skim this section and just read about the *window-name* and *ontop-state* attributes.

The information here is a summary of the secondary window attributes. For a complete description of each attribute, refer to the [WINDOWS] section in Chapter 16, “The Help Project File.”

Window Name

The *window-name* identifies a secondary window and gives it a name that you can then use when creating jumps and Help macros. In the sample definition above, the window name is “graph.”

Whenever you include this name in a jump hot spot, Help displays the topic in a secondary window. (For an example of a jump to a secondary window, see “Creating Jumps to Secondary Windows,” later in this chapter.)

Caption

The *caption* is the text that appears in the secondary window’s title bar and in the icon label when the window is minimized. In the sample definition above, the window caption is Charts.

X-Coord and Y-Coord

The *x-coord* and *y-coord* attributes are numbers that define where the secondary window appears on the user’s computer screen. The numbers represent the horizontal and vertical location of the upper-left corner of the window as defined in terms of Help’s 1024-by-1024 coordinate system. For example, the upper-left corner coordinates for the sample secondary window “graph” are (0,0), the extreme upper-left corner of the screen.

When Help first displays the window, it uses the position values you’ve specified. Users can move the window after it has been displayed.

Width and Height

The *width* and *height* attributes are numbers that define the normal width and height of the window, again in terms of the Windows Help coordinate system. In the example definition, the window is 600-by-723.

When Help first displays the window, it uses the size values you've specified. Users can change the window's size after it has been displayed.

Window State

The *window-state* is a number (0 or 1) that indicates whether the window is displayed at normal size or maximized when Help first opens it. In the example, the "graph" window is displayed at normal size.

The following table shows the effect of the window-state attribute.

State	Initial window position
Normal	Window appears in the location and size defined by your position and size values.
Maximized	Window fills the entire screen. If the user restores the window to its normal state, it occupies the part of the screen defined by your position and size values.

Scrolling and Nonscrolling Region Background Colors

The *scrolling-RGB* and *nonscrolling-RGB* are number sequences that define the background color of the window's scrolling and nonscrolling regions. The numbers represent the red, green, and blue (RGB) components of the color. If these numbers are not given, the scrolling and nonscrolling areas use the user's default colors as defined by the Control Panel Colors setting. Help supports any solid or dithered colors available from the Windows system colors. In the example, the secondary window uses light gray [(192,192,192)] as the background color for the nonscrolling region.

Use the Color setting in Control Panel to see the colors resulting from different RGB values. To change the color of text in the window, use your word processor.

Format the text in the color that you want it to appear. Just remember not to create conflicts between the text color and any custom background color you might have specified.

On Top State

The *ontop state* specifies whether the secondary window stays on top of other windows. If the author defines the window as “on top,” the user cannot change the window behavior using the Always On Top command in Help. The on-top value is either 0 for normal behavior or 1 for on-top behavior. The main Help window cannot be authored as a topmost window. The example window “graph” is not defined as an on-top window.

Creating Jumps to Secondary Windows

After you have defined a secondary window for your Help file, you will probably want to create hot spots with the topics that access the secondary window. In most cases, you can create a hot spot that causes a jump:

- From the main Help window to a secondary window.
- From the secondary window back to the main window.
- To a secondary window in a different Help file.

Creating Standard Jumps to Secondary Windows

You create jumps to topics in secondary windows the same way you create standard jumps. The difference is that you must include the name of the secondary window in the hot spot information, as shown here:

context-string>window-name

Context-string is the context string for the topic you are jumping to, and *window-name* is the name of the secondary window where you want the topic to appear.

To create a jump to a secondary window

1. Follow the steps to create a standard jump hot spot.
2. Insert a right angle bracket (>) immediately after the last character of the context string.
3. Type the name of the secondary window where you want the topic to

appear.

Note The angle bracket and window name must be formatted as hidden text.

Figure 9.x shows a correctly formatted hot spot for a secondary window jump.

Graphic

Note

As with other hot spots, you can change the basic appearance of a hot spot that references a secondary window. Because the secondary window information is completely contained within the hidden text, the procedures are exactly the same as those for regular jump hot spots. Refer to the “Changing the Standard Appearance of Jump Hot Spots”section in Chapter 8, “Creating Links and Hot Spots,” for details.

