Dock Tile Programming Guide

Cocoa > User Experience

2009-03-04

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

Simultaneously published in the United States and Canada.

Contents

Figures and Listings

Introduction

Note: This document was previously titled *Customizing Your Application Dock Tile*.

The Dock is a desktop application designed to reduce desktop clutter, provide users with feedback about an application, and allow users to switch easily between tasks. With Mac OS X version 10.1 and later, you can customize your application Dock tile by modifying the Dock icon and adding items to the contextual menu displayed for your application. With Mac OS X version 10.5 and later, Cocoa applications can customize the Dock icon of a minimized window.

This document provides an overview of the Dock and describes how you can customize the contextual menu associated with your application Dock tile. It does not describe how to create icons. You should consult *Apple Human Interface Guidelines* for tips on designing Dock icons, naming Dock menu items, and using badges in your Dock tile.

Organization of This Document

This document is organized into the following sections:

- "Dock Tile Concepts" (page 9) describes the purpose and use of the Dock, the icon used by an application Dock tile, and the contextual menu that is available for a Dock tile.
- "Dock Tile Tasks for Cocoa Applications" (page 13) describes how to customize Dock icons and the application Dock tile contextual menu inside a Cocoa application.
- "Dock Tile Tasks for Carbon Applications" (page 17) describes how to customize Dock icons and the application Dock tile contextual menu inside a Carbon application.

See Also

These references are relevant to customizing a Dock tile on Cocoa:

- *NSApplication Class Reference*
- *NSDockTile Class Reference*

These references are relevant to customizing a Dock tile on Carbon:

- *Application Manager Reference*
- *Tiler* is a sample application that shows how to set Dock tiles and badges.

For information on creating icons, see:

- *Obtaining and Using Icons With Icon Services*
- *Icon Services and Utilities Reference*
- *Apple Human Interface Guidelines*

Dock Tile Concepts

The Dock was designed to make the Macintosh easier to use:

- The Doc organizes your applications and documents.
- The Dock allows users to switch between tasks simply by clicking the application or window icon in the Dock.
- Items in the Dock provide users with useful feedback about what each item represents. For example, images in the Dock are shown in preview mode, so it's apparent what the image is without opening it.

A **Dock tile** is the area that contains the Dock icon and provides users access to the Dock item's name and contextual menu. When a user rolls the cursor over a Dock tile, the item's name appears. When the user presses and holds the mouse button while the cursor is on a Dock tile, a contextual menu appears. The following sections describe in more detail the Dock icon and contextual menu.

Dock Icon

An application's Dock Icon is, by default, its application icon. It is possible for you to modify or replace the default icon with another image that indicates the current state of your application. Figure 1-1 shows an example of an application, Mail, that does just that. The icon for Mail changes when messages are waiting to be read. A badge—the red circle and number in the figure—is overlaid onto Mail's application icon to indicate the number of unread messages. The number changes each time Mail retrieves more messages.

Figure 1-1 The red badge in the Mail's Dock tile indicates the number of waiting messages

A window that is minimized into the Dock also has a Dock tile icon. By default, this is a miniaturized version of the window's content that is badged with the application's Dock Icon.

Figure 1-2 A Mail window minimized into the Dock.

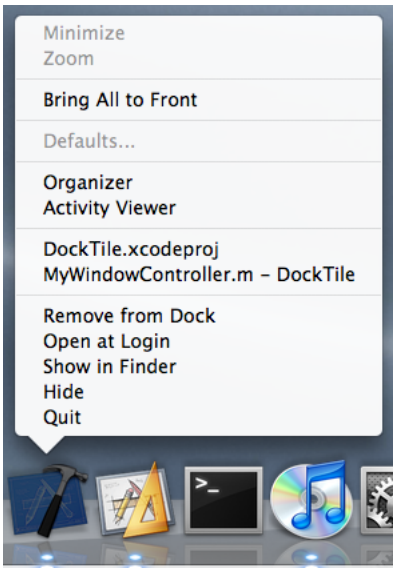
Dock Menu

When the user presses and holds the mouse button while the cursor is over a Dock tile, a contextual menu appears. If your application does nothing to customize the contextual menu, the application Dock tile's contextual menu contains a list of the application's open documents (if any), followed by the standard menu items Keep in Dock, Open at Login, Show in Finder, Hide, and Quit.

