
Providing Help Tags in Carbon

[Carbon](#) > [User Experience](#)



2002-07-01



Apple Inc.
© 2002 Apple Computer, Inc.
All rights reserved.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, mechanical, electronic, photocopying, recording, or otherwise, without prior written permission of Apple Inc., with the following exceptions: Any person is hereby authorized to store documentation on a single computer for personal use only and to print copies of documentation for personal use provided that the documentation contains Apple's copyright notice.

The Apple logo is a trademark of Apple Inc.

Use of the "keyboard" Apple logo (Option-Shift-K) for commercial purposes without the prior written consent of Apple may constitute trademark infringement and unfair competition in violation of federal and state laws.

No licenses, express or implied, are granted with respect to any of the technology described in this document. Apple retains all intellectual property rights associated with the technology described in this document. This document is intended to assist application developers to develop applications only for Apple-labeled computers.

Every effort has been made to ensure that the information in this document is accurate. Apple is not responsible for typographical errors.

Apple Inc.
1 Infinite Loop
Cupertino, CA 95014
408-996-1010

Apple, the Apple logo, Aqua, Carbon, Mac, Mac OS, Macintosh, and QuickDraw are trademarks of Apple Inc., registered in the United States and other countries.

Finder and Shuffle are trademarks of Apple Inc.

Simultaneously published in the United States and Canada.

Even though Apple has reviewed this document, APPLE MAKES NO WARRANTY OR REPRESENTATION, EITHER EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT, ITS QUALITY, ACCURACY, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. AS A RESULT, THIS DOCUMENT IS PROVIDED "AS IS," AND YOU, THE READER, ARE ASSUMING THE ENTIRE RISK AS TO ITS QUALITY AND ACCURACY.

IN NO EVENT WILL APPLE BE LIABLE FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES RESULTING FROM ANY DEFECT OR INACCURACY IN THIS DOCUMENT, even if advised of the possibility of such damages.

THE WARRANTY AND REMEDIES SET FORTH ABOVE ARE EXCLUSIVE AND IN LIEU OF ALL OTHERS, ORAL OR WRITTEN, EXPRESS OR IMPLIED. No Apple dealer, agent, or employee is authorized to make any modification, extension, or addition to this warranty.

Some states do not allow the exclusion or limitation of implied warranties or liability for incidental or consequential damages, so the above limitation or exclusion may not apply to you. This warranty gives you specific legal rights, and you may also have other rights which vary from state to state.

Contents

Introduction **Introduction to Providing Help Tags in Carbon** 7

See Also 7

Chapter 1 **Carbon Help Manager Concepts** 9

Help Tags 9
How Help Tags Operate 9
The Help Tag Structure 10
 The Hot Rectangle 10
 Display Location 11
 Help Tag Content 14
Writing Help Tag Content 16

Chapter 2 **Carbon Help Manager Tasks for Adding Help Tags** 19

Using Interface Builder to Create Help Tags 19
Attaching a Help Tag Directly to a User Interface Element 21
Providing Help Tags With a Help Tag Callback 23
 Installing a Help Tag Callback 24
 Supplying Help Tag Content With Callbacks 25
 Disposing of Help Tag Content 27
Displaying a Help Tag at an Application-Defined Location 27
Creating a Help Tag for a Menu Title in Versions of Mac OS X Prior to 10.2 28

Document Revision History 31

Figures and Listings

Chapter 1 **Carbon Help Manager Concepts 9**

- Figure 1-1 A help tag 9
- Figure 1-2 The hot rectangle for a help tag associated with a control 10
- Figure 1-3 A help tag displayed at the default location 11
- Figure 1-4 Help tag locations above and below the hot rectangle 12
- Figure 1-5 Help tag locations to the right and left of the hot rectangle 13
- Figure 1-6 Help tag locations inside the hot rectangle 14
- Figure 1-7 A help tag 14
- Figure 1-8 An expanded help tag 15

Chapter 2 **Carbon Help Manager Tasks for Adding Help Tags 19**

- Figure 2-1 Help in the Info window 20
- Figure 2-2 Selecting the help tag location 21
- Figure 2-3 A help tag explicitly displayed by the Finder 28
- Listing 2-1 A function that creates a help tag and attaches it a window 22
- Listing 2-2 Installing a help tag callback with a control 24
- Listing 2-3 A help tag callback for a control 25
- Listing 2-4 A help tag callback for a menu title in Mac OS X 10.1.x and earlier 29

Introduction to Providing Help Tags in Carbon

This document describes how to use the Carbon Help Manager to add help tags to your user interface.

Help tags are short contextual help messages that appear onscreen when the user hovers the pointer over an element in an application's user interface. They allow an application to identify the objects in its user interface. Help tags replace Balloon Help, available in Mac OS 9.2 and earlier. Help tags supplement user assistance provided by Apple Help help books.

Note: Apple Help is the primary help system on the Macintosh; applications can use it to display online documentation in Help Viewer, a browser-like application. This user help is stored as a collection of HTML documents called a **help book**.

The Carbon Help Manager is an application programming interface (API) that allows your application to create and display help tags. The Carbon Help Manager also provides a Help menu through which users can access your application's help books and other help resources. The Carbon Help Manager is available in Mac OS X 10.0 and later, and in Mac OS 8.1 and later when CarbonLib 1.0.4 and later is present.

If you are writing a Carbon application with a user interface, you should read this document to learn how you can add help content to your interface. [Chapter 2, "Carbon Help Manager Concepts"](#), (page 9) gives an overview of help tags and how they are displayed by the Carbon Help Manager, as well as guidelines for writing help tag content. [Chapter 3, "Carbon Help Manager Tasks for Adding Help Tags"](#), (page 19) provides examples and code listings showing you how to create help tags.

