Menu Manager Reference

Carbon > User Experience

2006-09-15

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

Simultaneously published in the United States and Canada.

Contents

Tables

Menu Manager Reference

Framework: Carbon/Carbon.h

Declared in Menus.h

Overview

You can use the Menu Manager to create and manage the menus in your application. Menus allow the user to view or choose from a list of choices and commands that your application provides. All Mac OS applications should provide these standard menus: the Application menu, the File menu, and the Edit menu.

Carbon supports the Menu Manager, with the following changes:

- Your application must use the functions defined by the Menu Manager whenever it creates and disposes of Menu Manager data structures. Some applications, for example, create menus by using the Resource Manager function `GetResource` (instead of the Menu Manager function `GetMenu`) and dispose of them by calling `ReleaseResource` instead of `DisposeMenu`. This practice is not supported in Carbon.
- The `MenuInfo` structure is opaque in Carbon. You must revise your application so that it accesses Menu Manager data structures only through accessor functions.
- Menu color information tables are not recommended in Carbon.
- You are strongly encouraged to adopt the standard Mac OS menu definition functions (also known as MDEFs) in your application. Your menus will then inherit the systemwide appearance and, furthermore, take advantage of other Menu Manager enhancements planned for the future. If you decide to customize the appearance of a menu and you are deploying your application for Mac OS X v10.3 and later, you should use a custom HIView instead of a custom menu definition function.

Functions by Task

Function descriptions are grouped by the tasks for which you use the functions. For an alphabetical list of functions, go to the API index at the end of the document.

Creating and Disposing of Menus

`CreateNewMenu` (page 34)
Creates a new, untitled, empty menu.

`DuplicateMenu` (page 41)
Creates a new menu that is a copy of another menu.

`DisposeMenu` (page 40)
Decrements the retain count of a menu.

`CreateCustomMenu` (page 33)
Creates a new, untitled, empty menu using a custom menu definition function.

`RegisterMenuDefinition` (page 89)
Registers a binding between a resource ID and a menu definition function.

`SetMenuDefinition` (page 95)
Sets the menu definition structure for a menu.

`GetMenuDefinition` (page 55)
Obtains the menu definition structure for a menu.

`CreateStandardFontMenu` (page 35)
Creates a standard font menu.

`UpdateStandardFontMenu` (page 111)
Updates a standard Font menu.

`ReleaseMenu` (page 184) Deprecated in Mac OS X v10.5
Decrements the retain count of a menu.

`RetainMenu` (page 185) Deprecated in Mac OS X v10.5
Increments the reference count of a menu.

Manipulating the Root Menu

`AcquireRootMenu` (page 21)
Get the menu whose contents are displayed in the menubar.

`SetRootMenu` (page 110)
Sets the menu whose contents are displayed in the menubar.

Manipulating the Menu Bar

`HideMenuBar` (page 72)
Conceals the menu bar.

`ShowMenuBar` (page 110)
Displays the menu bar.

`IsMenuBarVisible` (page 77)
Reports whether the menu bar is currently visible.

`FlashMenuBar` (page 45)
Highlights a menu title or the entire menu bar.

`SetMenuBar` (page 93)
Sets the current menu list to a specified menu list.

`GetMenuBar` (page 52)
Gets a handle to a copy of the current menu list.

`DrawMenuBar` (page 41)
Draws the menu bar based on the current menu list.

`InvalMenuBar` (page 76)
Invalidates the menu bar.

`GetMBarHeight` (page 50)
Determines the current height of the menu bar.

`DuplicateMenuBar` (page 42)
Duplicates a menubar handle.

`DisposeMenuBar` (page 41)
Releases a menubar handle.

Adding and Removing Menus

`InsertMenu` (page 73)
Inserts an existing menu into the current menu list.

`DeleteMenu` (page 36)
Deletes an existing menu from the current menu list.

`ClearMenuBar` (page 26)
Deletes all menus from the current menu list.

`GetMenuRetainCount` (page 173) Deprecated in Mac OS X v10.5
Returns the retain count of this menu.

Manipulating and Accessing Menu Characteristics

`GetMenuRef` (page 68)
Obtains a menu reference corresponding to a menu ID.

`GetMenuHandle` (page 57)
Obtains a menu reference corresponding to a menu ID.

`SetMenuTitleWithCFString` (page 108)
Sets the title of a menu to the text contained in a CFString.

`CopyMenuTitleAsCFString` (page 32)
Returns a CFString containing the title of a menu.

`GetMenuType` (page 69)
Gets the display type (pulldown, hierarchical, or popup) of a menu.

`SetMenuTitleIcon` (page 108)
Sets the title of a menu to be an icon.

`GetMenuTitleIcon` (page 68)
Retrieves the icon, if any, being used as the title of a menu.

`SetMenuID` (page 98)
Assigns a menu ID to a menu.

`GetMenuID` (page 58)
Obtains the ID of a menu.

`LMGetTheMenu` (page 83)
Returns the menu ID of the currently highlighted menu in the menu bar.

`SetMenuHeight` (page 97)
Set the height of a menu.

`GetMenuHeight` (page 57)
Obtains the height of a menu, in pixels.

`SetMenuWidth` (page 109)
Sets the width of a menu.

`GetMenuWidth` (page 70)
Obtains the width of the menu, in pixels.

`CalcMenuSize` (page 22)
Recalculates the horizontal and vertical dimensions of a menu.

`IsMenuSizeInvalid` (page 81)
Determines if a menu's size is invalid and should be recalculated.

`GetMenuAttributes` (page 51)
Gets the attributes of a menu.

`ChangeMenuAttributes` (page 23)
Changes the attributes of a menu.

`IsValidMenu` (page 83)
Determines if a menu is valid.

`SetMenuFlashCount` (page 189) Deprecated in Mac OS X v10.5
Specifies whether a menu should fade slowly or immediately disappear when closing.

Drawing Menus and Menu Items

`HiliteMenu` (page 72)
Highlights or unhighlights menu titles.

`UpdateInvalidMenuItems` (page 111)
Redraws the invalid items of an open menu.

`InvalidateMenuEnabling` (page 74)
Requests that the menu's enable state be recalculated.

`InvalidateMenuItems` (page 75)
Invalidates a group of menu items so that they will be redrawn when `UpdateInvalidMenuItems` is next called.

`InvalidateMenuSize` (page 76)
Invalidates the menu size so that it will be recalculated when next displayed.

`IsMenuBarInvalid` (page 77)
Determines if the menubar is invalid and should be redrawn.

`IsMenuItemInvalid` (page 80)
Determines if a menu item is invalid and should be redrawn.

`EraseMenuBackground` (page 169) Deprecated in Mac OS X v10.5
Erases the menu background to prepare for additional drawing.

`ScrollMenuImage` (page 185) Deprecated in Mac OS X v10.5
Scrolls a portion of the menu image.

Adding and Deleting Menu Items

`CopyMenuItems` (page 31)
Copies menu items from one menu to another.

`DeleteMenuItem` (page 36)
Deletes an item from a menu.

`DeleteMenuItems` (page 37)
Deletes multiple menu items.

Associating Data With Menu Items

`SetMenuItemProperty` (page 105)
Associates data with a menu item.

`SetMenuCommandProperty` (page 94)
Sets the property data for a menu item with the specified command ID.

`GetMenuItemProperty` (page 64)
Obtains a piece of data that has been previously associated with a menu item.

`GetMenuCommandProperty` (page 53)
Retrieves property data for a menu item with the specified command ID.

`GetMenuItemPropertySize` (page 66)
Obtains the size of a piece of data that has been previously associated with a menu item.

`GetMenuCommandPropertySize` (page 54)
Retrieves the size of the property data for a menu item with the specified command ID.

`RemoveMenuItemProperty` (page 90)
Removes a piece of data that has been previously associated with a menu item.

`RemoveMenuCommandProperty` (page 90)
Removes a property from a menu item with the specified command ID.

`GetMenuItemPropertyAttributes` (page 65)
Gets the attributes of a menu item property.

`ChangeMenuItemPropertyAttributes` (page 24)
Changes the attributes of a menu item property.

`SetMenuItemRefCon` (page 106)
Sets application-specific information for a menu item.

`GetMenuItemRefCon` (page 67)
Obtains application-specific information for a menu item.

Enabling Menus and Menu Items

`CountMenuItems` (page 32)
Obtains the number of menu items in a menu

`CountMenuItemsWithCommandID` (page 33)
Counts the menu items with a specified command ID.

`EnableMenuItem` (page 44)
Enables a menu item or a menu.

`EnableMenuCommand` (page 43)
Enables the menu item with a specified command ID.

`DisableMenuItem` (page 39)
Disables a menu item or a menu.

`DisableMenuCommand` (page 38)
Disables the menu item with a specified command ID.

`EnableAllMenuItems` (page 43)
Enables all items in a menu.

`DisableAllMenuItems` (page 38)
Disables all items in a menu.

`IsMenuItemEnabled` (page 78)
Reports whether a given menu or menu item is enabled.

`IsMenuCommandEnabled` (page 78)
Determines if the menu item with a specified command ID is enabled.

`MenuHasEnabledItems` (page 85)
Determines if any items in a menu are enabled.

`EnableMenuItemIcon` (page 44)
Enables the icon associated with a menu item.

`DisableMenuItemIcon` (page 39)
Disables the icon associated with a menu item.

`IsMenuItemIconEnabled` (page 79)
Reports whether a given menu item icon is enabled.

Manipulating Menu Item Text

`SetMenuItemTextWithCFString` (page 107)
Sets the text of a menu item to the text contained in a CFString.

`CopyMenuItemTextAsCFString` (page 31)
Returns a CFString containing the text of a menu item.

`AppendMenuItemTextWithCFString` (page 21)
Appends a new menu item with text from a CFString.

`InsertMenuItemTextWithCFString` (page 74)
Inserts a new menu item with text from a CFString.

`GetMenuFont` (page 56)
Obtains the font used in a menu.

`SetMenuFont` (page 97)
Sets the font to be used in a menu.

`GetMenuItemFontID` (page 60)
Obtains a menu item's font ID.

`SetMenuItemFontID` (page 101)
Sets the font for a menu item.

`GetFontFamilyFromMenuSelection` (page 46)
Gets the font family reference and style from the menu identifier and menu item number returned by the function `MenuSelect`.

`SetItemStyle` (page 93)
Sets a menu item's text style.

`GetItemStyle` (page 49)
Returns a menu item's text style.