Note: When your application is not running, a minimal menu will be displayed.

Figure 1-3 shows the application Dock contextual menu for Xcode. Xcode provides a number of custom menu items.

Figure 1-3 XCode's application Dock contextual menu



Custom items that an application adds to its Dock menu appear between the list of open documents and the standard items, as shown in Figure 1-4. The sample application has one window, named Window, open. The menu items Play, Pause, Stop, Next Song, Previous Song, and the Submenu (with its items) are custom items added by the application.

Figure 1-4 Custom items in an application's Dock menu

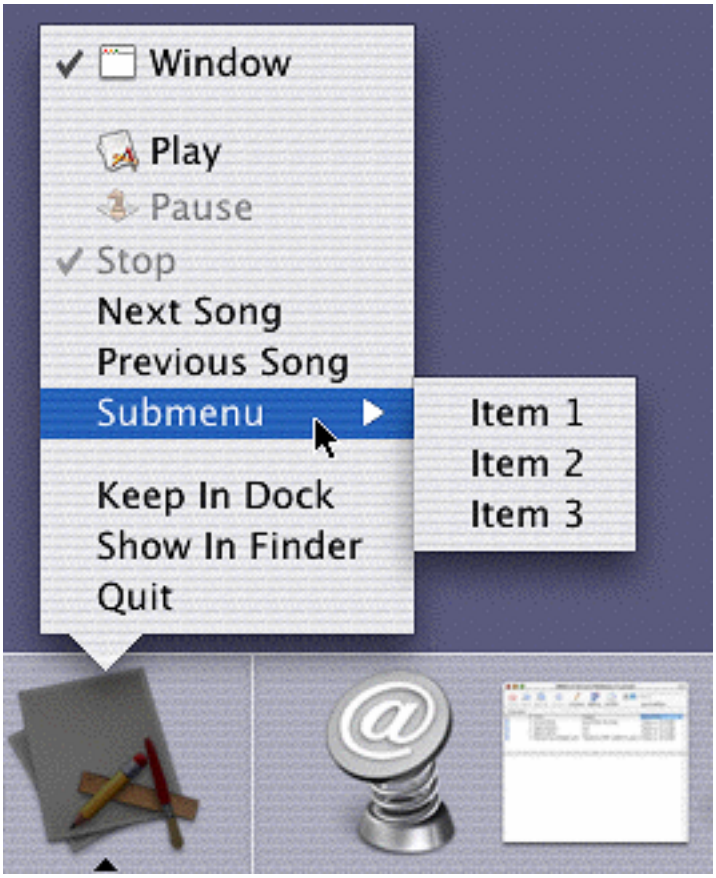


Figure 1-4 illustrates that you can further customize menu items—add an icon to an item. A window icon appears next to the window name, the application's icon appears to the left of the item Play, and a custom icon appears next to Pause. It's best to use icons that convey meaning to the user. For example, it would be preferable for the sample application to use a standard icon that indicates play for the Play item and pause for the Pause item. See *Apple Human Interface Guidelines* for additional information on designing and using a menu for the Dock.

Dock Tile Tasks for Cocoa Applications

Cocoa applications can customize both the application's Dock Icon and a minimized window's Dock icon.

- The simplest way to customize the application's Dock icon is to provide a new image to replace the default application icon. See "Customizing the Application's Dock Icon" (page 13).
- To change a window's Dock icon, or to dynamically change the application's Dock Icon, you can draw a Dock icon using a custom view. See "Using a Custom View to Draw a Dock Icon" (page 13).
- To add text to a Dock icon, you can apply a badge label. See "Changing the Text of a Badge Label" (page 14).
- To remove the application badge from a window's Dock icon, see "Hiding the Application Icon Badge on a Window's Dock Tile Icon" (page 14).

Your application can also customize the contextual menu for your application Dock tile.

- To add static menu items to the contextual menu, you provide a menu in a nib file and reference this nib file inside your application's Information Property List. See "Adding Static Menu Items With a Nib File" (page 14).
- To dynamically change the menu when the user clicks in the Dock, provide an `applicationDockMenu:` function in your application's delegate. See "Dynamically Adding Contextual Menu Items With the Application Delegate" (page 15).