See Also

For a detailed discussion of the functions and data types in the Carbon Help Manager API, see the *Carbon Help Manager Reference*

For more information on using Apple Help to create HTML-based user assistance, see the *Apple Help Reference*

INTRODUCTION

Introduction to Providing Help Tags in Carbon

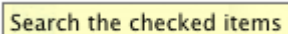
Carbon Help Manager Concepts

This chapter gives a brief description of help tags and explains how they are displayed by the Carbon Help Manager. This chapter also provides guidelines for writing effective help tag messages.

Help Tags

A **help tag** is a small window containing a short help message that is visually associated with an element in an application's user interface. A help tag is displayed when the user pauses with the pointer over the user interface element with which it is associated. The tag disappears when user input occurs. Help tags are always enabled. Figure 2-1 shows a help tag.

Figure 1-1 A help tag



Search the checked items

Help tags assist users by identifying your application's interface elements and their purposes. You can quickly and effectively communicate to more experienced users what they can accomplish in your application without referring them to your full suite of documentation. You can create help tags for controls, windows, menu titles, and menu items. You can also display a help tag directly on the screen without associating the tag with an interface element.

How Help Tags Operate

The Carbon Help Manager automatically displays help tags as the user moves the pointer around the screen. When a mouse-moved event occurs, the Carbon Help Manager records the current position of the pointer. If the pointer does not move from that position within the current help tag display time (by default, 1.5 seconds) and no other user input occurs, the Carbon Help Manager examines the user interface element underneath the pointer. If it has help content associated with it, the Carbon Help Manager displays a help tag with that content. When the user moves the pointer, clicks the mouse, presses a key on the keyboard, or uses a mouse scroll wheel, the Carbon Help Manager removes the help tag from the screen.

Your application can disable the Carbon Help Manager's automatic display of help tags with the `HMSetHelpTagsDisplayed` function, but this function affects help tag display in only *your* application. Help tag display in other applications is unaffected.

To supply help tag content, your application must

- define a help tag structure to describe the tag
- associate the help content with the appropriate interface element

You can associate help tag content with an interface element in one of two ways:

- attach the help tag content directly to the element
- attach a help tag callback to the element

The Carbon Help Manager tracks the interface element for you; when the pointer pauses over the item, the Carbon Help Manager determines if either help content or a help tag callback is associated with the element. If the content is directly attached to the element, the Carbon Help Manager displays a help tag with the content. If you have installed a callback, the Carbon Help Manager sends a request for the help content to your callback, and then displays the content that your callback returns in a help tag.

The Help Tag Structure

A help tag is described by the help tag structure, `HMHelpContentRec`. The help tag structure contains the following information:

- the structure version
- the hot rectangle associated with the help tag
- the display location for the help tag
- the help content

The Hot Rectangle

The **hot rectangle** defines the area on the screen over which the pointer must pause to trigger the display of the help tag. As the user moves the pointer onto and off the hot rectangle, the Carbon Help Manager displays and removes the tag. You can specify an empty hot rectangle—a rectangle with coordinates (0,0,0,0). Before displaying a help tag, the Carbon Help Manager verifies that the current pointer position is within the hot rectangle associated with the tag. If the hot rectangle is empty, the Carbon Help Manager automatically substitutes the current bounds of the interface element with which the tag is associated, before performing this check. Figure 2-2 shows the hot rectangle for a help tag associated with a control.

Figure 1-2 The hot rectangle for a help tag associated with a control

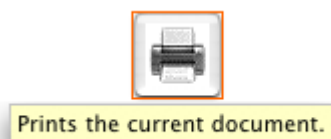


Note: Specifying an empty hot rectangle from a help tag callback is only supported on Mac OS X version 10.2 and later. On earlier versions of Mac OS X, you must provide a valid hot rectangle for the help tag.

Display Location

The location at which a help tag is displayed is calculated relative to the hot rectangle. By default, help tags are displayed below the hot rectangle, centered horizontally. Figure 2-3 shows a help tag displayed at the default location, relative to the bounds of the control that it describes.

Figure 1-3 A help tag displayed at the default location



Your application can change the location at which tags are displayed on a per-tag basis. The Carbon Help Manager provides constants to describe the various locations, relative to the hot rectangle, at which help tags can be displayed. Figure 2-4 shows the positions above and below the hot rectangle at which help tags can be displayed. You can use the constant listed below each example to specify the given help tag location to the Carbon Help Manager.

