Application Manager Reference

Carbon > User Experience

2007-10-31

Contents

Application Manager Reference

Framework: Carbon/Carbon.h

Declared in MacApplication.h

Overview

Note: This document was previously titled *Dock Manager Reference*.

The Application Manager provides a set of functions that Mac OS X applications can use to perform various application-level tasks. For example, you can use the Application Manager to:

- Control the display of system-provided user interface elements such as the menu bar and Dock while your application is in the foreground
- Customize your application's Dock tile by modifying the Dock icon and adding items to the contextual menu displayed for your application
- Display a Spotlight search window
- Display a custom about box for your application
- Retrieve the current application object (HIObject)

Functions by Task

Drawing in the Application Dock Tile

`HIApplicationCreateDockTileContext` (page 12)
  Returns a Quartz graphics context for drawing in the application Dock tile.

`BeginCGContextForApplicationDockTile` (page 7)
  Returns a Quartz graphics context for drawing in the application Dock tile.

`EndCGContextForApplicationDockTile` (page 8)
  Releases the Quartz graphics context for an application Dock tile.

`BeginQDContextForApplicationDockTile` (page 23) Deprecated in Mac OS X v10.5
  Returns a QuickDraw graphics port for drawing in the application Dock tile. (Deprecated. Use `BeginCGContextForApplicationDockTile` (page 7) or `HIApplicationCreateDockTileContext` (page 12) instead.)

`EndQDContextForApplicationDockTile` (page 23) Deprecated in Mac OS X v10.5
: Releases the QuickDraw graphics port for an application Dock tile. (Deprecated. Use `EndCGContextForApplicationDockTile` (page 8) instead.)

Working With the Dock Menu

`GetApplicationDockTileMenu` (page 9)
: Returns the menu containing items added to the contextual menu for your application Dock tile.

`SetApplicationDockTileMenu` (page 16)
: Adds items to the contextual menu for your application Dock tile.

Working With the Dock Icon

`SetApplicationDockTileImage` (page 15)
: Replaces an application Dock icon.

`OverlayApplicationDockTileImage` (page 14)
: Composites an image with your application's Dock icon.

`RestoreApplicationDockTileImage` (page 15)
: Restores your application Dock icon to the application icon.

`CreateCGImageFromPixMaps` (page 7)
: Creates a Quartz image from an image and a mask.

Getting Scripts and Encodings

`GetApplicationScript` (page 9)
: Returns the application script.

`GetApplicationTextEncoding` (page 10)
: Returns the application text encoding for Resource Manager resources.

Displaying an About Box

`HIAboutBox` (page 11)
: Displays a generic, HI-compliant about box.

Controlling System-Provided User Interface Elements

`SetSystemUIMode` (page 17)
: Sets the presentation mode of the calling application.

`GetSystemUIMode` (page 10)
: Gets the presentation mode of the calling application.

`HISearchWindowShow` (page 14)
: Displays a Spotlight search window.

Getting the Application Object

`HIApplicationGetCurrent` (page 13)
Returns the currently running Carbon application object.

Getting the Focused Window

`HIApplicationGetFocus` (page 13)
Returns either the modeless or effective focused window.

Functions

BeginCGContextForApplicationDockTile

Returns a Quartz graphics context for drawing in the application Dock tile.

```
CGContextRef BeginCGContextForApplicationDockTile (
   void
);
```

Return Value
A graphics context you can use to draw in the application Dock tile with Quartz 2D.

Discussion
This function makes it possible to draw into the application Dock tile at a resolution of 128x128, which is the size of all Dock tiles prior to Mac OS X v10.5. If the user interface scale factor is not 1.0, the drawing will be scaled to the actual size of the tile.

This function locks the application Dock tile to prevent the Dock from drawing in the tile. When you are finished using the context, you must call the function `EndCGContextForApplicationDockTile` to release the context and the lock. Do not use `CGEndContext` or `CFRelease` for this purpose. To ensure that drawing to the context appears onscreen, you should call `CGContextFlush` before releasing the context.

Availability
Available in Mac OS X v10.0 and later.
Not available to 64-bit applications.