`GetMenuItemTextEncoding` (page 173) Deprecated in Mac OS X v10.5
Obtains the text encoding used for a menu item's text.

`SetMenuItemTextEncoding` (page 191) Deprecated in Mac OS X v10.5
Sets the text encoding for a menu item's text.

Manipulating and Accessing Menu Item Characteristics

`SetMenuItemIconHandle` (page 102)
Sets a menu item's icon.

`GetMenuItemIconHandle` (page 62)
Obtains a handle to a menu item's icon.

`SetItemMark` (page 92)
Sets the mark of a menu item.

`SetMenuCommandMark` (page 94)
Locates the menu item with a specified command ID and sets its mark character.

`GetItemMark` (page 48)
Returns a menu item's mark.

`GetMenuCommandMark` (page 53)
Locates the menu item with a specified command ID and returns its mark character.

`CheckMenuItem` (page 25)
Adds or removes a check mark from a menu item.

`SetMenuItemIndent` (page 103)
Sets the indent level of a menu item.

`GetMenuItemIndent` (page 62)
Gets the indent level of a menu item.

`SetMenuItemKeyGlyph` (page 104)
Sets the command key glyph code for a menu item.

`GetMenuItemKeyGlyph` (page 63)
Obtains the keyboard glyph for a menu item's keyboard equivalent.

`SetMenuItemModifiers` (page 105)
Sets the modifier key(s) that must be pressed with a character key to select a particular menu item.

`GetMenuItemModifiers` (page 64)
Obtains the modifier keys that must be pressed with a character key to select a particular menu item.

`SetMenuItemCommandID` (page 98)
Sets a menu item's command ID.

`GetMenuItemCommandID` (page 59)
Obtains a menu item's command ID.

`SetMenuItemHierarchicalMenu` (page 101)
Attaches a submenu to a menu item.

`GetMenuItemHierarchicalMenu` (page 61)
Returns the submenu attached to a menu item.

`SetMenuItemCommandKey` (page 99)
Sets the keyboard equivalent of a menu item.

`GetMenuItemCommandKey` (page 59)
Gets the keyboard equivalent of a menu item.

`GetIndMenuItemWithCommandID` (page 46)
Finds a menu item with a specified command ID.

`SetMenuItemData` (page 100)
Sets multiple attributes of a menu item at once.

`CopyMenuItemData` (page 30)
Obtains multiple menu item attributes at once.

`GetMenuItemAttributes` (page 58)
Gets the attributes of a menu item.

`ChangeMenuItemAttributes` (page 24)
Changes the attributes of a menu item.

`GetItemIcon` (page 169) Deprecated in Mac OS X v10.5
Returns a menu item's icon or text encoding.

`GetMenuItemHierarchicalID` (page 172) Deprecated in Mac OS X v10.5
Obtains the menu ID of a specified submenu.

`SetItemIcon` (page 186) Deprecated in Mac OS X v10.5
Sets a menu item's icon or text encoding.

`SetMenuItemHierarchicalID` (page 189) Deprecated in Mac OS X v10.5
Attaches a submenu to a menu item.

Responding to Menu Events and User Selections

`IsMenuKeyEvent` (page 80)
Determines if an event corresponds to a menu command key.

`GetMenuTrackingData` (page 69)
Gets information about the menu currently selected by the user.

`CancelMenuTracking` (page 22)
Cancels menu tracking.

`MenuSelect` (page 86)
Allows the user to choose a menu item from a menu in the menu bar.

`MenuChoice` (page 84)
Returns the menu ID and index of the menu item under the cursor.

`IsShowContextualMenuEvent` (page 82)
Determines whether a particular Carbon event could invoke a contextual menu.

`ContextualMenuSelect` (page 28)
Displays a contextual menu.

`PopUpMenuSelect` (page 88)
Displays a pop-up menu without using the standard pop-up control definition function.

Handling Contextual Menu Plugins

`CMPluginExamineContext` (page 26)
An application-defined callback function that examines the context in a contextual menu CFPlugin.

`CMPluginHandleSelection` (page 27)
An application-defined callback function that handles menu item selection in a contextual menu CFPlugin.

`CMPluginPostMenuCleanup` (page 28)
An application-defined callback function that handles any post-selection cleanup in a contextual menu CFPlugin.

Obsolete Functions

`GetItemCmd` (page 48)
Returns the value of a menu item's keyboard equivalent field.

`GetMenu` (page 50)
Creates a menu from the specified menu and extended menu resources.

`GetMenuExcludesMarkColumn` (page 55)
Returns whether a menu contains space for mark characters.

`GetNewMBar` (page 70)
Reads in the definition of a menu bar from an '`MBAR`' resource.

`IsShowContextualMenuClick` (page 81)
Determines whether a particular event could invoke a contextual menu.

`MenuEvent` (page 84)
Maps a keyboard combination from the event structure to the keyboard equivalent of a menu item in a menu in the current menu list.

`SetItemCmd` (page 91)
Sets the value of the keyboard equivalent field of a menu item.

`SetMenuExcludesMarkColumn` (page 96)
Sets whether a menu contains space for mark characters.

`AppendMenu` (page 163) Deprecated in Mac OS X v10.5
Appends one or more items to a menu previously created.

`AppendMenuItemText` (page 164) Deprecated in Mac OS X v10.5
Appends a menu item to a menu.

`AppendResMenu` (page 165) Deprecated in Mac OS X v10.5
Searches all resource files open to your application for a given resource type and appends the names of any resources it finds to a specified menu.

`DeleteMCEntries` (page 166) Deprecated in Mac OS X v10.5
Deletes a menu item entry, a menu title entry, the menu bar entry, or all menu item entries of a specific menu from your application's menu color information table.

DisposeMCInfo (page 167) Deprecated in Mac OS X v10.5
Disposes of a menu color information table.

DisposeMenuDefUPP (page 168) Deprecated in Mac OS X v10.5
Disposes of universal procedure pointer to a custom menu definition.

GetMCEntry (page 170) Deprecated in Mac OS X v10.5
Gets information about an entry in an application's menu color information table.

GetMCInfo (page 171) Deprecated in Mac OS X v10.5
Returns a handle to a copy of your application's menu color information table.

GetMenuItemText (page 172) Deprecated in Mac OS X v10.5
Obtains the text of a menu item.

GetMenuTitle (page 174) Deprecated in Mac OS X v10.5
Obtains the title of the menu

InitContextualMenus (page 175) Deprecated in Mac OS X v10.5
Adds a program to the system registry of contextual menu clients.

InsertFontResMenu (page 175) Deprecated in Mac OS X v10.5
Inserts menu items from a font resource.

InsertIntlResMenu (page 176) Deprecated in Mac OS X v10.5
Inserts menu items from an internationalized resource.

InsertMenuItem (page 176) Deprecated in Mac OS X v10.5
Inserts one or more items into a menu previously created.

InsertMenuItemText (page 177) Deprecated in Mac OS X v10.5
Inserts a menu item into a menu.

InsertResMenu (page 178) Deprecated in Mac OS X v10.5
Searches all resource files open to your application for a given resource type and inserts the names of any resources it finds in the specified menu.

InvokeMenuDefUPP (page 179) Deprecated in Mac OS X v10.5
Calls your custom menu definition through a universal procedure pointer.

MenuKey (page 180) Deprecated in Mac OS X v10.5
Maps a character key with the command key to determine the keyboard equivalent of a menu item in a menu in the current menu list.

NewMenu (page 182) Deprecated in Mac OS X v10.5
Creates an empty menu with a specified title and menu ID.

NewMenuDefUPP (page 183) Deprecated in Mac OS X v10.5
Creates a new univeral procedure pointer to your custom menu definition.

ProcessIsContextualMenuClient (page 183) Deprecated in Mac OS X v10.5
Determines whether a given program is a contextual menu client.

SetMCEntries (page 187) Deprecated in Mac OS X v10.5
Sets entries in an application's menu color information table.

SetMCInfo (page 188) Deprecated in Mac OS X v10.5
Makes a copy of your application's menu color information table.

SetMenuItemText (page 190) Deprecated in Mac OS X v10.5
Sets menu item text to a specified string.

SetMenuTitle (page 192) Deprecated in Mac OS X v10.5
Sets the title of a menu.

Functions

AcquireRootMenu

Get the menu whose contents are displayed in the menubar.

```
MenuRef AcquireRootMenu (
   void
);
```

Return Value
The current root menu. See page for a description of the `MenuRef` data type. If no root menu currently exists, the Menu Manager creates one and returns its menu reference.

Discussion
This function increments the reference count of the root menu. The caller should call `ReleaseMenu` (page 184) when done with the menu.

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

Declared In
`Menus.h`

AppendMenuItemTextWithCFString

Appends a new menu item with text from a CFString.

```
OSStatus AppendMenuItemTextWithCFString (
   MenuRef inMenu,
   CFStringRef inString,
   MenuItemAttributes inAttributes,
   MenuCommand inCommandID,
   MenuItemIndex *outNewItem
);
```

Parameters

menu
The menu to which to append the new item.

inString
The text of the new item.

inAttributes
The attributes of the new item.

inCommandID
The command ID of the new item.

outNewItem
On exit, the index of the new item. May be `NULL` if the caller does not need this information.

Return Value
A result code. See "Menu Manager Result Codes" (page 161).

Discussion
The Menu Manager will make its own copy of the CFString before returning from this function. Modifying the string after calling `AppendMenuItemTextWithCFString` will have no effect on the menu item's actual text. The caller may release the string after the call.

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

Related Sample Code
BSDLLCTest
HID Explorer

Declared In
`Menus.h`

CalcMenuSize

Recalculates the horizontal and vertical dimensions of a menu.

```
void CalcMenuSize (
   MenuRef theMenu
);
```

Parameters

`theMenu`
The menu whose dimensions need recalculating.

Discussion
The `CalcMenuSize` function uses the menu definition function of the specified menu to calculate the dimensions of the menu. In most cases, your application does not need to use the `CalcMenuSize` function.

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

Declared In
`Menus.h`

CancelMenuTracking

Cancels menu tracking.

```
OSStatus CancelMenuTracking (
   MenuRef inRootMenu,
   Boolean inImmediate,
   UInt32 inDismissalReason
);
```

Parameters

inRootMenu

The root menu of the tracking session to dismiss. For menu bar tracking, pass the result from `AcquireRootMenu` (page 21); for popup menu tracking, pass the menu that was passed to `PopUpMenuSelect` (page 88).

inImmediate

Pass `true` if you want the open menus to disappear immediately, `false` if you want them to fade out.

inDismissalReason

Why the menu is being dismissed. This value is passed in the `kEventMenuEndTracking` event. You can pass the constants in "Menu Dismissal Constants" (page 159). If you pass zero here, the `kEventMenuEndTracking` event contains `kMenuDismissedByCancelMenuTracking`.