Creating Jumps to the Main Help Window

Generally, if you create one or more secondary windows, you will also need to create hot spots in secondary window topics that jump back to the main window. To do that, you use the same jump format and specify “main” as the *window-name*, as in this example:

context-string>main

The window name “main” is reserved for the main Help window and must be used when specifying jumps to the main window.

To create a jump from a secondary window to the main Help window

1. Follow the steps to create a standard jump hot spot.
2. Insert a right angle bracket (>) immediately after the last character of the context string.
3. Type **main** as the window-name.

Note The angle bracket and **main** must be formatted as hidden text.

Figure 9.x shows a correctly formatted jump hot spot for the main window.

Graphic

To help you coordinate the links in your Help file so that the correct topics are displayed in the appropriate window, the following table presents a complete summary of the possible links you can create between the main Help window and a secondary window.

Jump location	Type of jump	Result of the jump
Main window	standard	The topic is displayed in the main Help window.
Main window	>main	The topic is displayed in the main Help window.
Main window	>*window-name*	The topic is displayed in the specified secondary window. If Help opens a new secondary window to display the topic, the previous size and location of the secondary window may change. The main Help window is unaffected by this type of jump.
Secondary window	standard	The topic is displayed in the specified secondary window. The main Help window is unaffected by this type of jump.
Secondary window	>main	The topic is displayed in the main Help window.
Secondary window	>*window-name*	The topic is displayed in the specified secondary window. If Help opens a new secondary window to display the topic, the previous size and location of the secondary window may change. The main Help window is unaffected by this type of

jump.

Creating Interfile Jumps to Secondary Windows

If the destination topic is in another Help file, you can also include the name of the Help file before the *window-name*, as shown here:

*context-string***[@***filename***]**>*window-name*

The *filename* is the name of the Help file that contains the topic you want to display in the secondary window.

To create an interfile jump to a secondary window

1. Follow the steps to create a standard jump hot spot.
2. Insert an at sign (@) immediately after the last character of the context string.
3. Type the name of the file that contains the context string assigned to the topic that is the target of the interfile jump.

Note The at sign and Help filename must be formatted as hidden text.

4. Insert a right angle bracket (>) immediately after the last character of the Help filename.
5. Type the name of the secondary window where you want the topic to appear.

Note The angle bracket and window name must be formatted as hidden text.

Figure 9.x shows a correctly formatted interfile jump hot spot in a topic file.

Graphic

Closing Secondary Windows

Because secondary windows are independent windows, they do not close by themselves. The user has the greatest responsibility for closing a secondary window, but applications can also close a secondary window. When dealing with Help requests from different applications, Help automatically closes secondary windows to avoid situations in which the content of the main Help window and

the secondary window are totally unrelated.

The following list describes all the possible ways to close a secondary window:

- The user can choose the Close command from the Control menu (or double-click the Control-menu box).
- The user can choose a custom control provided by the Help author that executes the **CloseWindow** macro.
- The user can open a new Help file.
- The user can exit Help.
- The application can send a HELP_QUIT message to Help and close Windows Help.
- The application can send a **CloseWindow** macro to Help and close a specific secondary window.
- Another application, different from the one currently displaying the secondary window, can request Help. This request will cause Help to close any open secondary windows and display the requested Help file in the main window.

Note

If a different application from the one displaying Help topics requests Help and requests Help to display a topic in a secondary window, Help will close the main window before opening the new Help

Defining Nonscrolling Regions

file and displaying the topic in the secondary window.

The main Help window and secondary windows display a scroll bar when the topic is too long or too wide to fit in the window's current size. By using the scroll bar, users can scroll the topic to see the out-of-view information. In addition to defining the Help window itself, you can also set aside part of the window as a *nonscrolling region*, which displays text or graphics in a fixed

location immediately below the button bar. Text or graphics in the nonscrolling region remain in place even when users scroll the topic. Because they don't scroll with the rest of the topic, nonscrolling regions can have a number of different uses. Here are just a few possibilities.