Customizing the Application's Dock Icon

While your application is running, you can call the `setApplicationIconImage:` method of the NSApplication object to directly change the application Dock tile icon.

```
myImage = [NSImage imageNamed: @"ChangedIcon"];
[NSApp setApplicationIconImage: myImage];
```

To restore your application's original icon, you call `setApplicationIconImage:` with a nil parameter:

```
[NSApp setApplicationIconImage: nil];
```

Using a Custom View to Draw a Dock Icon

Dock tile icons can be customized using an `NSView` object. This is useful if your application needs to dynamically generate Dock tile icons at run time. To provide a custom view, you instantiate a new view object, retrieve the dock tile object from the application or window object, and set your view as its `contentView`.

```
myView = [[[MyViewClass alloc] init] autorelease];
[[NSApp dockTile] setContentView: myView];
```

When the Dock icon needs to be updated, you instruct the Dock to update the icon by calling the dock tile object's `display` method.

```
[[NSApp dockTile] display];
```

Note: When a window's dock tile object has a custom view, no application badge is provided. You are responsible for all content except for the badge label.

Changing the Text of a Badge Label

The dock tile object can overlay a short text message on top of the Dock icon. To change the badge label, you call the Dock tile's `setBadgeLabel:` method.

```
[[myWindow dockTile] setBadgeLabel:@"42"];
```

Note: A window's Dock tile may only include a badge label if has a custom content view.

Hiding the Application Icon Badge on a Window's Dock Tile Icon

By default, a window's Dock icon consists of a miniaturized image of the window's contents with a badge of the application's Dock Icon layered on top of it. This includes any customized icon you may have provided for the application's Dock icon. You can optionally turn off the application badge by calling the `setShowsApplicationBadge:` method.

```
[[myWindow dockTile] setShowsApplicationBadge: NO];
```

The application's Dock Tile icon does not show an application badge, and ignores attempts to show one.

Note: If the window's dock tile object has a custom view, the application badge is not provided, and the dock tile will ignore this method.

Adding Static Menu Items With a Nib File

If your application needs to add static items to the application's Dock tile's contextual menu, you can provide those items in a nib file. To do this, perform the following steps.

1. Launch Interface Builder.
2. Create a new nib file for your menu.
3. Create a menu that includes the items you wish to add to the contextual menu.

4. Connect the `dockMenu` outlet of the file's owner (which by default is `NSApplication`) to your menu.
5. Add the nib name to the `Info.plist`, using the key `AppleDockMenu`. The nib name is specified without an extension.

Dynamically Adding Contextual Menu Items With the Application Delegate

An application can also provide items dynamically to your application's Dock tile's contextual menu. To do this, your application's delegate object provides a `applicationDockMenu:` method. This method returns a NSMenu object that provides all the custom menu items you wish to add to the menu. If you also provided a menu using a nib file (see "Adding Static Menu Items With a Nib File" (page 14)), any menu returned by your delegate replaces the menu provided in the nib file.

Dock Tile Tasks for Carbon Applications

There are two things you can do to customize your application's Dock tile—modify the Dock icon or add custom items to the Dock menu. If you want to modify your application's Dock icon see the *Tiler* sample application.

You should also see the *Application Manager Reference* for a description of the functions you can use to modify and restore your application's Dock icon.

This chapter shows you three ways you can add custom menu items to the contextual menu for your application's Dock tile.

- Add a property to the application's information property list. If your application is a bundle and only needs to add static menu items, then the easiest way to add custom items to the contextual menu is to add the property `AppleDockMenu` to the application's `Info.plist` file. See "Adding a Property to the Information Property List" (page 17).
- Use the functions `SetApplicationDockTileMenu` and `GetApplicationDockTileMenu`. If your application is not a bundle or if it needs to create the menu at runtime, then you can use these functions to set and retrieve the items in the menu. See "Using the Application Dock Tile Menu Functions" (page 20).
- Handle the Dock tile menu event. If your application needs to create the contextual menu at the time the user clicks the application's icon in the Dock, you can set up a handler to handle the associated Carbon event—event class `kEventClassApplication` and event kind `kEventAppGetDockTileMenu`. See "Handling the Get Dock Tile Menu Event" (page 21).