Return Value

A result code. See "Menu Manager Result Codes" (page 161).

Availability

Available in Mac OS X v10.3 and later.

Not available to 64-bit applications.

Declared In

`Menus.h`

ChangeMenuAttributes

Changes the attributes of a menu.

```
OSStatus ChangeMenuAttributes (
   MenuRef menu,
   MenuAttributes setTheseAttributes,
   MenuAttributes clearTheseAttributes
);
```

Parameters

menu

The menu whose attributes you want to change.

setTheseAttributes

The attributes to add to the menu. See "Menu Attribute Constants" (page 138) for a list of possible values.

clearTheseAttributes

The attributes to remove from the menu. See "Menu Attribute Constants" (page 138) for a list of possible values.

Return Value

A result code. See "Menu Manager Result Codes" (page 161).

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

Declared In
`Menus.h`

ChangeMenuItemAttributes

Changes the attributes of a menu item.

```
OSStatus ChangeMenuItemAttributes (
   MenuRef menu,
   MenuItemIndex item,
   MenuItemAttributes setTheseAttributes,
   MenuItemAttributes clearTheseAttributes
);
```

Parameters

menu
The menu.

item
The index of the menu item.

setTheseAttributes
The attributes to add to the menu item. See "Menu Item Attribute Constants" (page 139) for a list of possible values.

clearTheseAttributes
The attributes to remove from the menu item.

Return Value
A result code. See "Menu Manager Result Codes" (page 161).

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

Declared In
`Menus.h`

ChangeMenuItemPropertyAttributes

Changes the attributes of a menu item property.

```
OSStatus ChangeMenuItemPropertyAttributes (
   MenuRef menu,
   MenuItemIndex item,
   OSType propertyCreator,
   OSType propertyTag,
   OptionBits attributesToSet,
   OptionBits attributesToClear
);
```

Parameters

menu

The menu containing the menu item whose properties you want ot change.

item

The index of the menu item.

propertyCreator

The creator code of the property.

propertyTag

The property tag.

attributesToSet

The attributes to add to the menu item property.

attributesToClear

The attributes to remove from the menu item property.

Return Value

A result code. See "Menu Manager Result Codes" (page 161).

Discussion

Currently you can only specify the `kMenuPropertyPersistent` attribute (See "Menu Item Property Attribute Constant" (page 157)), which currently has no effect on Mac OS X. Therefore, `SetMenuItemPropertyAttributes` is not useful at this time.

Availability

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

Declared In

`Menus.h`

CheckMenuItem

Adds or removes a check mark from a menu item.

```
void CheckMenuItem (
    MenuRef theMenu,
    short item,
    Boolean checked
);
```

Parameters

theMenu

The menu containing the menu item to check or uncheck.

item
The menu index of the item to check or uncheck.

checked
Pass `true` to add a check, `false` to remove it.

Discussion
You can also add or remove a check mark using the `SetItemMark` (page 92) function.

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

Declared In
`Menus.h`

ClearMenuBar

Deletes all menus from the current menu list.

```
void ClearMenuBar (
   void
);
```

Discussion
The `ClearMenuBar` function deletes all menus from the current menu list. `ClearMenuBar` decrements the reference count of each menu in the menu list; if the reference count reaches zero, the memory associated with the menu is released. To explicitly release the memory occupied by a menu, use `DisposeMenu` (page 40).

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

Declared In
`Menus.h`

CMPluginExamineContext

An application-defined callback function that examines the context in a contextual menu CFPlugin.

```
OSStatus CMPluginExamineContext (
   void *thisInstance,
   const AEDesc *inContext,
   AEDescList *outCommandPairs
);
```

Parameters
thisInstance
The instance of this plugin.

inContext
The Apple event descriptor describing the selection that invoked the contextual menu.

outCommandPairs

On return, `outCommandPairs` points to an array of Apple event descriptors, each of which contains information about a menu item to display in the contextual menu.

Return Value
A result code. See "Menu Manager Result Codes" (page 161).

Discussion
Your contextual menu plugin must implement this function to determine what menu items to display given the user selection. The Apple event descriptor for each menu item contains the information to display in the item (such as text) and a command ID.

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

Declared In
`Menus.h`

CMPluginHandleSelection

An application-defined callback function that handles menu item selection in a contextual menu CFPlugin.

```
OSStatus CMPluginHandleSelection (
   void *thisInstance,
   AEDesc *inContext,
   SInt32 inCommandID
);
```

Parameters

thisInstance

The instance of this plugin.

inContext

The Apple event descriptor describing the selection that invoked the contextual menu.

inCommandID

The command ID associated with the user's menu item selection.

Return Value
A result code. See "Menu Manager Result Codes" (page 161).

Discussion
Your contextual menu plugin must implement this function to handle a user's selection of an item in the contextual menu. The command ID indicates which of the choices you gave in `CMPluginExamineContext` the user selected, so you can use that and the user selection stored in the `inContext` parameter to execute the conxtexual menu selection appropriately.

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

Declared In
`Menus.h`

CMPluginPostMenuCleanup

An application-defined callback function that handles any post-selection cleanup in a contextual menu CFPlugin.

```
void CMPluginPostMenuCleanup (
   void *thisInstance
);
```

Parameters

thisInstance

The instance of this plugin.

Discussion

Your contextual menu plugin must implement this function to handle any necessary cleanup required after displaying the contextual menu. If no cleanup is needed, you can simply return from this function.

Availability

Available in Mac OS X v10.1 and later.

Declared In

`Menus.h`

ContextualMenuSelect

Displays a contextual menu.

```
OSStatus ContextualMenuSelect (
   MenuRef inMenu,
   Point inGlobalLocation,
   Boolean inReserved,
   UInt32 inHelpType,
   ConstStr255Param inHelpItemString,
   const AEDesc *inSelection,
   UInt32 *outUserSelectionType,
   MenuID *outMenuID,
   MenuItemIndex *outMenuItem
);
```

Parameters

inMenu

The menu containing application commands to display. The caller creates this menu based on the current context, the mouse location, and the current selection (if it was the target of the mouse). If you pass `NULL`, only system commands are displayed.

inGlobalLocation

The location (in global coordinates) of the mouse near which the menu is to be displayed.

inReserved

Reserved for future use. Pass `false` for this parameter.

inHelpType

An identifier specifying the type of help provided by the application; see "Contextual Menu Help Type Constants" (page 132).

inHelpItemString

The string containing the text to be displayed for the help menu item. This string is unused unless you also pass the constant `kCMOtherHelp` in the `inHelpType` parameter.

inSelection

A pointer to an object specifier for the current selection. This allows the system to examine the selection and add special system commands accordingly. Passing a value of `NULL` indicates that no selection should be examined, and most likely, no special system actions will be included.

outUserSelectionType

A pointer to an unsigned 32-bit value. On return, the value indicates what the user selected from the contextual menu; see "Contextual Menu Selection Type Constants" (page 133) for a list of possible values.

outMenuID

A pointer to a signed 16-bit value. On return, if `outUserSelectionType` is set to `kCMMenuItemSelected`, the value is set to the menu ID of the chosen item.

outMenuItem

A pointer to an unsigned 16-bit value. On return, if `outUserSelectionType` is set to `kCMMenuItemSelected`, the value is set to the index of the menu item chosen.

Return Value

A result code. See "Menu Manager Result Codes" (page 161). Returns `userCanceledErr` and sets `outUserSelectionType` to `kCMNothingSelected` if the user selects an item that requires no additional actions on your part.

Discussion

If your application uses the standard window handler, you may want to install handlers for the `kEventWindowContextualMenuSelect` or `kEventControlContextualMenuClick` events and call `ContextualMenuSelect` from within your handler. The standard window handler automatically detects contextual menu clicks and sends the `kEventWindowContextualMenuSelect` and `kEventControlContextualMenuClick` events.

If the `IsShowContextualMenuEvent` function returns `true` or you receive the appropriate contextual menu Carbon event, you should call the `ContextualMenuSelect` function after generating your own menu and preparing an Apple Event descriptor (`AEDesc`) that describes the item for which your application is displaying a contextual menu. This descriptor may contain an object specifier or raw data and will be passed to all contextual menu plug-ins.

The system will add other items before displaying the contextual menu, and it will remove those items before returning, leaving the menu in its original state.

After all the system commands are added, the contextual menu is displayed and tracked. If the user selects one of the system items, it is handled by the system and the call returns as though the user didn't select anything from the menu. If the user selects any other item (or no item at all), the Menu Manager passes back appropriate values in the parameters `outUserSelectionType`, `outMenuID`, and `outMenuItem`.

Your application should provide visual feedback indicating the item that was clicked upon. For example, a click on an icon should highlight the icon, while a click on editable text should not eliminate the current selection.

If the `outUserSelectionType` parameter contains `kCMMenuItemSelected`, you should look at the `outMenuID` and `outMenuItem` parameters to determine what menu item the user chose and handle it appropriately. If the `outUserSelectionType` parameter contains `kCMShowHelpSelected`, you should open the proper help sequence.

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

Declared In
`Menus.h`

CopyMenuItemData

Obtains multiple menu item attributes at once.

```
OSStatus CopyMenuItemData (
   MenuRef inMenu,
   MenuItemID inItem,
   Boolean inIsCommandID,
   MenuItemDataPtr ioData
);
```

Parameters

menu

The menu whose attributes you want to get. Note that if you pass `true` for the `inIsCommandID` parameter, you can pass `NULL` here, in which case the Menu Manager searches the root menu for the first menu that matches the specified command ID.

item

The menu item index or the command ID of the menu item.

isCommandID

A Boolean value indicating whether the value passed for the `inItem` parameter is a command ID or a menu item index. Pass `true` to indicate a command ID, `false` to indicate that it is a menu item index. If you pass `true`, the Menu Manager returns the data for the first menu item that matches the specified command ID.

outData

A pointer to a `MenuItemDataRec` structure. Before calling, you should set the `whichData` field to indicate what data you want to obtain. (Individual fields may also require initialization before calling.) On return, the structure contains the data you requested. For more details on the types of data you can obtain, see "Menu Item Data Flags" (page 152).