Figure 1-4 Help tag locations above and below the hot rectangle

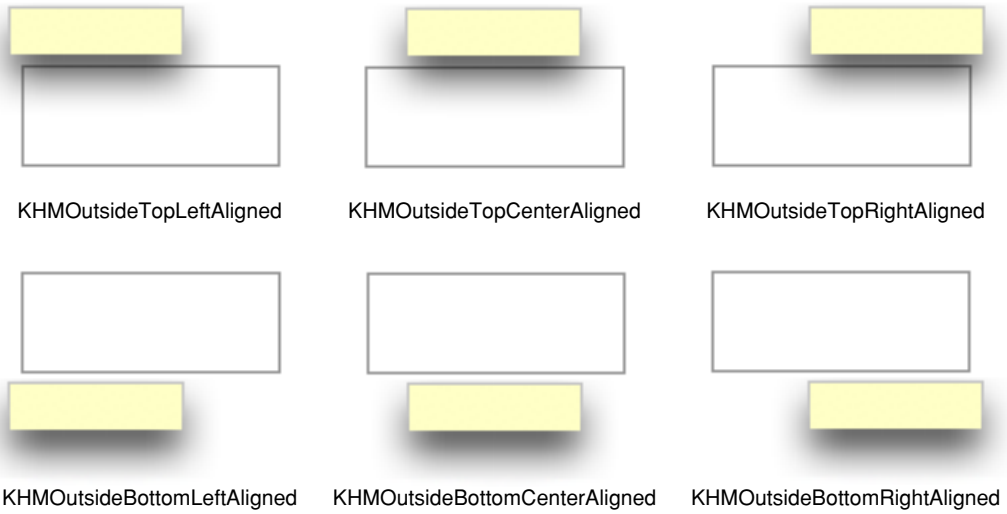
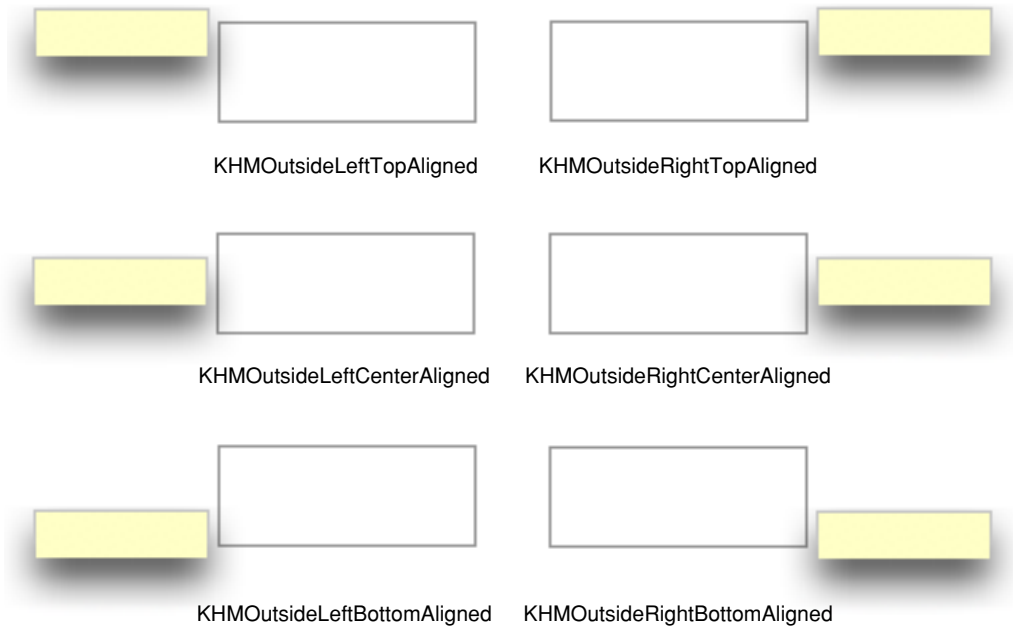


Figure 2-5 shows the locations to the right and left of the hot rectangle at which you can display help tags.

Figure 1-5 Help tag locations to the right and left of the hot rectangle



Help tags can also be displayed inside the bounds of the hot rectangle. Figure 2-6 shows the locations inside of the hot rectangle at which help tags can be displayed.

Figure 1-6 Help tag locations inside the hot rectangle

The Carbon Help Manager constants `kHMOutsideTopScriptAligned` and `kHMOutsideBottomScriptAligned` specify locations above and below the hot rectangle, respectively. However, the help tags are aligned with the right or left edges of the hot rectangle based on the orientation of the current system script. If the orientation of the system script is left-to-right (as in the Roman script system), the constants `kHMOutsideTopScriptAligned` and `kHMOutsideBottomScriptAligned` describe the same locations as the constants `kHMOutsideTopLeftAligned` and `kHMOutsideBottomLeftAligned`, respectively. If the orientation of the system script is right-to-left (as in the Hebrew or Arabic script system), `kHMOutsideTopScriptAligned` and `kHMOutsideBottomScriptAligned` describe the same locations as the constants `kHMOutsideTopRightAligned` and `kHMOutsideBottomRightAligned`.

Help Tag Content

The text of a help tag is described in an `HMHelpContent` structure. Each help tag structure (`HMHelpContentRec`) contains an array of two `HMHelpContent` structures; one for the help tag's minimum, or default, content and one for its maximum, or expanded, content.

The minimum content is displayed by default when the user moves the pointer over the interface element with which the tag is associated. Figure 2-7 shows a help tag displaying the default content.

Figure 1-7 A help tag

Maximum content is displayed when the user presses the Command key as the pointer hovers over the item. Figure 2-8 shows a help tag displaying the expanded content.

Figure 1-8 An expanded help tag



When the user moves the pointer onto the item, the Carbon Help Manager displays either the default or expanded content, depending upon whether the Command key is down. If the user presses or releases the Command key after the help tag has been displayed, without moving the mouse off the item, the Carbon Help Manager removes the previous tag and displays a tag with the other content.

Default content is mandatory for a help tag; expanded content is optional. Expanded content should be used sparingly to

- further explain what the item does
- explain why or when the user would want to use the item
- provide additional information about how the user uses the item, if it is not clear from the item itself

The Carbon Help Manager supports a number of formats for the help content. However, help tags display only text; they do not support images, icons, or other graphical data. You can supply help content in any of the following forms:

- a Core Foundation string (`CFString`)
- a Pascal string
- a 'STR#' resource
- a 'STR' resource
- a TextEdit handle
- a 'TEXT' resource

The last two formats are not supported in versions of Mac OS X prior to Mac OS X version 10.2.

Important: Apple recommends that you use Core Foundation strings for your help tag content. A `CFString` represents an array of Unicode characters and a count of the number of characters. Because `CFString` objects are Unicode-based, you can easily localize your help tag content for most of the world's written languages. Only `CFString` objects can represent all possible characters in all of the languages currently supported in Mac OS X.

In Mac OS X version 10.2 and later, the Carbon Help Manager will retrieve localized help strings stored in a `Localizable.strings` file. When you specify help content of type `kHMCFStringLocalizedContent`, the value in the `content` field of the help content structure should be a `CFString` key identifying the help tag message. The Carbon Help Manager checks the `Localizable.strings` file in the appropriate language folder and returns the localized help string associated with that key.