You can use the nonscrolling region to:

- Display topic titles.

That way users don't have to remember what topic they are reading when they scroll.

GRAPHIC

- Keep one kind of information anchored at the beginning of the topic.

For example, you can display syntax statements in the nonscrolling region so that users can refer to them as they read the parameter descriptions.

GRAPHIC

- Display custom controls that change with the content of each topic.

Because the nonscrolling region is part of each topic, the controls you place in it can be limited to helping the user with the information in the current topic.

GRAPHIC

- Display information that you never want to scroll.

The nonscrolling region can contain entire topics, especially if you want to use all the available window space without having scroll bars cover up part of the information. For example, you can create a topic that just contains a graphic, and you can display that topic in the nonscrolling region of a secondary window that is the same size as the graphic.

GRAPHIC

- Create two different views of the same information.

Help can display either the nonscrolling or scrolling region of a topic in a pop-up window. Help displays the region that contains the context-string footnote (# footnote) in the pop-up window. If you include the context-string footnote in the scrolling region, you can have some hot spots in the Help file that display the nonscrolling region in a pop-up window and some hot spots that display the entire topic in the main

window.

How the Nonscrolling Region Works

To differentiate the nonscrolling region from the scrolling region of a topic, Help places a thin line between the two regions. If the scrolling region contains a vertical scroll bar, the scroll bar does not extend beyond the line into the nonscrolling region. The width of the nonscrolling region is the same as the window in which it appears. Therefore, changing the width of the Help window also changes the width of the nonscrolling region.

Any text that appears in the nonscrolling region wraps to fit in the visible portion of the window unless the text has been formatted as nonwrapping. If the nonwrapping text does not fit in the current Help window size, users must make the Help window larger to see it,because the horizontal scroll bar does not scroll text in the nonscrolling region.

The height of the nonscrolling region is determined by the amount of information you include in the nonscrolling region. A nonscrolling region can include any Windows Help feature that appears in the scrolling region, including text, bitmaps, and hot spots. Help includes hot spots from both regions in the tabbing order, the hot spots in the nonscrolling region coming first in the order.

Users can copy or print the information in a nonscrolling region. The information in the nonscrolling region appears first when copied or printed. The line that separates the nonscrolling and scrolling regions is not copied to the Clipboard or printed with the topic.

The background color of the nonscrolling region is determined either by a predefined setting that the author specifies in the [WINDOWS] section of the Help project file or by the user’s default system colors. If the nonscrolling region includes pop-up window hot spots, the background color in the pop-up windows will match the scrolling region rather than the colors in the nonscrolling region.

Creating a Nonscrolling Region

Help displays topic information in a nonscrolling region only if a Help author creates one by applying the Keep with Next formatting attribute to at least the first paragraph in a topic. If the first paragraph is not formatted as Keep With

Next, Help does not create a nonscrolling region. Also, Help ignores any random paragraphs within a topic that might be formatted as Keep With Next.

To create a nonscrolling region

1. Be sure the text and graphics you want to place in the nonscrolling region are the first paragraphs of the topic.
2. Select the paragraph (or paragraphs) containing the information.
3. From the Format menu, choose Paragraph.
4. Select the Keep With Next check box, and then choose OK.

The formatted paragraph(s) will be placed in a nonscrolling region when you build the Help file.

Setting the Background Color

To make the nonscrolling region visually distinct from the scrolling region, you can change the background color. You specify a background color for a nonscrolling region in the [WINDOWS] section of the Help project file. For example, the following entry in the Help project file tells Windows Help to use a light gray background for the nonscrolling region in the main Help window:

```
[WINDOWS]
main="Command Reference", (0,0,1023,1023), , , (192,192,192)
```

The numbers (192, 192, 192) are the only numbers in this entry you need to change to set the background color. They represent the RGB components of the background color.

Hint

Use the Color application in the Windows Control Panel to see the colors resulting from different RGB values. Choose Color Palette, and then choose Define Custom Colors.

To change the foreground color (in other words, the text), simply format the text using the appropriate Color option in Word for Windows. (For more information, see the [WINDOWS] section in Chapter 16, “The Help Project File.”)