See Also
`EndCGContextForApplicationDockTile` (page 8)

Declared In
`MacApplication.h`

CreateCGImageFromPixMaps

Creates a Quartz image from an image and a mask.

```
OSStatus CreateCGImageFromPixMaps (
   PixMapHandle inImage,
   PixMapHandle inMask,
   CGImageRef *outImage
);
```

Parameters

inImage

A handle to the image you want to use to create the Quartz image. The image should be the same size as the mask. For use in the Dock, the image should be 128 pixels square. Otherwise, the image can be any size.

inMask

A handle to the mask to use as the alpha channel. The mask should be the same size as the image.

outImage

On return, a Quartz image.

Return Value

A result code.

Discussion

The function `CreateCGImageFromPixMaps` uses the mask as the alpha channel for the resulting image. This allows you to have any level of transparency in the resulting image. You can pass the Quartz image as a parameter to any Quartz 2D drawing function, as well as to Dock tile functions such as the functions `SetApplicationDockTileImage` and `OverlayApplicationDockTileImage`. You can use `CreateCGImageFromPixMaps` to create an image for a badge, and then apply the badge to your application Dock icon.

Availability

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

Declared In

`MacApplication.h`

EndCGContextForApplicationDockTile

Releases the Quartz graphics context for an application Dock tile.

```
void EndCGContextForApplicationDockTile (
   CGContextRef inContext
);
```

Parameters

inContext

A Quartz graphics context created by calling `BeginCGContextForApplicationDockTile` or `HIApplicationCreateDockTileContext`. On output, the context is invalid and should no longer be used.

Discussion

This function also releases the lock on the application Dock tile, signaling the Dock that you are done drawing in the tile.

Availability

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

See Also

`HIApplicationCreateDockTileContext` (page 12)

`BeginCGContextForApplicationDockTile` (page 7)

Declared In

`MacApplication.h`

GetApplicationDockTileMenu

Returns the menu containing items added to the contextual menu for your application Dock tile.

```
MenuRef GetApplicationDockTileMenu (
   void
);
```

Return Value

The menu containing items added to your application Dock tile menu using the function `SetApplicationDockTileMenu` (page 16), or `NULL` if there are no additional menu items.

Availability

Available in Mac OS X v10.1 and later.

Not available to 64-bit applications.

See Also

`SetApplicationDockTileMenu` (page 16)

Declared In

`MacApplication.h`

GetApplicationScript

Returns the application script.

```
ScriptCode GetApplicationScript (
   void
);
```

Return Value

The application script.

Discussion

Your application needs to get the application script when it uses a function, such as `UseThemeFont`, that takes a script code as a parameter.

Availability

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

Declared In

`MacApplication.h`

GetApplicationTextEncoding

Returns the application text encoding for Resource Manager resources.

```
TextEncoding GetApplicationTextEncoding (
   void
);
```

Return Value
The application text encoding.

Discussion
Your application needs to use the application text encoding when it creates a CFString from text stored in Resource Manager resources. Typically the text uses a Mac encoding such as MacRoman or MacJapanese. For more information, see *Programming With the Text Encoding Conversion Manager*.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
QTCarbonShell

Declared In
`MacApplication.h`

GetSystemUIMode

Gets the presentation mode of the calling application.

```
void GetSystemUIMode (
   SystemUIMode *outMode,
   SystemUIOptions *outOptions
);
```

Parameters

outMode
On output, the caller's presentation mode. Pass `NULL` if you don't need this information. For a list of possible modes, see "Presentation Modes" (page 19). The presentation mode of an application determines which system-provided user interface elements are visible on the screen.

outOptions
On output, the options for the caller's presentation mode. Pass `NULL` if you don't need this information. For a list of possible options, see "Presentation Options" (page 20). Presentation options are used to inhibit or allow certain user interface elements and commands.

Discussion
This function returns information about the presentation mode of the calling application, not the presentation mode of the current login session. The login session mode may be different, since the login session mode is determined by the presentation mode of the frontmost application. If the calling application is not currently the frontmost application, then its presentation mode will not be in use. To track changes in the login session's presentation mode, you may handle the `kEventAppSystemUIModeChanged` Carbon event.

Availability
Available in Mac OS X v10.2 and later.