Writing Help Tag Content

Help tags are designed to allow your application to briefly tell users

- what an object is
- what action the user can take with the object
- how to use the object

Because tags appear next to the user interface elements that they describe, information you supply in the tag unambiguously applies to the element. Help tags are therefore a very effective form of communicating short hints to users who already have some idea of what they want to accomplish or have enough knowledge to achieve their goals simply by experimenting with your user interface. Help tags are not designed to

- explain why users should use your application
- teach users how to use your application
- explain complex tasks

Inside Mac OS X: Aqua Human Interface Guidelines offers the following suggestions to help you create effective help tags:

- Use the fewest words possible. Try to keep your tags to a maximum of 60 to 75 characters. Since help tags are always on, it is important to keep your tag text unobtrusive—that is, *short*—and useful. Present one concept per tag and make sure the concept is directly related to the item. Localization lengthens the text by 20 to 30 percent, which is another good reason to keep the tag short.
- Write the main help tag in any of these ways, depending on the interface you're documenting:
 - Describe what the user will accomplish by using the control. Examples: "Add or remove a language from your list." "Reduce red tint in the selected area." Most help tags can use this format.
 - Give extra information to explain the results of the user's action. This kind of tag is most effective in an interface that already includes some instructional text, because the tag and the interface text work together to describe what the control does and how the user manipulates it.
 - Define terms that may be unknown to the user. This kind of tag should be used only if the interface already contains instructions to the user.
- You can create contextually sensitive help tags but you don't have to; the same text can appear when an item is selected, dimmed, and so on. By describing what the item accomplishes, you may help the user understand the current state of the control even if the tag is applicable to all situations.
- Use help tags to provide functional information for controls that are unique to your application. Don't tag window controls, scroll bars, and other parts of the standard Mac OS X interface.
- Don't put the item's name in the tag unless the name helps the user and isn't available onscreen. If an item is referred to by name in the documentation and in the tag, make sure the names match.
- You can use a sentence fragment beginning with a verb, for example, "Restores default settings." You can also omit articles to limit the size of the tag. If the tag text is a complete sentence, end it with a period.
- Describe only the item the user points to.
- Use help tags primarily to provide necessary information, rather than incidental tips.

- If you implement an expanded tag to add another layer of information, don't repeat the text in the original tag. An expanded tag should do one of the following:
 - More fully explain or describe the results of the action described in the small help tag.
Help tag: Shuffles the play order. *Expanded tag:* Plays the current list of songs in random order.
 - Explain when or why the user would do the action described in the small tag.
Help tag: Creates a playlist. *Expanded tag:* Playlists are groups of songs and are used to create CDs.

If the original tag is an explanation or a definition that helps supplement instructions in the interface, you're less likely to need an expanded tag.

Carbon Help Manager Tasks for Adding Help Tags

This chapter contains instructions and sample code showing you how to use the Carbon Help Manager to add help tags to your application's user interface.

The code samples assume you are developing your application on the Mac OS using Carbon. All code samples in this chapter are in C.

You have several options for creating and displaying help tags in your application. You can use any one of the following methods, or any combination of them:

- Specify help tags in Interface Builder, Apple's graphical interface editor. This is the simplest way to create help tags for your application's user interface.
- Call Carbon Help Manager functions to attach help content directly to objects in your application's user interface.
- Install help tag callbacks for objects in your application's user interface. Use this approach to dynamically determine help content.
- Directly display a help tag. If your application needs to display a help tag that is not associated with a control, window, or menu, you can create a help tag and tell the Carbon Help Manager to display it immediately.

Using Interface Builder to Create Help Tags

If you are using Interface Builder to create your application's user interface, you can use the Info window to easily add help tags to interface elements. Interface Builder stores this help tag content in the nib file, along with other information describing the objects in your application's interface. When you unarchive an object with help content, the help tag is automatically attached to the user interface element using the appropriate Carbon Help Manager `HMSet...Content` call. These functions are described further in the section [“Attaching a Help Tag Directly to a User Interface Element”](#) (page 21). For instructions on using Interface Builder to create your application's user interface, choose Interface Builder Help from the Help menu in Interface Builder.

Note: You can use Interface Builder to create help tags for controls, windows, and menu items. To create a help tag for a menu title, use the function `HMSetMenuItemHelpContent` to attach the help content directly to the menu title, or install a help tag callback.

You cannot specify expanded content for a help tag in Interface Builder. If you wish to create an expanded help tag for a user interface element, you must call the appropriate Carbon Help Manager function to attach the help content to the element, or you must install a help tag callback.

To create a help tag for a user interface element using Interface Builder, perform the following steps:

1. Select the user interface element in the Interface Builder window.

2. Open the Info window by choosing Show Info from the Tools menu, or by pressing Shift-Command-I.
3. Choose Help from the pop-up menu at the top of the Info window. Figure 3-1 shows the Info window after choosing Help.

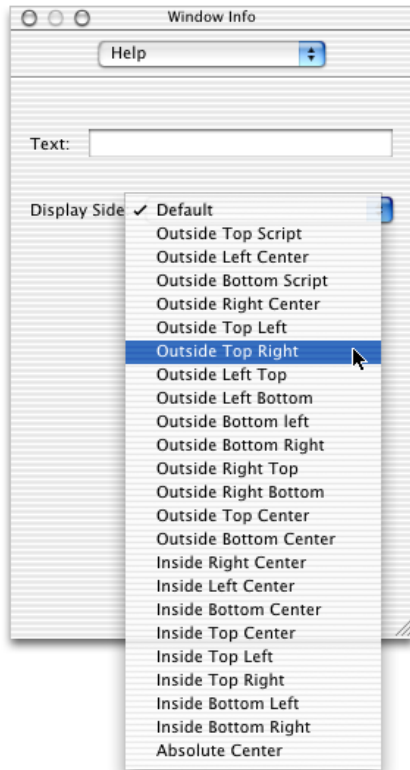
Figure 2-1 Help in the Info window



4. Type the help tag message in the Text field.

- Choose the location for the help tag from the Display Side pop-up menu. The locations listed in this pop-up menu are relative to the hot rectangle for the help tag. These locations correspond to the locations represented by the display location constants described in more detail in the section “Display Location” (page 11). Figure 3-2 shows the Info window with the Display Side pop-up menu open.