Pop-Up Windows

Within topics, certain terms or concepts often require further explanation. Instead of having Windows Help jump away from the current topic to provide the additional information, you can have Help display the information in a *pop-up window* on top of the current topic. A pop-up window is a temporary window you can use to display subordinate or complementary information. Pop-up windows have many uses but are very effective when used to display any subordinate information that complements the information in the main topic. Here are a few of the most common uses.

You can use pop-up windows to:

- Display glossary definitions of difficult or special terms in the application software or Help file.

GRAPHIC

- Display example text or graphics of information that is explained in the main topic.

GRAPHIC

- Display a list of cross-references for the current topic information.

GRAPHIC

- Complement a hypergraphic by displaying a brief explanation of the object when the user chooses a hot spot.

GRAPHIC

- Provide quick context-sensitive Help on dialog box options, commands, and other interface elements in the application.

GRAPHIC

- Display examples, hints, tips, and notes.

GRAPHIC

How Pop-Up Windows Work

Pop-up windows are stripped-down windows that do not include any standard window elements, such as title bar, menu bar, Control menu, scroll bars, or even

standard-width window borders. They do include a shadow to make them more distinguishable from the main window. Users cannot move or resize pop-up windows.

Popup-topics can include most standard Help features, including any mixture of text, graphics, hot spots (jump hot spots, pop-up hot spots, or macro hot spots), and embedded windows. However, if you have a hot spot within a pop-up window, the first pop-up window closes when the users chooses the hot spot. In other words, you can display only one pop-up window at a time on the screen. Topics displayed in pop-up windows cannot use topic-entry macros, but they can contain jumps and macros executed from hot spots.

Pop-up windows appear on demand, usually when the user chooses a menu item, button, or hot spot. A macro in the Help project file or a **WinHelp** API call can also be used to display a pop-up window. For information about using Help macros to display a pop-up window, see Chapter 15, “Help Macro Reference.” For information about how an application can send an API call to Help in order to display a pop-up window, see Chapter 19, “The WinHelp API.”

Pop-up windows remain open until the user clicks the mouse button or presses any key (which includes choosing a hot spot within the pop-up window).

When displaying pop-up windows, Help follows these guidelines:

- The pop-up window has approximately the same width as the Help window.
- The pop-up window has only as much vertical height as necessary to display all the text. If the pop-up topic contains more text than will fit in a full-size pop-up window, the user will not be able to see the hidden portion of the topic.
- Help attempts to position the pop-up window immediately below the hot spot so that the user can read both the text in the pop-up and the hot spot text that initiated the pop-up window.
- If the topic to be displayed in the pop-up window includes both a nonscrolling region and a scrolling region, Help displays the region that contains the context-string footnote (# footnote) in the pop-up window.

Creating a Pop-Up Window

Creating pop-up windows involves two steps:

- Creating the topic information that will be displayed in the pop-up window
- Creating the hot spot or custom control in the main topic that will access the pop-up topic

Creating Pop-Up Topics

To create a pop-up window, you must first create the topic contents that will be displayed in the pop-up window. Pop-up topics can reside in the same file as regular topics, or they can be placed in a separate topic file. Like regular topics, pop-up topics are referenced by their unique context strings.

Creating Pop-Up Window Hot Spots

After creating pop-up topics, you create pop-up hot spots for displaying the pop-up window. The pop-up hot spot can be text, a bitmap, a hypergraphic hot spot, a menu item, a Help button, a custom control, or a macro hot spot. The coding for a pop-up window hot spot has two parts:

- A word or phrase, formatted as underlined text (the hot spot text), that the user chooses to display the pop-up window
- The context string, formatted as hidden text, that identifies the pop-up topic (what the user sees in the pop-up window)

To create a pop-up hot spot using text or graphics

- Follow the steps to create a standard pop-up hot spot.

For more information, see Chapter 8, “Creating Links and Hot Spots.”

To create a pop-up hot spot using a hypergraphic

- Define a hot spot and then select Pop-up as the hot spot type in the Attributes dialog box of Hotspot Editor.

For more information, see Chapter 11, “Creating Hypergraphics.”