See Also

SetSystemUIMode (page 17)

Declared In

`MacApplication.h`

HIAboutBox

Displays a generic, HI-compliant about box.

```
OSStatus HIAboutBox (
   CFDictionaryRef inOptions
);
```

Parameters

`inOptions`

A dictionary of standard strings, a dictionary with the name of a localized strings file from which to retrieve the strings, or `NULL` to retrieve the strings from the `Info.plist` file. See the discussion for details.

Return Value

A result code.

Discussion

When this function is called, it displays a window called an about box that contains your application icon, name, software version, and other optional information. In a Carbon event–based application, the standard application event handler responds to the `kHICommandAbout` command by calling the function `HIAboutBox` for you. If your application menu has an About menu item, you will get this behavior for free. You don't need to call this function unless you want to customize the contents of the about box.

In addition to the application name and version, this function is designed to display two additional strings in the about box, a copyright string and a description string. You can customize what this function displays by passing in various options in the `inOptions` parameter:

- You can pass `NULL` to display application information defined in the `Info.plist` file or your bundle resource (not recommended). This function looks the `Info.plist` file for three keys to get the name, version, and copyright strings: `CFBundleName`, `CFBundleVersion`, and `CFBundleGetInfoString`.
- You can pass a dictionary with replacement values for one or more strings. See "About Box Keys" (page 18) for a list of valid keys in this dictionary. If a replacement string is not passed, the default behavior applies. For example, you could pass some variant of your application name in the dictionary, but not pass a replacement version string or copyright string. The function would display your replacement string, and fall back to looking in the `Info.plist` file for the other strings.
- You can pass a dictionary with a single entry, the name of a localized strings file that contains replacement values for one or more strings. The dictionary key is `kHIAboutBoxStringFileKey`, and the value is the name of the strings file without the `.strings` extension. This function automatically uses that file to find the strings for the about box. This example shows the key-value pairs in a typical strings file:

  ```
  HIAboutBoxName = "AboutBox";
  HIAboutBoxVersion = "v1.0";
  HIAboutBoxCopyright = "© Apple Computer, 2006";
  HIAboutBoxDescription = "An Example Application";
  ```

  Again, if a string is not found in that file, this function falls back to looking for a string in the dictionary, and then finally the `Info.plist` file.

Note that the description string can only be specified in an options dictionary or a strings file; this function does not check your `Info.plist` file for a description string.

Availability
Available in Mac OS X v10.3 and later.
Not available to 64-bit applications.

Declared In
`MacApplication.h`

HIApplicationCreateDockTileContext

Returns a Quartz graphics context for drawing in the application Dock tile.

```
CGContextRef HIApplicationCreateDockTileContext (
   HISize *outContextSize
);
```

Parameters

`outContextSize`
: On output, the size of the graphics context in which the application should draw.

Return Value
A Quartz graphics context you can use to draw in the application Dock tile. For more information about this context, see the Discussion below.

Discussion
This function makes it possible to draw into the application Dock tile at a resolution other than 128x128, which is the size of all Dock tiles prior to Mac OS X v10.5. In Mac OS X v10.5 and later, dock tiles may use different sizes when the user interface scale factor is not 1.0.

Unlike `BeginCGContextForApplicationDockTile` (page 7), this function returns a context that has no transform applied to it; user space and device space are 1:1. Your application must use the output context size to determine the area in which you should draw in the context.

Because the Dock's tile size can change dynamically, applications that use this function should be prepared to redraw their Dock tile as necessary. A `kEventAppUpdateDockTile` Carbon event is sent when the application needs to redraw its Dock tile.

This function locks the application Dock tile to prevent the Dock from drawing in the tile. When you are finished using the context, you must call the function `EndCGContextForApplicationDockTile` to release the context and the lock. Do not use `CGEndContext` or `CFRelease` for this purpose. To ensure that drawing to the context appears onscreen, you should call `CGContextFlush` before releasing the context.

Availability
Available in Mac OS X v10.5 and later.
Not available to 64-bit applications.

See Also
`EndCGContextForApplicationDockTile` (page 8)

Declared In
`MacApplication.h`

HIApplicationGetCurrent

Returns the currently running Carbon application object.

```
HIObjectRef HIApplicationGetCurrent (
   void
);
```

Return Value
The current application object.