Return Value
A result code. See "Menu Manager Result Codes" (page 161).

Discussion
You can use this function to obtain multiple menu item attributes simultaneously, which is often more efficient than making several different calls. For example, a menu definition function could use `CopyMenuItemData` to obtain all the individual attributes necessary for drawing a menu all at once.

This function returns copies of the data in the menu, so you should release any data in the `MenuItemDataRec` structure that was allocated dynamically (such as the CFString item text).

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

Declared In
`Menus.h`

CopyMenuItems

Copies menu items from one menu to another.

```
OSStatus CopyMenuItems (
   MenuRef inSourceMenu,
   MenuItemIndex inFirstItem,
   ItemCount inNumItems,
   MenuRef inDestMenu,
   MenuItemIndex inInsertAfter
);
```

Parameters

inSourceMenu

The menu from which to copy items.

inFirstItem

The first item to copy.

inNumItems

The number of items to copy.

inDestMenu

The menu to which to copy items.

inInsertAfter

The menu item in the destination menu after which to insert the copied items. Pass zero to insert items at the beginning of the menu. Note that you cannot specify an index value greater than the number of items in the destination menu.

Return Value

A result code. See "Menu Manager Result Codes" (page 161).

Availability

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

Declared In

`Menus.h`

CopyMenuItemTextAsCFString

Returns a CFString containing the text of a menu item.

```
OSStatus CopyMenuItemTextAsCFString (
   MenuRef inMenu,
   MenuItemIndex inItem,
   CFStringRef *outString
);
```

Parameters

menu

The menu containing the item.

item

The item whose text you want to copy.

outString
On exit, a CFString containing the item's text. The caller must release this string when it is no longer needed.

Return Value
A result code. See "Menu Manager Result Codes" (page 161).

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

Declared In
`Menus.h`

CopyMenuTitleAsCFString

Returns a CFString containing the title of a menu.

```
OSStatus CopyMenuTitleAsCFString (
   MenuRef inMenu,
   CFStringRef *outString
);
```

Parameters
inMenu
The menu whose title you want to obtain.

outString
On exit, a CFString containing the menu's title. This string must be released by the caller.

Return Value
A result code. See "Menu Manager Result Codes" (page 161).

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

Declared In
`Menus.h`

CountMenuItems

Obtains the number of menu items in a menu

```
UInt16 CountMenuItems (
   MenuRef theMenu
);
```

Parameters
theMenu
The menu whose items you want to count.

Return Value
The number of menu items in the menu.

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

Related Sample Code
HID Explorer

Declared In
`Menus.h`

CountMenuItemsWithCommandID

Counts the menu items with a specified command ID.

```
ItemCount CountMenuItemsWithCommandID (
   MenuRef inMenu,
   MenuCommand inCommandID
);
```

Parameters

menu

The menu in which to begin searching for items with the specified command ID. Pass `NULL` to begin searching with the root menu. The search will descend into all submenus of this menu.

commandID

The command ID for which to search.

Return Value
The number of menu items that match the specified command ID.

Version Notes
In CarbonLib 1.0.x, this function always returns zero or one; it stops after finding the first menu item with the specified command ID. In CarbonLib 1.1 and Mac OS X v10.0, it will count all menu items with the specified command ID.

In Mac OS X v10.0 and CarbonLib 1.0 through 1.4, this function searches only top-level menus (that is, those that are visible in the menu bar) and submenus of top-level menus. It does not search hierarchical menus that are in the menu bar but are not submenus of a top-level menu. For example, it does not search menus that are inserted for use in a popup menu. In Mac OS X v10.1 and later, and CarbonLib 1.5 and later, this function also searches hierarchical menus.

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

Declared In
`Menus.h`

CreateCustomMenu

Creates a new, untitled, empty menu using a custom menu definition function.

```
OSStatus CreateCustomMenu (
   const MenuDefSpec *inDefSpec,
   MenuID inMenuID,
   MenuAttributes inMenuAttributes,
   MenuRef *outMenuRef
);
```

Parameters

inDefSpec

A data structure that specifies a custom menu definition function.

inMenuID

The menu ID to use for the new menu.

inMenuAttributes

The menu attributes to use for the new menu. See "Menu Attribute Constants" (page 138) for a list of possible values.

outMenuRef

On exit, contains the new menu.

Return Value

A result code. See "Menu Manager Result Codes" (page 161).

Discussion

Similar to `CreateNewMenu` (page 34), but also allows you to create a menu from a custom menu definition function This definition can be procedure pointer–based or HIView-based, which you specify in the `MenuDefSpec` structure. (Note that HIView–based menus are available only in Mac OS X v10.3 and later.)

Availability

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

Declared In

`Menus.h`

CreateNewMenu

Creates a new, untitled, empty menu.

```
OSStatus CreateNewMenu (
   MenuID inMenuID,
   MenuAttributes inMenuAttributes,
   MenuRef *outMenuRef
);
```

Parameters

inMenuID

The menu ID to use for the new menu. Note that zero is a valid ID in Carbon.

inMenuAttributes

The menu attributes to use for the new menu. See "Menu Attribute Constants" (page 138) for a list of possible values.

outMenuRef

On exit, contains the new menu.

Return Value
A result code. See "Menu Manager Result Codes" (page 161).

Discussion
Unless you need to create new menus on-the-fly, you should not use functions like `CreateNewMenu` that create menus programmatically. Instead, you should define menus in Interface Builder, store them as nib files, and then call the Interface Builder Services function `CreateMenuFromNib` to create them.

`CreateNewMenu` is preferred over `NewMenu` because it allows you to specify the menu's attributes and it does not require you to specify a `Str255`-based menu title. To set the menu title, you should use `SetMenuTitleWithCFString` (page 108).

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

Declared In
`Menus.h`

CreateStandardFontMenu

Creates a standard font menu.

```
OSStatus CreateStandardFontMenu (
   MenuRef menu,
   MenuItemIndex afterItem,
   MenuID firstHierMenuID,
   OptionBits options,
   ItemCount *outHierMenuCount
);
```

Parameters

menu
The menu to which you want to add the font items.

afterItem
The item number of the menu item after which the new menu items are to be added. If you want to insert the new items before the first menu items, specify 0. If you want to insert the items after the last item in the menu, specify a number greater than or equal to the last item in the menu. Otherwise, specify the item number for the menu item after which you want to insert new items.

firstHierMenuID
The first menu ID to use if any hierarchical menus are created. This ID is incremented for each additional hierarchical menu.

options
An option bits structure that specifies the behavior for the Font menu. Specify the `Hierarchical Font Menu Option Constant` (page 137) if you want to construct a hierarchical font menu.

outHierMenuCount
On return, this parameter contains the number of hierarchical menus attached to the standard font menu. This value may be `NULL` if the hierarchical menus count is not useful. For example, if the only submenus in your application are those created by `CreateStandardFontMenu`, then you don't need to worry about the hierarchical menu count, as any existing submenu must be a font menu.

Return Value
A result code. See "Menu Manager Result Codes" (page 161).

Discussion
You should use this function instead of the functions `AppendResMenu` (page 165) or `InsertResMenu` (page 178) to designate objects in the font database as menu items in the Font menu. The fonts objects will appear by name.

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

Declared In
`Menus.h`

DeleteMenu

Deletes an existing menu from the current menu list.

```
void DeleteMenu (
    MenuID menuID
);
```

Parameters

menuID

The menu ID of the menu to delete from the current menu list. If the menu list does not contain a menu with the specified menu ID, `DeleteMenu` does nothing.

Discussion
The `DeleteMenu` function deletes the menu identified by the specified menu ID from the current menu list. `DeleteMenu` decrements the reference count of the menu, and if the reference count reaches zero, the memory occupied by the menu is released. To explicitly release the memory occupied by the menu, use `DisposeMenu` (page 40).

The `DeleteMenu` function first checks the submenu portion of the current menu list for a menu ID with the specified ID. If it finds such a menu, it deletes that menu and returns. If `DeleteMenu` doesn't find the menu in the submenu portion, it checks the regular portion of the current menu list.

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

Declared In
`Menus.h`

DeleteMenuItem

Deletes an item from a menu.

```
void DeleteMenuItem (
   MenuRef theMenu,
   MenuItemIndex item
);
```

Parameters

theMenu

The menu from which you want to delete the menu item.

item

The item number of the menu item to delete. If you specify 0 or a number greater than the last item in the menu, `DeleteMenuItem` does not delete any item from the menu.

Discussion

The `DeleteMenuItem` function deletes a specified menu item from a menu. You should not delete items from an existing menu unless the user expects the menu (such as a menu that lists open documents) to change.

Availability

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

Declared In

`Menus.h`

DeleteMenuItems

Deletes multiple menu items.

```
OSStatus DeleteMenuItems (
   MenuRef inMenu,
   MenuItemIndex inFirstItem,
   ItemCount inNumItems
);
```

Parameters

inMenu

The menu from which to delete items.

inFirstItem

The first item to delete.

inNumItems

The number of items to delete.

Return Value

A result code. See "Menu Manager Result Codes" (page 161).

Discussion

You can call this function rather than calling `DeleteMenuItem` (page 36) multiple times.

Availability

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

Related Sample Code

HID Explorer

Declared In
`Menus.h`

DisableAllMenuItems

Disables all items in a menu.

```
void DisableAllMenuItems (
   MenuRef theMenu
);
```

Parameters

theMenu

The menu whose items you want to disable.

Discussion
This function is equivalent to older code that masked the `enableFlags` field of the `MenuInfo` structure (now opaque in Carbon) with `0x01`. It disables all items (including items past item 31) but does not affect the state of the menu title.

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

Declared In
`Menus.h`

DisableMenuCommand

Disables the menu item with a specified command ID.

```
void DisableMenuCommand (
   MenuRef inMenu,
   MenuCommand inCommandID
);
```

Parameters

theMenu

The menu in which to begin searching for the item. Pass `NULL` to begin searching with the root menu. The search will descend into all submenus of this menu.

commandID

The command ID of the menu item to be disabled. If more than one item has this command ID, only the first will be disabled.

Discussion
If you have access to the menu item index, in most cases you should use `DisableMenuItem` (page 39) instead, as that function is faster and requires no searching.