Figure 2-2 Selecting the help tag location



Attaching a Help Tag Directly to a User Interface Element

When you attach a help tag directly to a user interface element, the Carbon Help Manager displays and removes the help tag as necessary until you dispose of the interface element. To set the help tag content for a user interface element, you must

- create a help tag structure describing the help tag content for the control, window, or menu item
- attach the help tag to the control, window, or menu item using the appropriate Carbon Help Manager function

The help tag is described by the `HMHelpContentRec` structure. Before attaching the tag to an item in your user interface, you must initialize an `HMHelpContentRec` structure with values for

- the structure version

- the tag's hot rectangle
- the location, relative to the hot rectangle, at which the tag is displayed
- the help tag content

You can use an empty rectangle for the hot rectangle and the Carbon Help Manager will substitute the current bounds of the interface element for the empty rectangle when the help tag is displayed.

Once you have created the help tag structure, you can attach it to the user interface element. To attach the help tag to a window or control, pass a pointer to your `HMHelpContentRec` structure and a reference to the window or the control to either the `HMSetControlHelpContent` or the `HMSetWindowHelpContent` function, respectively. To attach the tag to a menu item, pass a pointer to your help tag, a reference to the menu containing the item, and the index number of the menu item to the `HMSetMenuItemHelpContent` function. Passing a menu item index of zero to `HMSetMenuItemHelpContent` sets the help content for the menu title.

Note: Attaching help content directly to a menu title with `HMSetMenuItemHelpContent` does not work in Mac OS X version 10.1 and earlier. See [“Creating a Help Tag for a Menu Title in Versions of Mac OS X Prior to 10.2”](#) (page 28) for a workaround.

Listing 3-1 shows a function that creates a help tag structure and then attaches it to a window.

Listing 2-1 A function that creates a help tag and attaches it a window

```
pascal OSStatus MyAttachHelpTag(WindowRef window)
{
    OSStatus status;
    HMHelpContentRec helpTag;

    helpTag.version = kMacHelpVersion;           // 1
    helpTag.tagSide = kHMDefaultSide;           // 2
    SetRect (&helpTag.absHotRect, 0, 0, 0, 0); // 3
    helpTag.content[kHMMinimumContentIndex].contentType = // 4
kHMCFStringLocalizedContent;
    helpTag.content[kHMMinimumContentIndex].u.tagCFString = CFSTR("The help tag// 5
for the window.");
    helpTag.content[kHMMaximumContentIndex].contentType = kHMNoContent; // 6

    status = HMSetWindowHelpContent(window, &helpTag); // 7

    return status;
}
```

Here's what the function does:

1. Sets the `version` field of the help tag structure to the current structure version, represented by the `kMacHelpVersion` constant.
2. Sets the location at which the help tag is displayed. Use one of the display location constants in the `tagSide` field to represent the location of the help tag relative to the window's absolute hot rectangle. The system default location, represented by the constant `kHMDefaultSide`, is centered horizontally below the hot rectangle.

3. Calls the QuickDraw function `SetRect` to set the hot rectangle for the help tag. The `MyAttachHelpTag` function sets the hot rectangle to an empty rectangle—that is, a rectangle with coordinates (0,0,0,0). The Carbon Help Manager substitutes the current location of the interface element—in this case, the window—for the empty rectangle when it displays the help tag.
4. Specifies the format for the help tag's default, or minimum, content. All help tags should have minimum content, which is displayed by default when the user moves the pointer over the interface element. The function sets the `contentType` field for the default content to `kHMCStringLocalizedContent`, indicating that the help tag message is a localized string contained in the `Localizable.strings` file.
5. Specifies the CFString key that the Carbon Help Manager should use when retrieving the localized help message. The Carbon Help Manager searches for this key in the `Localizable.strings` file for the appropriate language. If the key is found, the Carbon Help Manager displays the localized help text associated with the key in a help tag.
6. Specifies the format for the help tag's expanded content. Expanded content—which is displayed when the user presses the Command key and moves the pointer over the window—is optional. In this case, the window has no expanded help tag content, so the function sets the `contentType` field of the expanded content to `kHMNoContent`.
7. Calls the Carbon Help Manager function `HMSetWindowHelpContent` to attach the help tag to the window.

Providing Help Tags With a Help Tag Callback

Help tag callbacks allow your application to provide dynamic help tag content, including content that can be determined only while your application is running and content that may change or vary while your application is running. When you install a help tag callback with a control, window, or menu, you do not supply help tag content for that user interface element until the help tag is about to be displayed to the user. For example, you might use callbacks if your help content is stored in an external database. Rather than reading in all help content at application startup and attaching it directly to elements of the interface, your callback can query the database for help tag content only as it is needed.

You may also use help tag callbacks to create context-sensitive help tags. When the Carbon Help Manager notifies you of a request for help tag content for an interface element, you can check the current state of that element and alter the help content accordingly.

The Carbon Help Manager defines help tag callback types for controls, windows, menu items, and menu titles. These are `HMControlContentProcPtr`, `HMWindowContentProcPtr`, `HMMenuItemContentProcPtr` and `HMMenuTitleContentProcPtr`, respectively. See the *Carbon Help Manager Reference* for a detailed description of these callback types.