Discussion
In Mac OS X v10.5 and later, you can use this function to install your own HIObject delegates on the application object.

The function `GetApplicationEventTarget` returns the event target associated with the application object.

Availability
Available in Mac OS X v10.2 and later.
Not available to 64-bit applications.

Declared In
`MacApplication.h`

HIApplicationGetFocus

Returns either the modeless or effective focused window.

```
WindowRef HIApplicationGetFocus (
   Boolean inConsideringModalFocus
);
```

Parameters
`inConsideringModalFocus`
A Boolean value that specifies whether to return the effective focus (`true`) or the modeless focus (`false`).

Return Value
The focused window, or `NULL` if there is no focused window.

Discussion
With the introduction of the modal focus stack in Mac OS X v10.5, an application may have two different focused windows: the modeless focus (the window most recently passed to the function `SetUserFocusWindow`), and the effective focus (either the modeless focus or, if there is a non-empty modal focus stack, the topmost window in the focus stack). This function returns either window.

Applications can use this function to determine if the modeless focus and effective focus are different windows. An application with a custom HIView can also use this function to determine if the application should show an insertion point. The insertion point should only be visible if the view is inside the effective focus.

Note that the function `GetUserFocusWindow` returns the modeless focus—the same window returned when you pass `false` to `HIApplicationGetFocus`.

Availability
Available in Mac OS X v10.5 and later.
Not available to 64-bit applications.

Declared In
`MacApplication.h`

HISearchWindowShow

Displays a Spotlight search window.

```
OSStatus HISearchWindowShow (
   CFStringRef inSearchString,
   OptionBits inFlags
);
```

Parameters

`inSearchString`
: An initial query string. Pass `NULL` to open the search window with no initial query string.

`inFlags`
: Optional flags. Currently, you should pass `kNilOptions`.

Return Value
A result code.

Discussion
This function displays a window with the standard Spotlight search interface. For more information, see *Spotlight Query Programming Guide*.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`MacApplication.h`

OverlayApplicationDockTileImage

Composites an image with your application's Dock icon.

```
OSStatus OverlayApplicationDockTileImage (
   CGImageRef inImage
);
```

Parameters

`inImage`
: The image to overlay onto your application Dock icon.

Return Value
A result code.

Discussion
You can overlay an image, such as a badge, to indicate the application’s status to the user.

Availability
Available in Mac OS X v10.0 and later.
Not available to 64-bit applications.

Declared In
`MacApplication.h`

RestoreApplicationDockTileImage

Restores your application Dock icon to the application icon.

```
OSStatus RestoreApplicationDockTileImage (
   void
);
```

Return Value
A result code.

Discussion
If you've called the functions `SetApplicationDockTileImage` or `OverlayApplicationDockTileImage`, you can use the function `RestoreApplicationDockTileImage` to restore the Dock icon to the original application icon.

Availability
Available in Mac OS X v10.0 and later.
Not available to 64-bit applications.

Declared In
`MacApplication.h`

SetApplicationDockTileImage

Replaces an application Dock icon.

```
OSStatus SetApplicationDockTileImage (
   CGImageRef inImage
);
```

Parameters

`inImage`
The image to use for your application Dock tile.

Return Value
A result code.

Discussion
When an application starts up, by default the application icon is always used as the application Dock tile. You can use the function `SetApplicationDockTileImage` to replace the application icon with another image. This can be useful to indicate the state of the application to the user. If you set the image, it will not revert back to its original image when your application terminates. You need to manually restore it before quitting using the function `RestoreApplicationDockTileImage`.

Availability
Available in Mac OS X v10.0 and later.
Not available to 64-bit applications.

Declared In
`MacApplication.h`

SetApplicationDockTileMenu

Adds items to the contextual menu for your application Dock tile.

```
OSStatus SetApplicationDockTileMenu (
   MenuRef inMenu
);
```

Parameters

`inMenu`

A menu containing the additional items, or `NULL` to remove the current menu.

Return Value

A result code.