Regardless of the method you use to customize your application Dock menu, each custom menu item must have a command ID, or the application won't be notified by the Carbon Event Manager when that item is selected.

If you want to add an icon to an item in your application Dock menu, see "Adding an Icon to a Dock Menu Item" (page 21).

Adding a Property to the Information Property List

You can customize the contextual menu for your application Dock tile by doing the following:

1. Create a contextual menu in Interface Builder.
2. Add a property to your application's information property list to specify the name of the Interface Builder file that contains your contextual menu.

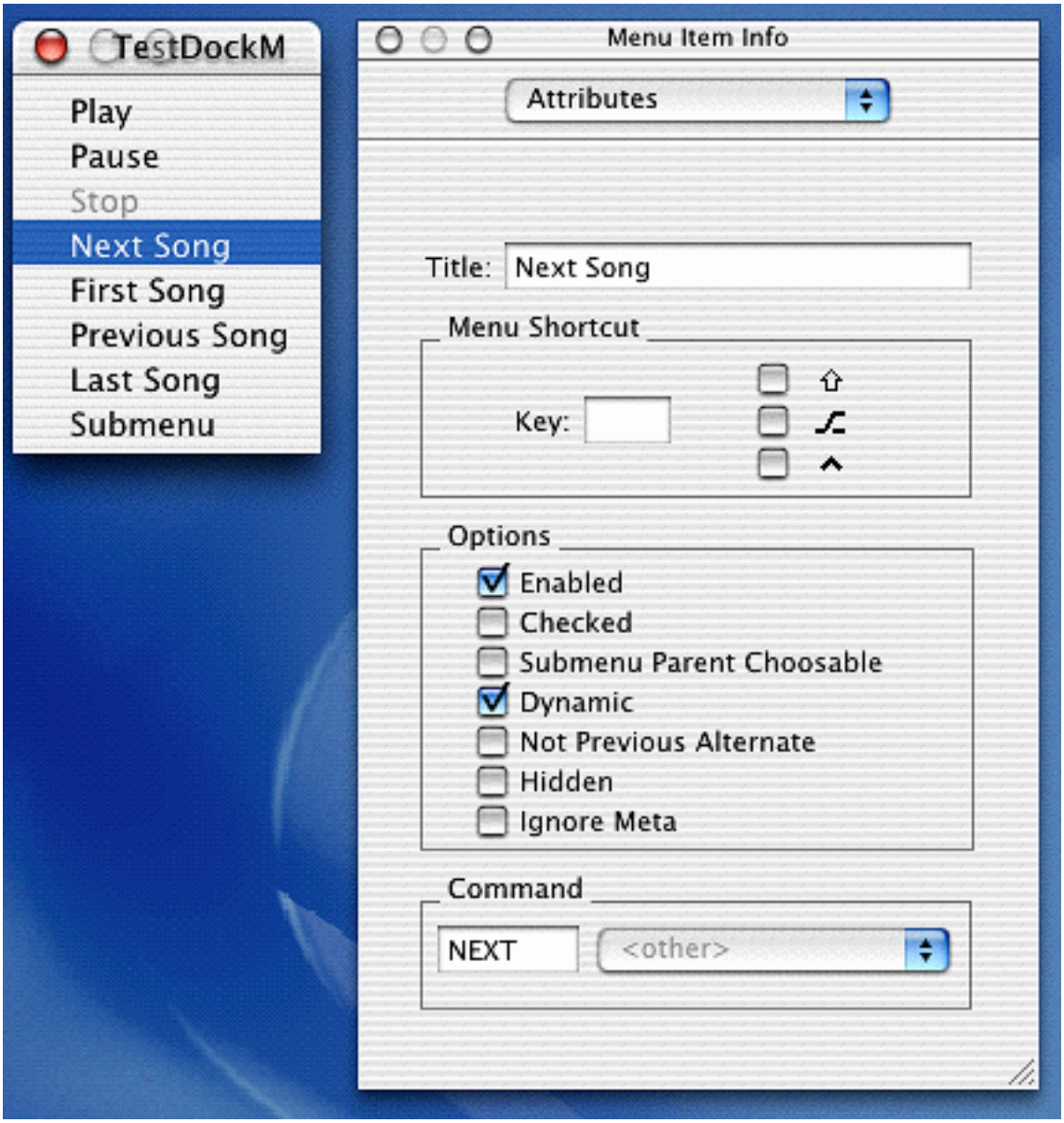
Once you do this, Carbon automatically loads the nib file, creates the menu, and provides it to the Dock. Each step is detailed in the following sections.