The Carbon Help Manager invokes your help tag callback when the user hovers the pointer over the control, window, or menu to which the callback is attached. When it calls your help tag callback, the Carbon Help Manager passes the following information to your callback:

- a reference to the user interface element for which to provide help tag content
- a pointer, in the `ioHelpContent` parameter, to the help tag structure for the user interface element's help tag
- a constant, in the `inRequest` parameter, specifying the action that your callback should take

Depending upon the value of the constant in the `inRequest` parameter, your help tag callback must either supply a help tag for the Carbon Help Manager to display, or release any memory used by the help tag content after the Carbon Help Manager has hidden the help tag.

Installing a Help Tag Callback

To install your help tag callback, use the Carbon Help Manager functions `HMInstallControlContentCallback`, `HMInstallWindowContentCallback`, `HMInstallMenuItemContentCallback`, and `HMInstallMenuItemContentCallback`. These functions install a help tag callback with a control, a window, a menu's items, and a menu's title, respectively. The installation functions take a universal procedure pointer (UPP) to your help tag callback and a reference to the user interface element—control, window, or menu—to which the callback should be attached. Listing 3-2 shows how to attach a help tag callback, described in the section “[Supplying Help Tag Content With Callbacks](#)” (page 25), to a control. For a description of the `HMInstallControlContentCallback`, `HMInstallWindowContentCallback`, `HMInstallMenuItemContentCallback`, and `HMInstallMenuItemContentCallback` functions, see the *Carbon Help Manager Reference*.

Listing 2-2 Installing a help tag callback with a control

```
pascal OSStatus MyInstallHelpTagCallback (WindowRef window)
{
    OSStatus status;
    HMControlContentUPP controlHelpTagCallbackUPP;
    ControlRef theControl;
    ControlID theControlID = {kMyControlSig, kMyControlID};

    status = GetControlByID(window, &theControlID, &theControl);           // 1
    controlHelpTagCallbackUPP = NewHMControlContentUPP(MyControlHelpTagCallback); // 2
    status = HMInstallControlContentCallback(theControl,                       // 3
controlHelpTagCallbackUPP);
    return status;
}
```

Here's what the code in Listing 3-2 does:

1. Calls the Control Manager function `GetControlByID` to retrieve a reference to the control to which to attach the help tag callback.
2. Calls the `NewHMControlContentUPP` function to create a universal procedure pointer to the help tag callback for the control.
3. Calls the `HMInstallControlContentCallback` function to associate the help tag callback with the control.

You should dispose of any UPPs created by your application once the callback functions they refer to are no longer needed; for example, after your application has disposed of the user interface element associated with the help tag callback.

Supplying Help Tag Content With Callbacks

When the pointer pauses over a user interface element for which you have installed a help tag callback, the Carbon Help Manager calls your callback with the `kHMSupplyContent` request in the `inRequest` parameter. When your help tag callback receives this request, it must do the following:

- set the hot rectangle of the help tag, in the `absHotRect` field of the help tag structure
- set the location, relative to the hot rectangle, of the help tag in the `tagSide` field of the help tag structure
- set the help content for the specified user interface element in the `contents` field of the help tag structure
- set the `outContentProvided` parameter to `kHMContentProvided`
- return `noErr`

After your callback returns, the Carbon Help Manager displays the help tag supplied by your callback. Listing 3-3 shows a help tag callback for a control.

Listing 2-3 A help tag callback for a control

```
pascal OSErr MyControlHelpTagCallback(ControlRef inControl,
    Point inGlobalMouse, HMContentRequest inRequest,
    HMContentProvidedType *outContentProvided,
    HMHelpContentPtr ioHelpContent)
{
    OSErr status = noErr;

    if (inRequest == kHMSupplyContent) { // 1
        GrafPtr savePort;
        ioHelpContent->version = kMacHelpVersion;
        ioHelpContent->tagSide = kHMDefaultSide; // 2
        if (!QDSwapPort(GetWindowPort(GetControlOwner(inControl)), &savePort)) // 3
            savePort = NULL;
        GetControlBounds(inControl, &ioHelpContent->absHotRect); // 4
        LocalToGlobal((Point*)&ioHelpContent->absHotRect.top); // 5
        LocalToGlobal((Point*)&ioHelpContent->absHotRect.bottom); // 6
        if (savePort != NULL) SetPort(savePort); // 7
        ioHelpContent->content[kHMMinimumContentIndex].contentType = // 8
            kHMCFStringContent;
        ioHelpContent->content[kHMMinimumContentIndex].u.tagCFString = // 9
            CFSTR("Turns help tags on or off.");
        ioHelpContent->content[kHMMaximumContentIndex].contentType = // 10
            kHMCFStringContent;
        ioHelpContent->content[kHMMaximumContentIndex].u.tagCFString = // 11
            CFSTR("Click to enable help tags. This is the maximum content.");
        *outContentProvided = kHMContentProvided; // 12
    }
    else if (inRequest == kHMDisposeContent) { // 13
    }
    return status;
}
```

Here's what the help tag callback does:

1. Tests for the help content request. If the Carbon Help Manager makes the `kHMSupplyContent` request, then the callback provides help tag content.
2. Sets the location for the help tag. The `tagSide` field of the help tag structure passed to the callback in the `ioHelpContent` parameter identifies the location, relative to the absolute hot rectangle, at which the help tag is displayed.
3. Saves the current port. The Control Manager function `GetControlOwner` returns a reference to the window containing the control; the Window Manager function `GetWindowPort` returns a pointer to the color graphics port of this window. The QuickDraw function `QDSwapPort` sets the current color graphics port (`CGrafPort`) drawing environment to the window's `CGrafPort` and returns a pointer to the previous `CGrafPort` in the `savePort` variable. If there was no previous `CGrafPort`, `savePort` is set to `NULL` so that it is not restored in step 7.
4. Sets the hot rectangle for the help tag to the current control bounds. The hot rectangle defines the area on screen over which the user must pause the pointer for the Carbon Help Manager to display the help tag. When the callback returns, the Carbon Help Manager verifies that the pointer is actually within the absolute hot rectangle before displaying the help tag.