Discussion

When you position the cursor over an application Dock tile and hold down the mouse button, a contextual menu automatically displays a list of the application's document windows and standard application Dock menu items such as Open at Login and Show in Finder. You can use the function `SetApplicationDockTileMenu` to add menu items to the contextual menu displayed for your application Dock tile. The items in the menu you pass to this function are inserted into your application Dock tile menu, between the list of document windows and the standard items.

This function increments the reference count of the menu you pass to it. Before the menu is displayed, it receives the Carbon events `kEventMenuPopulate`, `kEventMenuOpening`, and `kEventMenuEnableItems`, so any event handlers for these events can update the menu appropriately. You can receive notifications of and handle selections from the menu using `kEventCommandProcess` Carbon event handlers installed in the application event target. You must make sure each menu item has a command ID, as the `kEventCommandProcess` event sent to your application provides the menu item's command ID.

When you use this function to pass a menu to the Dock, the following state of each menu item is preserved:

- Information about whether the item is a text item or a separator
- The item text (if the item is not a separator)
- The item command ID
- The item command key modifiers (but not the command key itself)
- The item mark
- The item indent
- The item style
- The item icon, if the icon was specified with `kMenuSystemIconSelectorType` or `kMenuIconResourceType`
- The item's submenu
- These menu item attributes:
  - `kMenuItemAttrNotPreviousAlternate`
  - `kMenuItemAttrSectionHeader`
  - `kMenuItemAttrDisabled`
  - `kMenuItemAttrIconDisabled`
  - `kMenuItemAttrSubmenuParentChoosable`

- ❏ `kMenuItemAttrDynamic`

No other menu or menu item state is preserved when the menu is displayed by the Dock. For example, you can set a custom font for the menu or a menu item, but the menu as displayed by the Dock will not use that font.

Availability
Available in Mac OS X v10.1 and later.

Not available to 64-bit applications.

See Also
`GetApplicationDockTileMenu` (page 9)

Declared In
`MacApplication.h`

SetSystemUIMode

Sets the presentation mode of the calling application.

```
OSStatus SetSystemUIMode (
   SystemUIMode inMode,
   SystemUIOptions inOptions
);
```

Parameters

`inMode`

The new presentation mode. Pass one of the constants listed in "Presentation Modes" (page 19). The presentation mode of an application determines which system-provided user interface elements are visible on the screen.

`inOptions`

A mask that specifies options controlling how the specified presentation mode behaves. Pass one or more of the flags listed in "Presentation Options" (page 20), or zero to indicate that no options are needed. Presentation options are used to inhibit or allow certain user interface elements and commands.

Return Value
A result code.

Discussion
The purpose of this function is to make it easier to implement a dedicated kiosk system, in which the user is not permitted access to certain elements and features in the system user interface. This function gives your application control over the visibility of the Dock and the menu bar, and over various other system-provided user interface features such as process switching, logout, restart, and shutdown.

If your application is frontmost and you call this function to request a new presentation mode, your presentation mode will take effect immediately. If another application becomes frontmost, the presentation mode you requested will no longer be in effect. If your application becomes frontmost again, the presentation mode you previously established will come back into effect.

When the frontmost application uses this function to change its presentation mode, a `kEventAppSystemUIModeChanged` Carbon event is sent to all applications that have registered for the event. This event is also sent when an application is activated; it contains the newly active application's presentation mode.

In addition to using this function, an application may also specify an initial presentation mode when it is launched by using the `LSUIPresentationMode` key in its `Info.plist` file. This key should be of type `Number` and should have the value of one of the presentation mode constants listed in "Presentation Modes" (page 19).

Special Considerations

If your application uses the `LSUIElement` or `LSBackgroundOnly` key in its `Info.plist` file, you should not use this function. The presentation mode of the current login session is determined by the presentation mode of the frontmost application, and applications that use these keys generally do not become the frontmost application.

Availability
Available in Mac OS X v10.2 and later.