Creating a Contextual Menu in Interface Builder

You should open Interface Builder from your application's Project Builder file, then create a new nib file just for your contextual menu. (See *Learning Carbon* for information on using Interface Builder.) The contextual menu needs to contain only those items you want to add to your application's Dock menu; standard items (see "Dock Menu" (page 10)) are added automatically. "Setting the command for a custom item in a contextual menu" shows custom items for a contextual menu that's being built in Interface Builder.

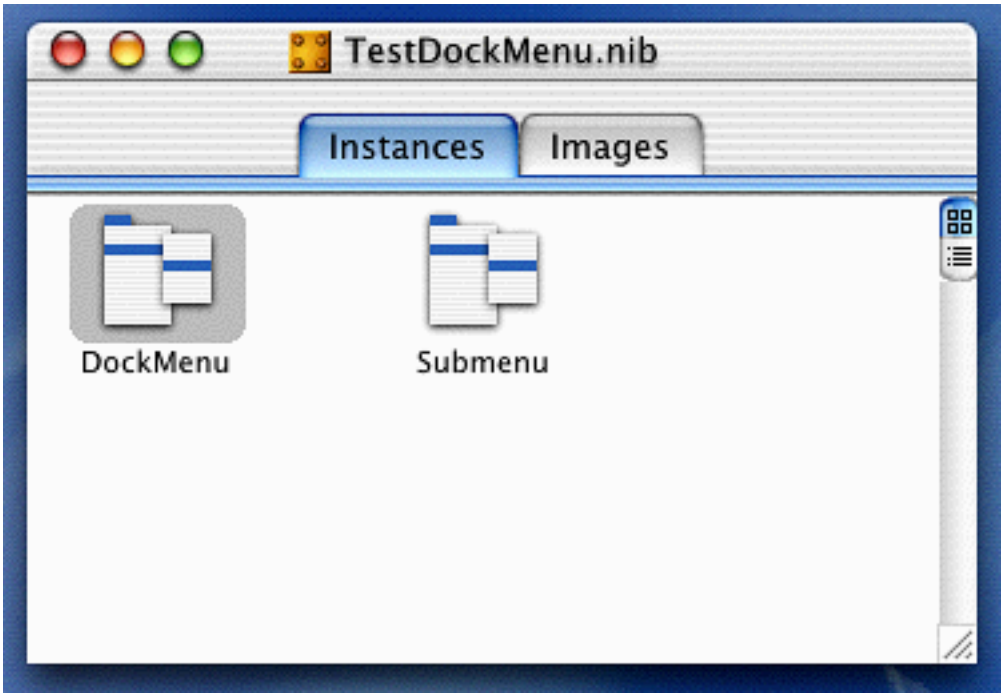
Each menu item needs to have an associated four-character command that's unique to your application. The command is what the Carbon Event Manager provides to your application to let you know which item was chosen by the user. The Next Song menu item in "Setting the command for a custom item in a contextual menu" has the command `NEXT`.

Figure 3-1 Setting the command for a custom item in a contextual menu



"Menu objects for a Dock menu" shows an Interface Builder Instances pane for the `TestDockMenu.nib` file. This file contains two items, a contextual menu for the Dock, named `DockMenu`, and a submenu for the contextual menu, named `Submenu`.