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

Related Sample Code
CarbonSketch

Declared In
`Menus.h`

DisableMenuItem

Disables a menu item or a menu.

```
void DisableMenuItem (
   MenuRef theMenu,
   MenuItemIndex item
);
```

Parameters

theMenu

The menu containing the item to be disabled.

item

The index of the menu item that you wish to disable. Pass 0 to specify the menu title (disabling the entire menu).

Discussion
The `DisableMenuItem` function disables a menu item (and any associated icon) so that the user cannot choose the item from the menu.

Note that `EnableMenuItem` calls the `InvalMenuBar` (page 76) function to update the menu bar the next time through the event loop.

See also the `EnableMenuItem` (page 44) and `IsMenuItemEnabled` (page 78) functions.

Carbon Porting Notes

Note that the implementation of Carbon on Mac OS 8.1 only supports disabling menu items less than or equal to 31.

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

Related Sample Code
QTCarbonShell

Declared In
`Menus.h`

DisableMenuItemIcon

Disables the icon associated with a menu item.

```
void DisableMenuItemIcon (
   MenuRef theMenu,
   MenuItemIndex item
);
```

Parameters

theMenu

The menu containing the icon to be disabled.

item

The index of the menu item containing the icon.

Discussion

Your application can use the `DisableMenuItemIcon` function to dim individual menu item icons. The menu item that contains the icon is unaffected by calling `DisableMenuItemIcon`. That is, if `DisableMenuItemIcon` disables an enabled menu item's icon, the menu item itself will remain enabled. Calling `DisableMenuItemIcon` on the icon of a menu item that is currently disabled will cause the icon to be disabled once the menu item is re-enabled.

See also the functions `EnableMenuItemIcon` (page 44) and `IsMenuItemIconEnabled` (page 79).

Availability

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

Declared In

`Menus.h`

DisposeMenu

Decrements the retain count of a menu.

```
void DisposeMenu (
   MenuRef theMenu
);
```

Parameters

theMenu

The menu whose retain count to decrement. If the retain count falls to zero, the menu is destroyed.

Discussion

The reference that you pass in the `theMenu` parameter is not valid after `DisposeMenu` returns. This function is identical to `ReleaseMenu` (page 184).

Availability

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

Related Sample Code

HID Explorer

Declared In

`Menus.h`

DisposeMenuBar

Releases a menubar handle.

```
OSStatus DisposeMenuBar (
   MenuBarHandle inMbar
);
```

Parameters

mbar

The menubar handle to release.

Return Value

A result code. See "Menu Manager Result Codes" (page 161).

Discussion

Carbon applications should call this function when releasing a handle returned from `GetNewMBar` (page 70), `GetMenuBar` (page 52), or `DuplicateMenuBar` (page 42). Doing so ensures that the reference counts of the menus in the menubar handle can be decremented when the handle is released.

Do not call the Memory Manager function `DisposeHandle` to release such a handle.

Availability

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

Declared In

`Menus.h`

DrawMenuBar

Draws the menu bar based on the current menu list.

```
void DrawMenuBar ();
```

Discussion

The `DrawMenuBar` function draws (or redraws) the menu bar according to the current menu list. Note that most Menu Manager calls that affect the menu bar call `InvalMenuBar` (page 76) so that the menu bar is redrawn the next time through the event loop; however, you can call `DrawMenuBar` if you want the changes to appear immediately.

Availability

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

Related Sample Code

Simple DrawSprocket

Declared In

`Menus.h`

DuplicateMenu

Creates a new menu that is a copy of another menu.

```
OSStatus DuplicateMenu (
   MenuRef inSourceMenu,
   MenuRef *outMenu
);
```

Parameters

inSourceMenu

The menu to duplicate.

outMenu

On exit, a copy of the source menu.

Return Value
A result code. See "Menu Manager Result Codes" (page 161).

Discussion
Unlike `RetainMenu` (page 185), `DuplicateMenu` creates an entirely new menu that is an exact copy of the original menu. The menu definition for the new menu will receive an initialization message/event after the menu has been fully created.

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

Declared In
`Menus.h`

DuplicateMenuBar

Duplicates a menubar handle.

```
OSStatus DuplicateMenuBar (
   MenuBarHandle inMbar,
   MenuBarHandle *outMbar
);
```

Parameters

inMbar

The menubar handle to duplicate.

outMBar

On exit, contains the new menubar handle.

Return Value
A result code. See "Menu Manager Result Codes" (page 161).

Discussion
Carbon applications should use this function when duplicating a handle returned from `GetMenuBar` or `GetNewMBar`. Doing so ensures that the reference counts of the menus in the menubar handle can be incremented when the handle is duplicated.

Do not use Memory Manager APIs (`HandToHand`, `NewHandle`, and so on) to duplicate such a handle.

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

Declared In
Menus.h

EnableAllMenuItems

Enables all items in a menu.

```
void EnableAllMenuItems (
   MenuRef theMenu
);
```

Parameters

theMenu

The menu whose items to enable.

Discussion
This function is equivalent to older code that OR'd the `enableFlags` field of the `MenuInfo` structure (now opaque in Carbon) with 0xFFFFFFFE. It enables all items (including items past item 31) but does not affect the state of the menu title.

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

Declared In
Menus.h

EnableMenuCommand

Enables the menu item with a specified command ID.

```
void EnableMenuCommand (
   MenuRef inMenu,
   MenuCommand inCommandID
);
```

Parameters

inMenu

The menu in which to begin searching for the item. Pass `NULL` to begin searching with the root menu. The search will descend into all submenus of this menu.

inCommandID

The command ID of the menu item to be enabled. If more than one item has this command ID, only the first will be enabled.

Discussion
If you have access to the menu item index, in most cases you should use `EnableMenuItem` (page 44) instead, as that function is faster and requires no searching. For example, when receiving a `kEventCommandUpdateStatus` event, the `HICommand` structure contains both the menu item's command ID and index. If you wanted to enable the menu item, you should call `EnableMenuItem` (page 44).

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

Related Sample Code
CarbonSketch
QTCarbonShell

Declared In
`Menus.h`

EnableMenuItem

Enables a menu item or a menu.

```
void EnableMenuItem (
    MenuRef theMenu,
    MenuItemIndex item
);
```

Parameters

theMenu
The menu containing the item to be enabled.

item
The item number of the menu item that you wish to enable. If you pass 0, `EnableMenuItem` enables the menu title and all items in the menu that were not previously individually disabled.

Discussion
The `EnableMenuItem` function enables a menu item so that the user can choose the item from the menu. If the menu item has an associated icon, that icon is also enabled, unless the icon was previously individually disabled with the function `DisableMenuItemIcon` (page 39).

Note that `EnableMenuItem` calls the `InvalMenuBar` (page 76) function to update the menu bar the next time through the event loop.

See also the `DisableMenuItem` (page 39) and `IsMenuItemEnabled` (page 78) functions.

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

Related Sample Code
QTCarbonShell

Declared In
`Menus.h`

EnableMenuItemIcon

Enables the icon associated with a menu item.

```
void EnableMenuItemIcon (
   MenuRef theMenu,
   MenuItemIndex item
);
```

Parameters

theMenu

The menu containing the icon to be enabled.

item

The item number of the menu item containing the icon.

Discussion

Your application can use the `EnableMenuItemIcon` function to enable individual menu item icons that have been previously disabled by a call to the function `DisableMenuItemIcon` (page 39). The menu item that contains the icon is unaffected by calling `EnableMenuItemIcon`. Note that enabling the icon of a currently disabled menu item has no visual effect; however, once the menu item is enabled, the icon is also enabled.

See also the `DisableMenuItemIcon` (page 39) and `IsMenuItemIconEnabled` (page 79) functions.

Availability

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

Declared In

`Menus.h`

FlashMenuBar

Highlights a menu title or the entire menu bar.

```
void FlashMenuBar (
   MenuID menuID
);
```

Parameters

menuID

The menu ID of the menu whose title you want to highlight. Pass zero in this parameter to signal a visual alert (that is, flash the entire screen) If you pass a menu ID that is not in the menubar, `FlashMenuBar` returns immediately without flashing the bar or any menu title.

Discussion

Call this function twice if you want the menu or menu bar to blink.

Only one menu title can be highlighted at a time. If no menus are currently highlighted, calling `FlashMenuBar` with a specific menu ID highlights the title of that menu. If you call `FlashMenuBar` again specifying another menu ID that is different from that of the previously highlighted menu title, `FlashMenuBar` restores the previously highlighted menu to normal and then highlights the title of the specified menu.

If you pass zero for the menu ID, this function flashes the entire screen, as if an alert had occurred while the user had specified the “Flash the screen when an alert sound occurs” checkbox in the Universal Access preference pane.

Availability

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

Declared In
Menus.h

GetFontFamilyFromMenuSelection

Gets the font family reference and style from the menu identifier and menu item number returned by the function MenuSelect.

```
OSStatus GetFontFamilyFromMenuSelection (
   MenuRef menu,
   MenuItemIndex item,
   FMFontFamily *outFontFamily,
   FMFontStyle *outStyle
);
```

Parameters

menu
A menu handle.

item
A menu item index.

outFontFamily
A pointer to the font family reference associated with the menu item.

outStyle
A pointer to the font style associated with the menu item.

Return Value
A result code. See "Menu Manager Result Codes" (page 161).

Discussion
You use this function to obtain information from a font menu created using CreateStandardFontMenu (page 35), AppendResMenu (page 165), InsertFontResMenu (page 175), InsertIntlResMenu (page 176), or InsertResMenu (page 178).

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

Declared In
Menus.h

GetIndMenuItemWithCommandID

Finds a menu item with a specified command ID.

```
OSStatus GetIndMenuItemWithCommandID (
   MenuRef inMenu,
   MenuCommand inCommandID,
   UInt32 inItemIndex,
   MenuRef *outMenu,
   MenuItemIndex *outIndex
);
```

Parameters

inMenu

The menu in which to begin searching for items with the specified command ID. Pass `NULL` to begin searching with the root menu. The search will descend into all submenus of this menu.

inCommandID

The command ID for which to search.

inItemIndex

The 1-based index of the menu item to retrieve. In CarbonLib 1.0.x, this parameter must be 1. In CarbonLib 1.1 and Mac OS X 1.0, this parameter may vary from 1 to the number of menu items with the specified command ID.

outMenu

On exit, the menu containing the menu item with the specified command ID.

outIndex

On exit, the item index of the menu item with the specified command ID.

Return Value

A result code. See "Menu Manager Result Codes" (page 161).