See Also
`GetSystemUIMode` (page 10)

Declared In
`MacApplication.h`

Constants

About Box Keys

Constants that specify keys used in an options dictionary passed to the function `HIAboutBox` (page 11).

```
#define kHIAboutBoxNameKey CFSTR("HIAboutBoxName")
#define kHIAboutBoxVersionKey CFSTR("HIAboutBoxVersion")
#define kHIAboutBoxCopyrightKey CFSTR("HIAboutBoxCopyright")
#define kHIAboutBoxDescriptionKey CFSTR("HIAboutBoxDescription")
#define kHIAboutBoxStringFileKey CFSTR("HIAboutBoxStringFile")
```

Constants

`kHIAboutBoxNameKey`

Key for the application name that replaces the name specified by the `CFBundleName` key in the `Info.plist` file.

Available in Mac OS X v10.3 and later.

Declared in `MacApplication.h`.

`kHIAboutBoxVersionKey`

Key for the application software version number that replaces the version number specified by the `CFBundleVersion` key in the `Info.plist` file.

Available in Mac OS X v10.3 and later.

Declared in `MacApplication.h`.

`kHIAboutBoxCopyrightKey`
: Key for the application copyright notice that replaces the text specified by the `CFBundleGetInfoString` key in the `Info.plist` file.

  Available in Mac OS X v10.3 and later.

  Declared in `MacApplication.h`.

`kHIAboutBoxDescriptionKey`
: Key for a short description of the application.

  Available in Mac OS X v10.3 and later.

  Declared in `MacApplication.h`.

`kHIAboutBoxStringFileKey`
: Key for the name of a localized strings file that contains about-box strings for the application.

  Available in Mac OS X v10.3 and later.

  Declared in `MacApplication.h`.

Discussion
The values associated with the keys in an options dictionary are all strings.

HIToolbox Version Number

Constant that specifies the current version number of HIToolbox.

```
const float kHIToolboxVersionNumber;
```

Constants
`kHIToolboxVersionNumber`
: The current HIToolbox version number, which is incremented each time that HIToolbox is rebuilt during the course of a Mac OS X release.

  Available in Mac OS X v10.3 and later.

  Declared in `MacApplication.h`.

Discussion
You can use this constant to check for the presence of bug fixes documented in HIToolbox release notes. For example, to test for the HIToolbox included in Mac OS X 10.4.2, check that `kHIToolboxVersionNumber` is at least 220. See the header file `MacApplication.h` for a list of the version numbers for specific releases.

Presentation Modes

Constants used to control the presentation of user interface elements provided by Mac OS X, such as the menu bar and Dock.

```
enum {
    kUIModeNormal = 0,
    kUIModeContentSuppressed = 1,
    kUIModeContentHidden = 2,
    kUIModeAllSuppressed = 4,
    kUIModeAllHidden = 3,
};
typedef UInt32 SystemUIMode;
```

Constants

`kUIModeNormal`

All standard system UI elements are visible.

Available in Mac OS X v10.2 and later.

Declared in `MacApplication.h`.

`kUIModeContentSuppressed`

System UI elements positioned in the content area of the screen (the area other than the menu bar) are hidden. However, these elements may automatically show themselves in response to mouse movements or other user activity.

Available in Mac OS X v10.2 and later.

Declared in `MacApplication.h`.

`kUIModeContentHidden`

System UI elements positioned in the content area of the screen (the area other than the menu bar) are hidden.

Available in Mac OS X v10.2 and later.

Declared in `MacApplication.h`.

`kUIModeAllSuppressed`

All system UI elements (including the menu bar) are hidden. However, these elements may automatically show themselves in response to mouse movements or other user activity.

Available in Mac OS X v10.3 and later.

Declared in `MacApplication.h`.

`kUIModeAllHidden`

All system UI elements (including the menu bar) are hidden.

Available in Mac OS X v10.2 and later.

Declared in `MacApplication.h`.

Discussion

The presentation mode of an application determines which system-provided user interface elements are visible on the screen. Your application can specify its presentation mode using the function `SetSystemUIMode` (page 17).

Presentation Options

Flags used to control optional behavior of system-provided user interface elements and features.

```
enum {
    kUIOptionAutoShowMenuBar = 1 << 0,
    kUIOptionDisableAppleMenu = 1 << 2,
    kUIOptionDisableProcessSwitch = 1 << 3,
    kUIOptionDisableForceQuit = 1 << 4,
    kUIOptionDisableSessionTerminate = 1 << 5,
    kUIOptionDisableHide = 1 << 6
};
typedef OptionBits SystemUIOptions;
```

Constants

`kUIOptionAutoShowMenuBar`

This flag specifies that the menu bar automatically shows itself when the user moves the mouse into the screen area that would ordinarily be occupied by the menu bar. Only valid for the presentation mode `kUIModeAllHidden`.