To create a pop-up hot spot using a Help menu item or button

- Use the **ChangeButtonBinding** macro to assign a pop-up macro to the standard menu item or button you want to use to display the pop-up window.
  Or create your own custom menu item or button and define a pop-up macro for it.
  For more information, see Chapter 13, “Customizing the Help File,” and Chapter 15, “Help Macro Reference.”

To create a pop-up hot spot using a Help macro

- Follow the steps to create a standard macro hot spot and define a pop-up macro as the macro to execute when the user chooses the hot spot.
  For more information, see Chapter 13, “Customizing the Help File,”

Embedded Windows

and Chapter 15, “Help Macro Reference.”

If you want to extend the functionality of Windows Help, you can create *embedded windows* for your Help file. An *embedded window* is a rectangular region within a topic that uses a dynamic-link library (DLL) to display information. Embedded windows can display many different kinds of information but are especially useful for displaying information that will maintain a consistent size within a topic. Here are a few possibilities.

You can use embedded windows to:

- Display 256-color bitmaps.
- Display multimedia elements and their playback controls, such as audio files, animations, and full-motion video.
- Assemble and display the contents of a Help topic dynamically based on information stored in an ASCII text file or database.
- Display screen images, such as interface elements and dialog boxes, stored in the application’s resource files to save disk space in the

compiled Help file.

- Display any other information that has special display requirements or that cannot be included using Help's standard feature set.

Whether you use embedded windows and how you use them depends entirely on your willingness and ability to create a dynamic-link library to control the contents within the embedded window. Help does not provide any custom DLLs for you.

How Embedded Windows Work in Help

An embedded window is simply a window (child window) within the topic window. Embedded windows do not have any standard window elements; they are just rectangular regions. Users cannot minimize, maximize, or resize an embedded window, and Help authors cannot use embedded windows as hot spots. However, you can include hot spots in an embedded window that are controlled by the DLL, but users will not be able to use the keyboard equivalents to access the hot spots. That is because embedded windows cannot receive the input focus which means they cannot process keystrokes. So, you should not place anything in an embedded window that requires keyboard input from users.

Windows Help positions the embedded window using the justification character (left, right, or character) specified by the author in the embedded window reference. The embedded window DLL is expected to display the information in the window and resize the window appropriately. Help expects the window size to remain fixed as long as the topic is displayed. The window element and DLL determine the size and content of the embedded window, and Help arranges the other elements of the topic around the embedded window.

Embedded windows are supported by DLLs that have specific components required by the embedded window interface. Developers familiar with DLL programming can write DLLs to support new user-interface or display elements not available in the standard Windows Help feature set. For example, developers can create a DLL to display 256-color bitmaps in an embedded window. For a discussion of the embedded window interface and for a description of a sample DLL, see Chapter 20, "Writing DLLs for Windows Help.".xe "Embedded window:controlling with DLLs:creating window"§

Windows Help displays embedded windows only when necessary–while the topic containing the embedded window is being displayed. However, an embedded window may exist while it is not being displayed (if the topic scrolls past the

embedded window, for example). Because it is part of a specific topic, an embedded window goes away when the user displays a different topic.

Creating an Embedded Window

To create an embedded window, you insert an embedded window reference in the RTF topic file where you want the window to appear. This reference has the following general form:xe "DLLs:embedded-window"§

{ew*x DLL-name, window-class, author-data***}**

Parameter	Description
x	A character specifying the alignment of the window: **l** for a left-justified window, **r** for a right-justified window, or **c** for a character-aligned window.
DLL-name	The name of the DLL that controls the embedded window. The filename should not include an extension or be fully qualified, but it can include a relative path. Windows Help assumes .DLL or .EXE to be the default extension.
window-class	The name of the embedded window class as defined in the source code for the DLL.
author-data	An arbitrary string, which Windows Help passes to the embedded window when it creates the window. This string can be one or more substrings separated by any punctuation mark except a comma. The DLL is responsible for parsing this string. The string is terminated by the closing brace (})xe "Embedded window:controlling with DLLs:creating window"§

Note

You may need to consult the developer of the DLL to obtain the information in the embedded window reference.