Discussion

This API searches the specified menu and its submenus for the n'th menu item with the specified command ID. You often want to use this function in conjunction with `CountMenuItemsWithCommandID` (page 33).

Version Notes

In CarbonLib 1.1 and earlier, only the first menu item will be returned. In CarbonLib 1.2 and Mac OS X v10.0 and later, this API will iterate over all menu items with the specified command ID and return the `itemIndex`'th one.

In Mac OS X v10.0 and CarbonLib 1.0 through 1.4, this function searches only top-level menus (that is, those that are visible in the menu bar) and submenus of top-level menus. It does not search hierarchical menus that are in the menu bar but are not submenus of a top-level menu. For example, it does not search menus that are inserted for use in a popup menu. In Mac OS X v10.1 and later, and CarbonLib 1.5 and later, this function also searches hierarchical menus.

Availability

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

Related Sample Code

CarbonSketch

Declared In

`Menus.h`

GetItemCmd

Returns the value of a menu item's keyboard equivalent field.

Not recommended

```
void GetItemCmd (
   MenuRef theMenu,
   MenuItemIndex item,
   CharParameter *cmdChar
);
```

Parameters

theMenu

The menu containing the menu item whose keyboard equivalent you want to obtain.

item

An integer representing the item number of the menu item whose keyboard equivalent you want to determine.

cmdChar

On output, an integer representing the item's keyboard equivalent field. The Menu Manager uses this value to map keyboard equivalents to menu commands or to indicate special characteristics of the menu item.

If the value referenced through the `cmdChar` parameter contains 0x1B, the menu item has a submenu; a value of 0x1C indicates that the item has a text encoding; a value of 0x1D indicates that the Menu Manager reduces the item's '`ICON`' resource; and a value of 0x1E indicates that the item has an '`SICN`' resource.

Discussion

You should call `GetMenuItemCommandKey` (page 59) , `GetMenuItemHierarchicalID` (page 172) , and `GetMenuItemTextEncoding` (page 173) instead of `GetItemCmd` to obtain a menu item's keyboard equivalent and text encoding and to determine that a menu item has a submenu.

The `GetItemCmd` function returns the value in the keyboard equivalent field of the specified menu item in the value pointed to by the `cmdChar` parameter (or 0 if the item doesn't have a keyboard equivalent, submenu, text encoding, reduced icon, or small icon).

Availability

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

Declared In

`Menus.h`

GetItemMark

Returns a menu item's mark.

```
void GetItemMark (
   MenuRef theMenu,
   MenuItemIndex item,
   CharParameter *markChar
);
```

Parameters

theMenu

The menu containing the item.

item

The menu index of the item.

markChar

On output, an integer representing the mark of the menu item or its submenu (item has a submenu). See the Font Manager for a list of character marking constants that this function can obtain. This parameter is set to 0 if the menu item has neither mark nor submenu.

Discussion

You should call `GetMenuItemHierarchicalID` (page 172) instead of `GetItemMark` to obtain the menu ID of a menu item's submenu. However, you can still use `GetItemMark` to obtain the mark of a menu item.

Availability

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

Related Sample Code

QTCarbonShell

Declared In

`Menus.h`

GetItemStyle

Returns a menu item's text style.

```
void GetItemStyle (
   MenuRef theMenu,
   MenuItemIndex item,
   Style *chStyle
);
```

Parameters

theMenu

The menu containing the item.

item

The menu index of the item.

chStyle

On output, an integer representing the menu item's text style. The functions returns one of the following constants: `normal`, `bold`, `italic`, `underline`, `outline`, `shadow`, `condense`, and `extend`.

Availability

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

Declared In
Menus.h

GetMBarHeight

Determines the current height of the menu bar.

```
short GetMBarHeight (
   void
);
```

Return Value
The current height, in pixels, of the menu bar.

Discussion
The GetMBarHeight function determines the menu bar height based on factors such as the current script system and theme.

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

Declared In
Menus.h

GetMenu

Creates a menu from the specified menu and extended menu resources.

Not recommended

```
MenuRef GetMenu (
    short resourceID
);
```

Parameters

resourceID

The resource ID of the menu and extended menu that defines the characteristics of the menu. You typically use the same number for a menu's resource ID as the number that you specify for the menu ID in the menu resource.

Return Value
The new menu. You can use the returned menu handle to refer to this menu in most Menu Manager functions. If GetMenu is unable to read the menu or menu definition function from the resource file, GetMenu returns null. See the description of the MenuRef data type.

Discussion
Unless you must support legacy code, you should not use functions like GetMenu that rely on menus stored as resources. Instead, you should define menus in Interface Builder, store them as nib files, and then call the Interface Builder Services function CreateMenuFromNib to create them.

GetMenu reads the menu definition function into memory (if not already present) and stores a handle to the menu definition function in the menu structure. GetMenu does not insert the newly created menu into the current menu list.

You typically use the GetMenu function only when you create submenus; you can create all your pull-down menus at once using the function GetNewMBar, and you can create pop-up menus using the standard pop-up menu button control definition function.

After reading the 'MENU' resource, GetMenu searches for an extended menu resource and an 'mctb' resource with the same resource ID as the 'MENU' resource. If the specified 'mctb' resource exists, GetMenu uses SetMCEntries to add the entries defined by the resource to the application's menu color information table. If the 'mctb' resource does not exist, GetMenu uses the default colors specified in the menu bar entry of the application's menu color information. If neither a menu bar entry nor a 'mctb' resource exists, GetMenu uses the standard colors for the menu.

Storing the definitions of your menus in resources (especially menu titles and menu items) makes your application easier to localize.

After creating a menu with GetMenu or NewMenu (page 182) , use InsertMenuItem (page 176) , AppendMenu (page 163) , or InsertResMenu (page 178) to add menu items to the menu. To add a menu created by GetMenu to the current menu list, use InsertMenu (page 73). To update the menu bar with any new menu titles, use DrawMenuBar (page 41).

Menus in a resource must not be purgeable nor should the resource lock bit be set. Do not define a "circular" hierarchical menu—that is, a hierarchical menu in which a submenu has a submenu whose submenu is a hierarchical menu higher in the chain.

Special Considerations

To release the memory associated with a menu that you created using GetMenu, first call DeleteMenu to remove the menu from the current menu list and to remove any entries for this menu in your application's menu color information table then call DisposeMenu to dispose of the menu structure.

Carbon Porting Notes

In Carbon, the GetMenu function always returns a newly created MenuRef. Prior to Carbon, GetMenu would first check if the menu was already in memory. If so, GetMenu would return the in-memory copy. This behavior is no longer supported.

Carbon does not support custom menu definitions stored in 'MDEF' resources. If you want to specify a custom menu definition for GetMenu, you must compile your definition function directly in your application and then register the function by calling RegisterMenuDefinition (page 89). When GetMenu gets a resourceID value that doesn't recognize, it checks a special mapping table to find the pointer that's registered for the resourceID parameter. It then calls that function to implement your menu.

Availability

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

Declared In

Menus.h

GetMenuAttributes

Gets the attributes of a menu.

```
OSStatus GetMenuAttributes (
   MenuRef menu,
   MenuAttributes *outAttributes
);
```

Parameters

menu

The menu.

outAttributes

On exit, contains the attributes of the menu. See "Menu Attribute Constants" (page 138) for a list of possible values.

Return Value
A result code. See "Menu Manager Result Codes" (page 161).

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

Declared In
`Menus.h`

GetMenuBar

Gets a handle to a copy of the current menu list.

```
MenuBarHandle GetMenuBar (
   void
);
```

Return Value
A handle to a copy of the current menu list. See the description of the `MenuBarHandle` data type.

Discussion
The `GetMenuBar` function creates a copy of the current menu list and returns a handle to the copy. You can save the returned menu list and then add menus to or remove menus from the current menu list using `InsertMenu` (page 73) , `DeleteMenu` (page 36) , or `ClearMenuBar` (page 26). You can later restore the saved menu list using `SetMenuBar` (page 93).

To release the memory occupied by a saved menu list, use the `DisposeMenuBar` (page 41) function.

`GetMenuBar` doesn't copy the menu structures, just the menu list (which contains handles to the menu structures). Do not dispose of any menus in a saved menu list if you wish to restore the menu list later.

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

Declared In
`Menus.h`

GetMenuCommandMark

Locates the menu item with a specified command ID and returns its mark character.

```
OSStatus GetMenuCommandMark (
   MenuRef inMenu,
   MenuCommand inCommandID,
   UniChar *outMark
);
```

Parameters

theMenu

The menu in which to begin searching for the item. Pass `NULL` to begin searching with the root menu. The search will descend into all submenus of this menu.

commandID

The command ID of the menu item to be examined. If more than one item has this command ID, only the first will be examined.

outMark

On exit, the menu item's mark character.

Return Value

A result code. See "Menu Manager Result Codes" (page 161).

Discussion

If you have access to the menu item index, in most cases you should use `GetItemMark` (page 48) instead, as that function is faster and requires no searching.

Availability

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

Declared In

`Menus.h`

GetMenuCommandProperty

Retrieves property data for a menu item with the specified command ID.

```
OSStatus GetMenuCommandProperty (
   MenuRef inMenu,
   MenuCommand inCommandID,
   OSType inPropertyCreator,
   OSType inPropertyTag,
   ByteCount inBufferSize,
   ByteCount *outActualSize,
   void *inPropertyBuffer
);
```

Parameters

inMenu

The menu in which to search for the command ID. Pass `NULL` to begin searching with the root menu. The search will descend into all submenus of this menu.

inCommandID

The command ID of the menu item containing the property. Note that if more than one item has the same command ID, only the first match is used.

inPropertyCreator

The four-character creator code of the application.

inPropertyTag

The four-character tag of the property to obtain.

inBufferSize

The size of the buffer to hold the retrieved data, in bytes.

outActualSize

The actual size of the property data. If you do not need this information, pass `NULL`.

inPropertyBuffer

A pointer to the buffer in which to place the data. On return, the buffer contains the property data.

Return Value

A result code. See "Menu Manager Result Codes" (page 161).

Discussion

If you have access to the menu item index, in most cases you should use `GetMenuItemProperty` (page 64) instead, as that function is faster and requires no searching.