Available in Mac OS X v10.2 and later.

Declared in `MacApplication.h`.

`kUIOptionDisableAppleMenu`

This flag disables all items in the Apple menu. Valid for all presentation modes.

Available in Mac OS X v10.2 and later.

Declared in `MacApplication.h`.

`kUIOptionDisableProcessSwitch`

This flag disables the Command-Tab and Command-Shift-Tab key sequences to switch the active process, and the global window rotation key sequence selected by the user in the Keyboard preference pane. The function `SetFrontProcess` may still be used to explicitly switch the active process. Only valid with presentation modes other than `kUIModeNormal`.

Available in Mac OS X v10.2 and later.

Declared in `MacApplication.h`.

`kUIOptionDisableForceQuit`

This flag disables the Command-Option-Escape key sequence and the Force Quit menu item in the Apple menu to open the Force Quit window. Only valid with presentation modes other than `kUIModeNormal`.

Available in Mac OS X v10.2 and later.

Declared in `MacApplication.h`.

`kUIOptionDisableSessionTerminate`

This flag disables the Power key (if present) and the Restart, Shut Down, and Log Out menu items in the Apple menu. Only valid with modes other than `kUIModeNormal`.

Available in Mac OS X v10.2 and later.

Declared in `MacApplication.h`.

`kUIOptionDisableHide`

This flag disables the Hide menu item in the Application menu. Note that this option does not prevent this application from being hidden if Hide Others is selected in some other application.

Available in Mac OS X v10.3 and later.

Declared in `MacApplication.h`.

Discussion

Presentation mode options are used to inhibit or allow certain user interface elements and commands. Your application can specify these options using the function `SetSystemUIMode` (page 17).

Deprecated Application Manager Functions

A function identified as deprecated has been superseded and may become unsupported in the future.

Deprecated in Mac OS X v10.5

BeginQDContextForApplicationDockTile

Returns a QuickDraw graphics port for drawing in the application Dock tile. (Deprecated in Mac OS X v10.5. Use `BeginCGContextForApplicationDockTile` (page 7) or `HIApplicationCreateDockTileContext` (page 12) instead.)

```
CGrafPtr BeginQDContextForApplicationDockTile (
   void
);
```

Return Value
A pointer to a graphics port. You can use this port to draw into your application Dock tile with QuickDraw.

Discussion
This function locks the application Dock tile to prevent the Dock from drawing in the tile. When you are finished using the graphics port, you must call the function `EndQDContextForApplicationDockTile` to release the port and the lock. Do not use `DisposePort` for this purpose.

Availability
Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.5.

Not available to 64-bit applications.

See Also
`EndQDContextForApplicationDockTile` (page 23)

Declared In
`MacApplication.h`

EndQDContextForApplicationDockTile

Releases the QuickDraw graphics port for an application Dock tile. (Deprecated in Mac OS X v10.5. Use `EndCGContextForApplicationDockTile` (page 8) instead.)

```
void EndQDContextForApplicationDockTile (
   CGrafPtr inContext
);
```

Parameters

inContext

A QuickDraw graphics port acquired by calling `BeginQDContextForApplicationDockTile`. On output, the port is invalid and should no longer be used.

Discussion

This function also releases the lock on the application Dock tile, signaling the Dock that you are done drawing in the tile.

Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.5.

Not available to 64-bit applications.

See Also

`BeginQDContextForApplicationDockTile` (page 23)

Declared In

`MacApplication.h`

Document Revision History

This table describes the changes to *Application Manager Reference*.

Date	Notes
2007-10-31	Made minor editorial corrections.
2006-10-26	Updated for Mac OS X v10.5. Changed the title from "Dock Manager Reference."
2005-07-07	Fixed typographical errors.
2002-10-23	Updated formatting.
2001-08-30	First release of this document.

Index