The following example shows a valid embedded window reference:

{ewl FADE, AmfWnd, clipbrd.amf}

Positioning Embedded Windows

Embedded windows are displayed within a topic paragraph. Often, the topic paragraph contains text to be positioned in relation to the embedded window element. Help supports three different positions for embedded windows– character-aligned, left-justified, or right-justified.

The following table describes the position attributes that control the placement of an embedded window in relation to the paragraph text.

Attribute	Description
Character	The embedded window is displayed at the exact position in which it was inserted. Help adjusts the height of the line containing the embedded window to allow space for the window.
Left	The left edge of the embedded window aligns with the left paragraph margin, and the paragraph text wraps to the right of the embedded window.
Right	The right edge of the embedded window aligns with the right paragraph margin, and the paragraph text wraps to the left of the embedded window.

Each position defines how the embedded window is positioned with respect to the topic text, and each position requires a slightly different command reference in the topic file:

character **{ewc** *DLL-name*, *window-class*, *author-data***}**

left-aligned **{ewl** *DLL-name, window-class, author-data***}**

right-aligned **{ewr** *DLL-name, window-class, author-data***}**

Character-Aligned Embedded Windows (ewc)

The **ewc** (embedded window character) command causes Windows Help to treat the window as a text character. It aligns the window on the baseline of the type in exactly the same place in the paragraph where the command occurs. Because the window is treated as text, paragraph formatting properties assigned to the paragraph also apply to the window. Text coming before or after the window does not wrap around the window. Help adjusts the height of the line containing the embedded window to allow enough space for the embedded window.

Figure 9.x shows how Word for Windows displays the embedded window reference in the topic file.

Graphic

After you compile your topic file, Windows Help displays the embedded window, as in Figure 9.x.

Graphic

You can display multiple character-aligned embedded windows on the same line. Help adjusts the line height to allow for the tallest embedded window.

Graphic

Note

Don’t specify absolute line spacing (also called “negative line spacing” or “exact line spacing”) for a paragraph that has an **ewc** reference. If you do, the embedded window might appear on top of the succeeding paragraph when Windows Help displays the topic.

Left- and Right-Justified Embedded Windows (ewl, ewr)

The **ewl** (embedded window left) command and **ewr** (embedded window right) command cause Windows Help to place the embedded window along the left or right paragraph margin and wrap the paragraph text around the edge of the embedded window. Text is aligned with the upper-right or upper-left corner of the window.

You normally place **ewl** and **ewr** references at the beginning of a paragraph to ensure proper wrapping of text around the right edge of the window. For example, typical **ewl** and **ewr** references are formatted as follows:

{ewl FADE, AmfWnd, clipbrd.amf}Paragraph text follows the embedded window reference...

{ewr FADE, AmfWnd, clipbrd.amf}Paragraph text follows the embedded window reference...

Figure 9.x shows how **ewl** and **ewr** references appear in the Word for Windows topic file.

Graphic

Note

Do not put any space characters between the **ewl** or **ewr** reference and the paragraph text unless you want the first line of text indented from the rest of the text that wraps along the right edge of the window (**ewl**) or from the left topic margin (**ewr**). To ensure that text wraps correctly around an embedded window, insert a soft carriage return at the end of each line of text.

After you compile the topic file, Windows Help displays the embedded window to the left or right of the text, as in Figure 9.x.

Graphic

You can also put an **ewl** or **ewr** reference at the end of a paragraph, as in this example:

Paragraph text precedes the embedded window reference.{ewl FADE, AmfWnd, clipbrd.amf}

Paragraph text precedes the embedded window reference.{ewr FADE, AmfWnd, clipbrd.amf}

Figure 9.x shows how this type of embedded window reference appears in the Word for Windows topic file.

Graphic

When you include an **ewl** or **ewr** reference this way, Windows Help wraps the text to the paragraph end and then displays the embedded window. After you compile the file, the embedded window appears under the text and to the left or right, as in Figure 9.x.

Graphic

Left and Right Margins

The paragraph indent determines the distance between the window border and the left or right edge of the embedded window.

Note

Within tables, the paragraph indent determines the distance between the cell border and the left or right edge of the embedded window.