In Mac OS X version 10.2 and later, you can provide an empty hot rectangle and the Carbon Help Manager will substitute the control's current bounds before drawing the help tag. If you supply an empty hot rectangle, steps 3 and 5 through 7 are not necessary.

5. Calls the QuickDraw function `LocalToGlobal` to convert the coordinates of the hot rectangle's top-left corner into global coordinates.
6. Calls the QuickDraw function `LocalToGlobal` to convert the coordinates of the hot rectangle's bottom-right corner into global coordinates.
7. If the saved port is valid, restores the current graphics port to its previous state, as recorded in the `savePort` variable in step 3.
8. Sets the format for the help tag's default, or minimum, content. The help tag content for the control is a `CFString` object, so the callback sets the `contentType` field to `kHMCStringContent`.
9. Sets the default help tag content. The `CFSTR` macro creates a constant `CFString` containing the help text.
10. Sets the format for the help tag's expanded, or maximum, content. Again, the content is a `CFString` object, represented by the `kHMCStringContent` constant.
11. Sets the expanded help tag content.
12. Sets the `outContentProvided` parameter to `kHMContentProvided`. This notifies the Carbon Help Manager that the callback has fulfilled the content request by supplying help tag content.
13. Handles the `kHMDisposeContent` request. Before removing the help tag from the screen, the Carbon Help Manager gives your application the opportunity to perform any necessary cleanup on help tag content provided by your callback, such as releasing memory. Your application would perform this cleanup when it receives the `kHMDisposeContent` request. In this case, the help tag content is constant and there is nothing to clean up.

If your help tag callback is unable to supply help tag content for the specified user interface element, your help tag callback should return `kHMContentNotProvided` or `kHMContentNotProvidedDontPropagate` in the `outContentProvided` parameter. When the Carbon Help Manager receives either of these constants, it ignores the values returned by your callback in all other parameters.

If your help tag callback returns the constant `kHMContentNotProvided`, the Carbon Help Manager next checks whether any help tag content is attached directly to the interface element. If content exists, the Carbon Help Manager displays it in a help tag. If no content is found and the interface element in question is a window, menu item, or menu title, the Carbon Help Manager assumes that no help content is available for that element. If, on the other hand, the object is a control, the Carbon Help Manager searches for help tag content further up the control embedding hierarchy. The Carbon Help Manager requests help content from the next control up the hierarchy, checking first for a help tag callback and then for content directly attached to the control. The Carbon Help Manager continues to propagate the request until it reaches the top of the embedding hierarchy, finds help tag content, or is instructed not to propagate the request. You instruct the Carbon Help Manager not to propagate the request for help tag content by returning the constant `kHMContentNotProvidedDontPropagate` in the `outContentProvided` parameter.

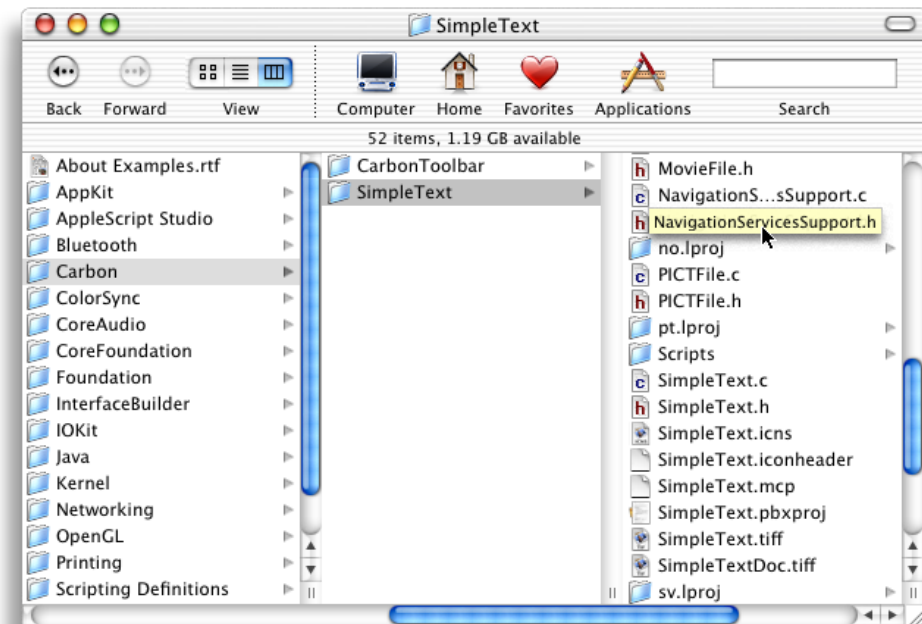
Disposing of Help Tag Content

The Carbon Help Manager automatically removes the help tag supplied by your help tag callback from the screen when user input occurs. Before it disposes of the help tag, the Carbon Help Manager calls your help tag callback with the `kHMDisposeContent` request in the `inRequest` parameter. When your help tag callback receives this request, it should release any memory allocated for the help tag in the `ioHelpContent` parameter. See [Listing 3-3](#) (page 25) for an example of a help tag callback for a control.

Displaying a Help Tag at an Application-Defined Location

The Carbon Help Manager also provides the `HMDisplayTag` and `HMHideTag` functions, which allow you to explicitly display and hide help tags. For example, you can use `HMDisplayTag` to display a help tag that is not associated with a control, window, menu item, or menu title. The Finder displays a help tag in this way when the user pauses with the pointer over a truncated filename. Figure 3-3 shows an example of the help tag used by the Finder to display the full filename.

