# NSHelpManager

| | |
|---|---|
| **Inherits From:** | NSObject |
| **Conforms To:** | NSObject (NSObject) |
| **Declared In:** | AppKit/NSHelpManager.h |

## Class Description

NSHelpManager provides a platform-independent approach to displaying on-line help. An application contains one instance of NSHelpManager. Your application's code rarely needs to access NSHelpManager directly. Instead, you use Interface Builder and Project Builder to set up on-line help for your application.

OpenStep applications can run on multiple platforms, and each platform provides its own support for on-line help. It's important to users that applications use the native on-line help system (on Microsoft Windows, for instance, users want the Microsoft Windows help system and don't want to have to learn how to use a different help system), so NSHelpManager does not provide a comprehensive solution for presenting help. Instead, it provides cross-platform support for context-sensitive help and allows you to present more comprehensive help (conceptual and task-based help) in any way you choose.

### Context Help

Context-sensitive help (also referred to as context help) gives the user a small amount of information when they help-click an interface item. For example, if the user help-clicks a menu item called "Copy," they should get context help that says something like "Copies the currently selected text to the pasteboard." This text appears in a small window near where the user help-clicked, and the window disappears when the user clicks anywhere else in the application.

Help-clicking is performed in any one of several ways, depending on the platform and the hardware used. On Mach platforms, help-clicking is performed when the user holds down the Help key, the F1 key, or the Alternate and Control keys while pressing the mouse button. On Microsoft Windows platforms, users press Shift-F1 and then press the mouse button to display context-sensitive help. Some Microsoft Windows applications also have a What's This menu item on the Help menu. When the user selects this item, the next mouse click displays context-sensitive help.

To provide context-sensitive help for your application, follow these steps:

1. For each interface item that needs context help, create an RTF or RTFD file containing the text and any images you want to appear when the user help-clicks that interface item. Try to keep the text as brief as possible and the images as small as possible.

The text that you write will appear in a small window just under the cursor when the user help-clicks an interface item. If the user help-clicks near the edge of the screen, text may appear off-screen. (This is especially prevalent when the user help-clicks a menu item on the Mach platform.) Use hard returns in your text so that the window will be as narrow as possible.

2. If you don't need to localize your context help files, in Project Builder simply add these files to your project under Context Help.

   If you do need to localize your context help files, first copy the files into the appropriate **.lproj** directory of your project, then add them to the project.

In Interface Builder, connect each interface item to its context help file by doing the following:

1. Bring up the Interface Builder inspector and choose the Help display. The Help display lists all the context help files associated with your application. (You may have to quit and restart Interface Builder to get this to occur.)
2. Select an interface item.
3. In the Inspector, choose the appropriate help file.

When you build your application, **/usr/bin/compileHelp** packages your help files into a property list named **Help.plist**. NSHelpManager knows how to extract context help from an **Help.plist** file.

## Comprehensive Help

Most applications provide some form of on-line help that is more comprehensive and detailed than context-sensitive help, such as conceptual or task help. NSHelpManager allows you to provide this sort of comprehensive help in any way you choose. Some help authors prefer to provide comprehensive help in HTML using a World-Wide Web browser; others use tools such as Digital Librarian or Concurrence; on Microsoft Windows a full-featured native help system is available.

When the user chooses the Help menu item, NSHelpManager simply asks NSWorkspace to open the help file you have specified for your application. That file should be the starting point of your help, and should allow users to access whatever information they might need.

To specify a help file for your application, do one of the following:

- In Project Builder, specify the name of the help file in the Project Attributes inspector. (If you are creating an application that will run on both the Mach and Windows platforms, you need to enter this file twice—once for Mach and once for Windows). The specified value can be a full or relative path, and if it is relative, it is assumed to be a resource in the application wrapper.
- As an alternative, you can place the help file in your application wrapper and name it after your application. If you haven't specified a help file, NSHelpManager looks in the application wrapper for an appropriately named file.

On Mach, it must be an RTF file called *appName***.rtf** (where *appName* is the name of the application).