Availability

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

Declared In

`Menus.h`

GetMenuCommandPropertySize

Retrieves the size of the property data for a menu item with the specified command ID.

```
OSStatus GetMenuCommandPropertySize (
   MenuRef inMenu,
   MenuCommand inCommandID,
   OSType inPropertyCreator,
   OSType inPropertyTag,
   ByteCount *outSize
);
```

Parameters

menu

The menu in which to search for the command ID. Pass `NULL` to begin searching with the root menu. The search will descend into all submenus of this menu.

commandID

The command ID of the menu item containing the property. If more than one menu item has the same command ID, only the first match is used.

propertyCreator

The four-character creator code of the application.

propertyTag

The four-character tag of the property data whose size you want to obtain.

size

On return, contains the size of the property data.

Return Value
A result code. See "Menu Manager Result Codes" (page 161).

Discussion
If you have access to the menu item index, in most cases you should use `GetMenuItemPropertySize` (page 66) instead, as that function is faster and requires no searching.

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

Declared In
`Menus.h`

GetMenuDefinition

Obtains the menu definition structure for a menu.

```
OSStatus GetMenuDefinition (
   MenuRef menu,
   MenuDefSpecPtr outDefSpec
);
```

Parameters

menu

The menu whose menu definition structure you want to obtain.

outDefSpec

On return, a pointer to the menu's `MenuDefSpec` structure.

Return Value
A result code. See "Menu Manager Result Codes" (page 161).

Discussion
Note that you cannot use this function to obtain the standard system menu definition. If you call `GetMenuDefinition` on a menu that uses the standard system MDEF or menu content HIView, the function returns `menuUsesSystemDefErr`.

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

Declared In
`Menus.h`

GetMenuExcludesMarkColumn

Returns whether a menu contains space for mark characters.

Not recommended

```
Boolean GetMenuExcludesMarkColumn (
   MenuRef menu
);
```

Parameters

menu

The menu whose width is to be examined.

Return Value

Returns `true` if the menu currently contains no space for mark characters; `false` if the menu is currently drawn in its full width, with space for mark characters.

Discussion

See also the `SetMenuExcludesMarkColumn` (page 96) function.

Carbon Porting Notes

You should instead inspect the `kMenuExcludesMarkColumn` menu attribute using the `GetMenuAttributes` function.

Availability

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

Declared In

`Menus.h`

GetMenuFont

Obtains the font used in a menu.

```
OSStatus GetMenuFont (
   MenuRef menu,
   SInt16 *outFontID,
   UInt16 *outFontSize
);
```

Parameters

menu

The menu whose font is to be obtained.

outFontID

On input, a pointer to a signed 16-bit integer. On return, this value identifies the font family ID for the menu font. Note that this is the menu item font, not the menu title font.

outFontSize

On input, a pointer to an unsigned 16-bit integer. On return, this value identifies the size of the font, in points.

Return Value

A result code. See "Menu Manager Result Codes" (page 161).

Discussion

Your application may use the `GetMenuFont` function to retrieve the font used for an individual menu, such as a pop-up menu.

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

Declared In
Menus.h

GetMenuHandle

Obtains a menu reference corresponding to a menu ID.

```
MenuRef GetMenuHandle (
   MenuID menuID
);
```

Parameters

menuID

The menu ID of the menu whose reference you want to obtain. (Note that this is not the resource ID, although you often assign the menu ID so that it matches the resource ID.) You assign a menu ID in a nib file or in the 'MENU' resource of a menu. If you do not define your menus in nib files or in 'MENU' resources, you can assign a menu ID using NewMenu (page 182) or SetMenuID (page 98).

Return Value
The menu corresponding to the specified ID. If the specified menu is not in the current menu list, GetMenuHandle returns NULL. See the description of the MenuRef data type.

Discussion
You can also call this function as GetMenuRef (menuID);.

Use the GetMenuHandle function to obtain the menu reference for any of your application's pull-down menus or submenus in the current menu list, other than the Help menu. You can also use the Help Manager function HMGetHelpMenuHandle to get a handle for your application's Help menu.

Special Considerations

To get a menu reference for a pop-up menu that you create using the pop-up control definition function, call the Control Manager functions GetControlData and GetControlDataSize, passing the tag constant kControlPopupButtonMenuRefTag in the tagName parameter to specify the menu reference.

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

Related Sample Code
Simple DrawSprocket

Declared In
Menus.h

GetMenuHeight

Obtains the height of a menu, in pixels.

```
SInt16 GetMenuHeight (
   MenuRef menu
);
```

Parameters

menu

The menu whose height you want to obtain.

Return Value
The height of the menu, in pixels.

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

Declared In
`Menus.h`

GetMenuID

Obtains the ID of a menu.

```
MenuID GetMenuID (
   MenuRef menu
);
```

Parameters

menu

Return Value
The menu ID of the menu. See the description of the `MenuID` data type.

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

Declared In
`Menus.h`

GetMenuItemAttributes

Gets the attributes of a menu item.

```
OSStatus GetMenuItemAttributes (
   MenuRef menu,
   MenuItemIndex item,
   MenuItemAttributes *outAttributes
);
```

Parameters

menu

The menu.

item

The index of the menu item.

outAttributes

On exit, contains the attributes of the menu item. See "Menu Item Attribute Constants" (page 139) for a list of possible values.

Return Value
A result code. See "Menu Manager Result Codes" (page 161).

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

Declared In
`Menus.h`

GetMenuItemCommandID

Obtains a menu item's command ID.

```
OSErr GetMenuItemCommandID (
   MenuRef inMenu,
   MenuItemIndex inItem,
   MenuCommand *outCommandID
);
```

Parameters

inMenu

The menu that contains the menu item whose command ID you want to obtain.

inItem

The menu index of the item.

outCommandID

Pass a pointer to an unsigned 32-bit integer value. On return, the value is set to the item's command ID.

Return Value
A result code. See "Menu Manager Result Codes" (page 161).

Discussion
After a successful call to `MenuSelect`, `MenuEvent` (page 84) , or `MenuKey` (page 180), call the `GetMenuItemCommandID` function to get a menu item's command ID. You can use a menu item's command ID as a position-independent method of signalling a specific action in an application.

See also the function `SetMenuItemCommandID` (page 98).

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

Declared In
`Menus.h`

GetMenuItemCommandKey

Gets the keyboard equivalent of a menu item.

```
OSStatus GetMenuItemCommandKey (
   MenuRef inMenu,
   MenuItemIndex inItem,
   Boolean inGetVirtualKey,
   UInt16 *outKey
);
```

Parameters

inMenu

The menu containing the item.

inItem

The item whose keyboard equivalent you want to retrieve.

inGetVirtualKey

Indicates whether to retrieve the item's character code (`false`) or virtual keycode equivalent (`true`).

outKey

On exit, the keyboard equivalent of the item.

Return Value

A result code. See "Menu Manager Result Codes" (page 161).

Discussion

A menu item's keyboard equivalent may be either a character code or a virtual keycode. An item's character code and virtual keycode are stored separately and may contain different values, but only one is used by the Menu Manager at any given time. When requesting a menu item's virtual keycode equivalent, you should first check that the item is using a virtual keycode by testing the `kMenuItemAttrUseVirtualKey` attribute for that item. If this attribute is not set, the item's virtual keycode is ignored by the Menu Manager. Note that zero is a valid virtual keycode, so you cannot test the returned keycode against zero to determine if the item is using a virtual keycode equivalent. You must test the `kMenuItemAttrUseVirtualKey` attribute.

Availability

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

Declared In

`Menus.h`

GetMenuItemFontID

Obtains a menu item's font ID.

```
OSErr GetMenuItemFontID (
   MenuRef inMenu,
   MenuItemIndex inItem,
   SInt16 *outFontID
);
```

Parameters

inMenu

The menu that contains the menu item for which you wish to get a font ID.

inItem

The menu index of the item.

outFontID

Pass a pointer to a signed 16-bit integer value. On return, the value is set to the font ID for the menu item.

Return Value
A result code. See "Menu Manager Result Codes" (page 161).

Discussion
See also the function `SetMenuItemFontID` (page 101).

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

Declared In
`Menus.h`

GetMenuItemHierarchicalMenu

Returns the submenu attached to a menu item.

```
OSStatus GetMenuItemHierarchicalMenu (
   MenuRef inMenu,
   MenuItemIndex inItem,
   MenuRef *outHierMenu
);
```

Parameters

inMenu

The parent menu.

inItem

The parent item.

outHierMenu

On exit, the item's submenu, or `NULL` if it does not have one.

Return Value
A result code. See "Menu Manager Result Codes" (page 161).

Discussion
`GetMenuItemHierarchicalMenu` will return the submenu attached to a menu item regardless of how the submenu was specified. If the submenu was specified by menu ID (using `SetItemCmd` or `SetMenuItemHierarchicalID`), `GetMenuItemHierarchicalMenu` will return the currently installed menu with that ID, if any. The only case where `GetMenuItemHierarchicalMenu` will fail to return the item's submenu is when the submenu is specified by menu ID, but the submenu is not currently inserted in the menu bar.

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

Declared In
`Menus.h`

GetMenuItemIconHandle

Obtains a handle to a menu item's icon.

```
OSErr GetMenuItemIconHandle (
   MenuRef inMenu,
   MenuItemIndex inItem,
   UInt8 *outIconType,
   Handle *outIconHandle
);
```

Parameters

inMenu

The menu that contains the menu item for which you wish to obtain the handle.

inItem

The menu index of the item.

outIconType

Pass a pointer to an unsigned 8-bit value. On return, the value specifies the type of icon (`'ICON'`, `'cicn'`, `'SICN'`, icon suite, `CGImageRef`, or `IconRef`) for which you are obtaining a handle. If the menu item has no icon attached, this parameter will contain `kMenuNoIcon`. See "Menu Item Icon Type Constants" (page 156) for descriptions of possible values.

outIconHandle

Pass a pointer to a handle. On return, `outIconHandle` contains a handle to the icon that is attached to the menu item. If the menu item has no icon attached, this parameter contains `NULL`.

Return Value

A result code. See "Menu Manager Result Codes" (page 161).

Discussion

The `GetMenuItemIconHandle` function gets the icon handle and type of icon of the specified menu item. If you wish to get a resource-based menu item icon, call `GetItemIcon`.

See also the function `SetMenuItemIconHandle` (page 102).

Availability

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

Declared In

`Menus.h`

GetMenuItemIndent

Gets the indent level of a menu item.

```
OSStatus GetMenuItemIndent (
   MenuRef inMenu,
   MenuItemIndex inItem,
   UInt32 *outIndent
);
```

Parameters

inMenu

The menu containing the item.

inItem

The item whose indent level you want to retrieve.

outIndent

On exit, the indent level of the item.