Figure 2-3 A help tag explicitly displayed by the Finder



When you display a help tag using `HMDisplayTag`, you are responsible for tracking the mouse and determining when the user has moved the pointer over an object for which it is appropriate to display a help tag. You must initialize the help tag structure with the same information described in [“Attaching a Help Tag Directly to a User Interface Element”](#) (page 21). However, you must provide a hot rectangle defining the onscreen area with which the tag is associated; you cannot supply an empty rectangle in the `absHotRect` field of the help tag structure.

You are not required to call `HMHideTag` to hide a tag displayed with `HMDisplayTag`; the Carbon Help Manager automatically removes the help tag from the screen when user input occurs.

Creating a Help Tag for a Menu Title in Versions of Mac OS X Prior to 10.2

In versions of Mac OS X prior to Mac OS X 10.2, the Carbon Help Manager does not display menu title help tags correctly. If you set help tag content for a menu title by calling `HMSetMenuItemHelpContent` with a menu item index of zero and an empty rectangle for the hot rectangle, the Carbon Help Manager does not correctly translate the empty rectangle into the current bounds of the menu title, and no help tag is displayed. To provide menu title help tag content in Mac OS X 10.1.x and earlier, use `HMInstallMenuItemContentCallback` to register a help tag callback for the menu title.

Because the Carbon Help Manager does not correctly interpret an empty rectangle to mean the current location of the menu title, your callback must supply the correct hot rectangle for the menu title tag. However, there is no function for determining the current bounds of a menu title. As a workaround, you can get the current pointer position and initialize the `absHotRect` field with a rectangle surrounding the current pointer position.

Listing 2-4 A help tag callback for a menu title in Mac OS X 10.1.x and earlier

```

pascal OSStatus MyMenuItemContentCallback(
    MenuRef inMenu, HMContentRequest inRequest,
    HMContentProvidedType *outContentProvided,
    HMHelpContentPtr ioHelpContent)
{
    if (inRequest == kHMSupplyContent)
    {
        CFStringRef title;
        Point mouse;

        ioHelpContent->version = kMacHelpVersion;
        ioHelpContent->tagSide = kHMDefaultSide;
        GetGlobalMouse(&mouse);
        SetRect(&ioHelpContent->absHotRect, mouse.h - 5, mouse.v - 5, mouse.h
+ 5, mouse.v +5);
        CopyMenuItemAsCFString(inMenu, &title);
        ioHelpContent->content[kHMMinimumContentIndex].contentType =
kHMCFSStringContent;
        ioHelpContent->content[kHMMinimumContentIndex].u.tagCFString =
CFCopyLocalizedString(title, NULL);
        CFRelease(title);
        ioHelpContent->content[kHMMaximumContentIndex].contentType = kHMNoContent;
        *outContentProvided = kHMContentProvided;
    }
    else if (inRequest == kHMDisposeContent)
    {
        if (ioHelpContent->content[kHMMinimumContentIndex].contentType ==
kHMCFSStringContent)
            CFRelease(ioHelpContent-> content[kHMMinimumContentIndex].u.tagCFString);
        if (ioHelpContent->content[kHMMaximumContentIndex].contentType ==
kHMCFSStringContent)
            CFRelease(ioHelpContent-> content[kHMMaximumContentIndex].u.tagCFString);
    }
    return noErr;
}

```

Here is what the callback does:

1. Tests for the Carbon Help Manager request. To handle the `kHMSupplyContent` request, the callback must supply a help tag for the menu title.
2. Sets the structure version to the current version.
3. Sets the display location for the help tag to the default display location.
4. Calls the Event Manager function `GetGlobalMouse` to get the current location of the pointer in global screen coordinates. This location is used to construct the hot rectangle for the menu title help tag.
5. Calls the QuickDraw function `SetRect` to create the hot rectangle for the help tag. Because there is no function for retrieving the current bounds of a menu title, the callback creates a rectangle around the pointer's current position.
6. Calls the Menu Manager function `CopyMenuItemAsCFString` to retrieve the menu title.

7. Sets the format type for the help tag's default content. In this case, the help tag content is a `CFString` object.
8. Sets the default content for the help tag. The Core Foundation Bundle Services function `CFCopyLocalizedString` retrieves the help string from the `Localizable.strings` file, using the menu title as the key.
9. Calls the Core Foundation Base Services function `CFRelease` to release the memory associated with the `CFString` object containing the menu title.
10. Sets the expanded content for the help tag. In this case, there is no expanded content.
11. Notifies the Carbon Help Manager that help tag content has been supplied.
12. Handles the Carbon Help Manager's request to dispose of help tag content.
13. Tests for `CFString` content in the help tag structure. If any exists, the callback calls `CFRelease` to release the memory associated with the content.

Document Revision History

This table describes the changes to *Providing Help Tags in Carbon*.

Date	Notes
2002-06-01	Changed text in the sections “ The Hot Rectangle ” (page 10) and “ Supplying Help Tag Content With Callbacks ” (page 25) to reflect the Carbon Help Manager change in Mac OS X version 10.2 adding support for an empty hot rectangle from a help tag callback.
	Noted addition of support for <code>TEHandle</code> and <code>'TEXT'</code> resource help tag content in Mac OS X version 10.2 in the section “ Help Tag Content ” (page 14).
	Added text in “ Help Tag Content ” (page 14) describing localized <code>CFString</code> content for help tags.
	Changed code in Listing 3-1 (page 22) to use localized <code>CFString</code> content.
	Correction: “ <code>Localizeable.strings</code> ” should be “ <code>Localizable.strings</code> ”
2002-05-01	First public release.

REVISION HISTORY

Document Revision History