On Microsoft Windows, it must be a Windows help file called *appName***.hlp**.

**Note:** It’s common for Windows applications to have more than one command under the Help menu and to have each command open a different help file. To implement this, connect each of the Help menu commands to a different action method. The actions methods should send **openFile:** to the shared NSWorkspace object to open the appropriate help file. For example:

```
[[NSWorkspace sharedWorkspace] openFile:@"AppKit.hlp"];
```

## Method Types

| | |
|---|---|
| Creating an NSHelpManager instance | + sharedHelpManager |
| Getting and setting context help mode | + setContextHelpModeActive:<br>+ isContextHelpModeActive |
| Returning context-sensitive help | – contextHelpForObject:<br>– showContextHelpForObject:locationHint: |
| Setting up context-sensitive help | – setContextHelp:forObject:<br>– removeContextHelp:forObject: |

## Class Methods

### isContextHelpModeActive

+ (BOOL)**isContextHelpModeActive**

Returns YES if the application is currently in context-sensitive help mode, NO otherwise. In context-sensitive help mode, when a user clicks a user interface item, help for that item is displayed in a small window just below the cursor.

**See also:** + **setContextHelpModeActive:**

### setContextHelpModeActive:

+ (void)**setContextHelpModeActive:**(BOOL)*flag*

Controls context-sensitive help mode. If *flag* is YES, the application enters context-sensitive help mode. If *flag* is NO, the application returns to normal operation.

You never send this message directly; instead, the NSApplication method **activateContextHelpMode** activates context-sensitive help mode, and the first mouse click after displaying the context-sensitive help window deactivates it.

When the application enters context-sensitive help mode, NSHelpManager posts NSContextHelpModeDidActivateNotification to the default notification center. When the application returns to normal operation, NSHelpManager posts NSContextHelpModeDidDeactivateNotification.

**See also:** + **isContextHelpModeActive**

### sharedHelpManager

+ (NSHelpManager *)**sharedHelpManager**

Returns the shared NSHelpManager instance, creating it if it does not already exist.

## Instance Methods

### contextHelpForObject:

– (NSAttributedString *)**contextHelpForObject:**(id)*object*

Returns context-sensitive help for *object*.

**See also:** – **setContextHelp:forObject:**, – **showContextHelpForObject:locationHint:**

### removeContextHelpForObject:

– (void)**removeContextHelpForObject:**(id)*object*

Removes the association between *object* and its context-sensitive help. If object does not have context-sensitive help associated with it, this method does nothing. Typically, you use Interface Builder to remove context-sensitive help from an item.

**See also:** – **setContextHelp:forObject:**

### setContextHelp:forObject:

– (void)**setContextHelp:**(NSAttributedString *)*help* **forObject:**(id)*object*

Associates *help* with *object*. When the application enters context-sensitive help mode, if *object* is clicked, *help* will appear in the context-sensitive help window. Typically, you use Interface Builder to associate context-sensitive help with an object.

**See also:** – **removeContextHelpForObject**:

### showContextHelpForObject:locationHint:

– (BOOL)**showContextHelpForObject:**(id)*object* **locationHint:**(NSPoint)*point*

Displays the context-sensitive help for *object* at or near the point on the screen specified by *point*. This point is usually just under the cursor. Returns YES if it successfully displays context-sensitive help for the object, NO if it cannot (for example, if there is no context-sensitive help associated with this object).

**See also:** – **contextHelpForObject:**

## Notifications

### NSContextHelpModeDidActivateNotification

Posted when the application enters context-sensitive help mode. This typically happens when the user holds down the Help key. It can also occur on Microsoft Windows platforms if the user chooses the What's This command from the Help menu.

The notification contains:

| | |
|---|---|
| **Notification Object** | The NSHelpManager object |
| Userinfo | None |

### NSContextHelpModeDidDeactivateNotification

Posted when the application exits context-sensitive help mode. This happens when the user clicks the mouse anywhere on the screen after displaying a context-sensitive help topic.

The notification contains:

| | |
|---|---|
| **Notification Object** | The NSHelpManager object |
| Userinfo | None |