Figure 3-2 Menu objects for a Dock menu



Adding a Property for the Dock Menu

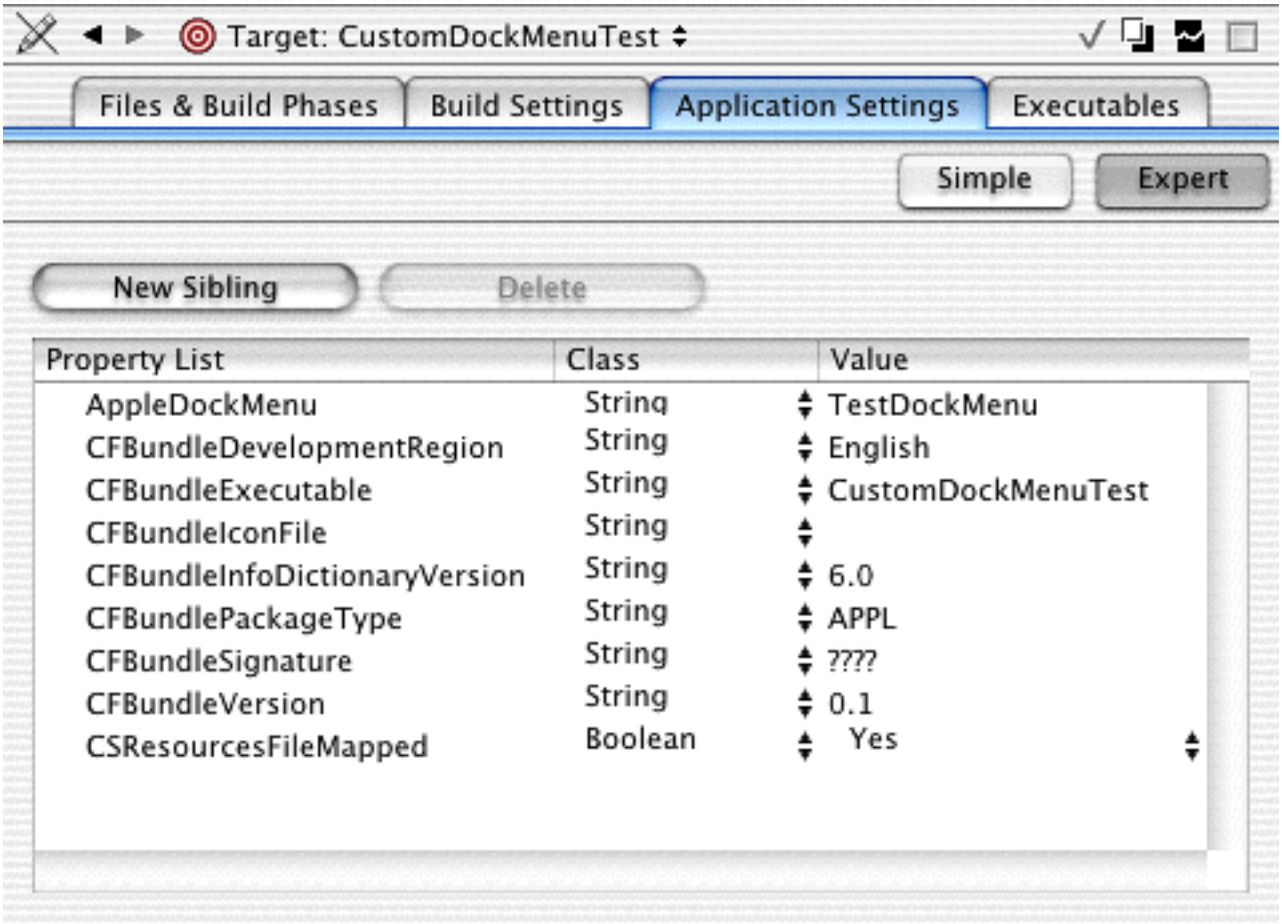
Follow these steps to add the `AppleDockMenu` property to the information property list for your application.

1. Open your application's project in Project Builder.
2. Click the Targets tab, then click the appropriate target in the Targets list.
3. Click the Application Settings tab, then click Expert.
4. Click the New Sibling button.
5. Type `AppleDockMenu` in the Property List column.
6. Make sure the class is set to string.
7. In the Value column, type the name of the Interface Builder nib file that contains the Dock menu object, but do not include the `.nib` extension.

   "The AppleDockMenu property" show an information property list as it appears in Project Builder. Note the value of the `AppleDockMenu` property is the string `TestDockMenu`. This implies the name of the nib file that contains the Dock menu items is `TestDockMenu.nib`. You must make sure the value you add matches the name of the nib file you create for your Dock menu.