Help reformats paragraphs when the user changes the horizontal width of the window. The resulting changes in word wrapping can change the position of embedded windows within the paragraph.

For example, in Figure 9.x, the right margin setting determines the distance of the right-aligned embedded window from the window border.

Graphic

In Figure 9.x, the user has reduced the width of the window, and the embedded window has changed position:

Graphic

Note

The Word Keep Lines Together attribute, used to prevent Help from word-wrapping a paragraph, does not prevent Help from moving a left- or right-aligned embedded window in relation to the window border. The window will move when the window is sized, but the paragraph text won't move.

Vertical Positioning

The vertical position of a left- or right-aligned embedded window depends on the insertion point of the element within the topic paragraph. Help uses the following rules to determine the vertical position:

- If the element is the first item in the paragraph, the embedded window is displayed in the first line of the paragraph.

- If the element is not the first item in the paragraph, the embedded window is displayed in the line immediately below the insertion point.

Left- and right-aligned windows are displayed with their top edge aligned with the tallest character in the line. The top edge of the embedded window is aligned with the top edge of the line, and the rest of the embedded window extends down into the paragraph. Figure 9.x shows a left- or right-aligned embedded window is positioned in a line.

Graphic

Note

Because character-aligned embedded windows are treated as large characters, they can increase the height of a line. This will affect the positioning of left- or right-aligned embedded windows displayed in the same line.

Figure 9.x shows three topic paragraphs displayed in Word and in Help. Each paragraph has identical content and Word formatting, but the insertion point of the embedded window changes from one to the next. In this example, the three embedded windows are left-aligned.

Graphic

<three paragraphs in Word and Help, with elements in three different positions: first char, after first word, and last char>

In Figure 9.x, the three embedded windows are right-aligned.

Graphic

<three paragraphs in Word and Help, with elements in three different positions: first char, after first word, and last char>

Help cannot display multiple left- or right-aligned embedded windows in the same vertical position. If you have multiple left- or right-aligned embedded windows inserted in the same position within the paragraph, Help displays the windows in different vertical positions within the paragraph. For example, Figure 9.x shows a topic paragraph displayed in Word and in Help.

Graphic

<Word paragraph. Callouts identifying icons as left- and right-aligned bitmaps>

Using Tables to Position Embedded Windows

If you need to display multiple embedded windows in the same vertical position, you can use tables to position the embedded windows. Table formatting lets you more closely control the position of embedded windows within a paragraph. For example, you can position two or more left- or right-aligned windows within the same vertical plane. Or you can place an embedded window in the center of a paragraph. Paragraphs within tables are not reformatted when the window is sized.

Left- and right-aligned embedded windows are aligned with the cell borders. The word-wrapping of paragraphs in tables is not affected by changes in window size position, so if you want to prevent an embedded window from moving with the window border, you must place the containing paragraph in a table cell.

Figure 9.x shows a Help topic that uses tables to create some sophisticated paragraph layouts.

Graphic

How Help Locates .DLLs

When executing custom DLLs, Windows Help loads the DLL only when it is needed by the Help file. To load a DLL, Help must be able to find it on the user's system. When preparing to use a DLL, Help looks in the following locations:

- Help's current directory.
- The MS-DOS current directory.
- The user's Windows directory.
- The Windows SYSTEM directory.
- The directory containing WINHELP.EXE.
- The directories listed in the user's PATH environment variable.
- The directories specified in WINHELP.INI.

If Help cannot find the DLL after searching in all these locations, it displays an error message.

To increase the likelihood that Help will locate the DLL quickly, you should also

observe the following guidelines:

- Use unique names for all DLLs accessed by the Help file.
- When installing your application on a user's hard disk drive, your setup program should copy all custom DLLs to the directory where Help is located.
- If your product is distributed on CD-ROM, copy WINHELP.EXE and any custom DLLs to the user's hard disk drive.
- Define a WINHELP.INI entry for each custom DLL that your Help file is using so that Help knows where to locate them.

For an explanation of the WINHELP.INI file, see the "Creating Links Between Help Files" section in Chapter 8, "Creating Links and Hot Spots."