8. Build and run the application.

Figure 3-3 The AppleDockMenu property



Using the Application Dock Tile Menu Functions

If your application is not a bundle or if it needs to create the application's contextual menu at runtime, then you can use the functions `SetApplicationDockTileMenu` and `GetApplicationDockTileMenu` to set and retrieve the items in the menu.

You can use any method (for example Interface Builder or Menu Manager functions) to define and create a menu that contains your custom items. Regardless of how you create the menu, each menu item needs to have an associated four-character command that's unique to your application.

Once the menu is created, you pass a reference to your menu to the function `SetApplicationDockTileMenu`. The items in the menu you pass to the function are inserted in the list, between the list of document windows and the standard items.

The function `SetApplicationDockTileMenu` increments the reference count of the menu you pass to it. Before the contextual menu is displayed, it receives the Carbon events `kEventMenuPopulate`, `kEventMenuOpening`, and `kEventMenuEnableItems`, so any event handlers for these events can update the menu appropriately.

Handling the Get Dock Tile Menu Event

If your application needs to create the contextual menu at the time the user clicks the application's icon in the Dock, you can set up a handler to handle the Carbon event that is of event class `kEventClassApplication` and event kind `kEventAppGetDockTileMenu`. The default handler for this event returns the menu provided by `SetApplicationDockTileMenu` or the menu located in the nib file for the Dock menu, if it exists.

You can override the default handler by writing and installing your own Carbon event handler. Your application needs to create the custom menu and your handler needs to call the Carbon Event Manager function `SetEventParameter` with the Carbon event parameter `kEventParamMenuRef` and a reference to the menu you want to be displayed, as shown in "Passing a menu reference to the function SetEventParameter."

Listing 3-1 Passing a menu reference to the function SetEventParameter

```
err = SetEventParameter (inEvent,
                kEventParamMenuRef,
                typeMenuRef,
                sizeof(menuRef),
                &myCustomApplicationDockMenu
                );
```

Carbon releases the menu reference after the menu is passed to the Dock, so be sure to retain the menu reference with the Menu Manager function `RetainMenu` if you want to keep the menu reference for later use.

Adding an Icon to a Dock Menu Item

If you want to display an icon next to an item in your application Dock menu (as shown in Figure 1-4 (page 11)) you need to call the function `SetMenuItemIconHandle` with one of the new selectors provided by the Menu Manager—`kMenuSystemIconSelectorType` or `kMenuIconResourceType`. If your application uses a standard system icon provided by Icon Services, you should use the selector `kMenuSystemIconSelectorType`, as shown in "Using a standard system icon in the application Dock Menu." To load the icon, you need to call the Icon Services function `GetIconRef`.

Listing 3-2 Using a standard system icon in the application Dock Menu

```
SetMenuItemIconHandle (myMenu,
                       1,
                       kMenuSystemIconSelectorType,
                       (Handle) kGenericApplicationIcon);
```

If you application uses a `.icns` file located in the application bundle, you should use the selector `kMenuIconResourceType` with the function `SetMenuItemIconHandle`, as shown in "Using an icon specified in a .icns file." The `inIconHandle` parameter to this function must be a `CFStringRef` that specifies the name of the `.icns` file. The Menu Manager locates the file and registers the `IconRef` for the icon specified in the file, and then displays the icon with the menu item. The `CFStringRef` is retained by the Menu Manager.

Listing 3-3 Using an icon specified in a .icns file

```
SetMenuItemIconHandle (myMenu,
                       2,
                       kMenuIconResourceType,
                       (Handle) CFSTR("mySpecialIcon.icns") );
```

Document Revision History

This table describes the changes to *Dock Tile Programming Guide*.

Date	Notes
2009-03-04	Updated to include guidelines for Cocoa programmers.
2007-07-10	Fixed figure numbering problem.
2006-09-05	Provided references to icon documentation and changed title from "Customizing Your Application Dock Tile."
2002-10-02	Fixed formatting and typographical errors; removed information about background applications. Background applications cannot have a Dock icon.
2001-08-29	First release of this document.

Document Revision History