Return Value
A result code. See "Menu Manager Result Codes" (page 161).

Discussion
The indent level of an item is an amount of extra space added to the left of the item's icon or checkmark. The level is simply a number, starting at zero, which the Menu Manager multiplies by a constant to get the indent in pixels. The default indent level is zero.

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

Declared In
`Menus.h`

GetMenuItemKeyGlyph

Obtains the keyboard glyph for a menu item's keyboard equivalent.

```
OSErr GetMenuItemKeyGlyph (
   MenuRef inMenu,
   MenuItemIndex inItem,
   SInt16 *outGlyph
);
```

Parameters

inMenu

The menu that contains the menu item for which you wish to get the keyboard glyph.

inItem

The menu index of the item.

outGlyph

A pointer to a signed 16-bit integer value. On return the value is set to the modifier key glyph. For a description of available keyboard glyphs, see "Menu Glyph Constants" (page 144).

Return Value
A result code. See "Menu Manager Result Codes" (page 161).

Discussion
See also the function `SetMenuItemKeyGlyph` (page 104).

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

Declared In
`Menus.h`

GetMenuItemModifiers

Obtains the modifier keys that must be pressed with a character key to select a particular menu item.

```
OSErr GetMenuItemModifiers (
   MenuRef inMenu,
   MenuItemIndex inItem,
   UInt8 *outModifiers
);
```

Parameters

inMenu

The menu that contains the menu item for which you wish to get the modifier key(s).

inItem

The menu index of the item.

outModifiers

A pointer to an unsigned 8-bit value. On return, the bits of the value are set to indicate the modifier keys that can be used in selecting the menu item; see "Modifier Key Mask Constants" (page 158).

Return Value

A result code. See "Menu Manager Result Codes" (page 161).

Discussion

See also the function `SetMenuItemModifiers` (page 105).

Availability

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

Declared In

`Menus.h`

GetMenuItemProperty

Obtains a piece of data that has been previously associated with a menu item.

```
OSStatus GetMenuItemProperty (
   MenuRef menu,
   MenuItemIndex item,
   OSType propertyCreator,
   OSType propertyTag,
   ByteCount bufferSize,
   ByteCount *actualSize,
   void *propertyBuffer
);
```

Parameters

menu

The menu containing the item to be examined for associated data.

item

The index number of the menu item or 0 if the data is associated with the menu as a whole.

propertyCreator

A four-character code. Pass your program's signature (also called a creator), as registered through Apple Developer Technical Support. If your program is of a type that would not normally have a signature (for example, a plug-in), you should still register and use a signature in this case, even though your program's file may not have the same creator code as the signature that you register. The `'macs'` property signature is reserved for the system and may not be used.

propertyTag

A four-character code. Pass the application-defined code identifying the data.

bufferSize

The size of the data to be obtained. If this is unknown, use the function `GetMenuItemPropertySize` (page 66) to get the data's size. If the size specified in the `bufferSize` parameter does not match the actual size of the property, `GetMenuItemProperty` only retrieves data up to the size specified or up to the actual size of the property, whichever is smaller, and an error is returned.

actualSize

On input, a pointer to an unsigned 32-bit integer. On return, this value is set to the actual size of the associated data. You may pass `NULL` for the `actualSize` parameter if you are not interested in this information.

propertyBuffer

On input, a pointer to a buffer. On return, this buffer contains a copy of the data that is associated with the specified menu item.

Return Value
A result code. See "Menu Manager Result Codes" (page 161).

Discussion
You may use the function `GetMenuItemProperty` to obtain a copy of data previously set with the function `SetMenuItemProperty` (page 105).

See also the `RemoveMenuItemProperty` (page 90) function.

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

Related Sample Code
HID Explorer
QTCarbonShell

Declared In
`Menus.h`

GetMenuItemPropertyAttributes

Gets the attributes of a menu item property.

```
OSStatus GetMenuItemPropertyAttributes (
   MenuRef menu,
   MenuItemIndex item,
   OSType propertyCreator,
   OSType propertyTag,
   OptionBits *attributes
);
```

Parameters

menu

The menu containing the item whose properties you want to obtain.

item

The menu index of the item.

propertyCreator

The creator code of the property.

propertyTag

The property tag.

attributes

On exit, contains the attributes of the property.

Return Value

A result code. See "Menu Manager Result Codes" (page 161).

Discussion

Currently the only attribute you can receive from this function is the `kMenuPropertyPersistent` attribute (See "Menu Item Property Attribute Constant" (page 157)), which currently has no effect on Mac OS X. Therefore, `GetMenuItemPropertyAttributes` is not useful at this time.

Availability

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

Declared In

`Menus.h`

GetMenuItemPropertySize

Obtains the size of a piece of data that has been previously associated with a menu item.

```
OSStatus GetMenuItemPropertySize (
   MenuRef menu,
   MenuItemIndex item,
   OSType propertyCreator,
   OSType propertyTag,
   ByteCount *size
);
```

Parameters

menu

The menu containing the item to be examined for associated data.

item

The index number of the menu item or 0 if the data is associated with the menu as a whole.

propertyCreator

A four-character code. Pass your program's signature, as registered through Apple Developer Technical Support. If your program is of a type that would not normally have a signature (for example, a plug-in), you should still register and use a signature in this case, even though your program's file may not have the same creator code as the signature that you register. The `'macs'` property signature is reserved for the system and may not be used.

propertyTag

A four-character code. Pass the application-defined code identifying the data.

size

On input, a pointer to an unsigned 32-bit integer. On return, this value is set to the actual size of the data.

Return Value
A result code. See "Menu Manager Result Codes" (page 161).

Discussion
If you want to retrieve a piece of associated data with the function `GetMenuItemProperty` (page 64) , you will typically need to use the `GetMenuItemPropertySize` function beforehand to determine the size of the associated data.

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

Not available to 64-bit applications.

Declared In
`Menus.h`

GetMenuItemRefCon

Obtains application-specific information for a menu item.

```
OSErr GetMenuItemRefCon (
   MenuRef inMenu,
   MenuItemIndex inItem,
   URefCon *outRefCon
);
```

Parameters

inMenu

The menu that contains the menu item for which you wish to get information.

inItem

The menu index of the item. In CarbonLib 1.6 and later and Mac OS X v10.2 and later, you may pass zero to obtain the reference constant for the menu itself.

outRefCon

A pointer to an unsigned 32-bit integer value. On return, the value is set to the reference constant associated with the menu item.

Return Value
A result code. See "Menu Manager Result Codes" (page 161).

Discussion
If you have assigned any data to a menu item using `SetMenuItemRefCon` (page 106) function, you can read it using the `GetMenuItemRefCon` function.

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

Declared In
`Menus.h`

GetMenuRef

Obtains a menu reference corresponding to a menu ID.

```
MenuRef GetMenuHandle (
    MenuID menuID
);
#define GetMenuRef GetMenuHandle
```

Discussion
This is simply a redefinition of the `GetMenuHandle` (page 57) function.

Declared In
`Menus.h`

GetMenuTitleIcon

Retrieves the icon, if any, being used as the title of a menu.

```
OSStatus GetMenuTitleIcon (
   MenuRef inMenu,
   UInt32 *outType,
   void **outIcon
);
```

Parameters

inMenu
The menu whose icon title to retrieve.

outType
On exit, contains the type of icon being used as the title of the menu. Contains `kMenuNoIcon` if the menu does not have an icon title.

outIcon
On exit, contains the icon reference, icon suite reference, or Core Graphics image reference of the icon being used as the title of the menu. If the menu does not have an icon title, this parameter is set to `NULL`.

Return Value
A result code. See "Menu Manager Result Codes" (page 161).

Discussion
This function does not increment the reference count of the returned icon, so the caller should not attempt to release it.

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

Declared In
Menus.h

GetMenuTrackingData

Gets information about the menu currently selected by the user.

```
OSStatus GetMenuTrackingData (
   MenuRef theMenu,
   MenuTrackingData *outData
);
```

Parameters

menu

The menu about which to get tracking information. Pass NULL to get information about the most recently opened menu; for example, if the user has selected a menu that contains a submenu, and the submenu is open, then GetMenuTrackingData returns the submenu not its parent menu.

outData

On exit, contains tracking data about the menu.

Return Value
A result code. See "Menu Manager Result Codes" (page 161). If the menu is not currently open, menuNotFoundErr is returned.

Discussion
You can call this function only during menu tracking. As the standard menu definition automatically handles tracking, you would probably need this function only when writing a custom menu definition.

This function replaces direct access to the pre-Carbon low-memory globals TopMenuItem, AtMenuBottom, MenuDisable, and mbSaveLoc. See the Carbon Porting Notes for MenuDefProcPtr (page 112) for more information.

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

Declared In
Menus.h

GetMenuType

Gets the display type (pulldown, hierarchical, or popup) of a menu.

```
OSStatus GetMenuType (
   MenuRef theMenu,
   UInt16 *outType
);
```

Parameters

theMenu

The menu whose type to get.

outType

On exit, the type of the menu. The returned value will be one of the Appearance Manager `ThemeMenuType` constants: `kThemeMenuTypePullDown`, `kThemeMenuTypePopUp`, or `kThemeMenuTypeHierarchical`. The `kThemeMenuTypeInactive` bit is never set.

Return Value
A result code. See "Menu Manager Result Codes" (page 161).

Discussion
You can call this function only when the menu is displayed. If the menu is not currently open, an error is returned. The display type of a menu may vary from one menu tracking session to another; for example, the same menu might be displayed as a pulldown menu and as a popup menu.

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

Declared In
`Menus.h`

GetMenuWidth

Obtains the width of the menu, in pixels.

```
SInt16 GetMenuWidth (
   MenuRef menu
);
```

Parameters

menu

The menu whose width you want to obtain.

Return Value
The width of the menu, in pixels.

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

Declared In
`Menus.h`

GetNewMBar

Reads in the definition of a menu bar from an '`MBAR`' resource.

Not recommended

```
MenuBarHandle GetNewMBar (
   short menuBarID
);
```

Parameters

menuBarID

The resource ID of an 'MBAR' resource that specifies the menus for a menu bar.

Return Value

A handle to the menu list. (If the resource isn't already in memory, GetNewMBar reads it into memory.) If GetNewMBar can't read the resource, GetNewMBar returns NULL. See the description of the MenuBarHandle data type.

Discussion

Unless you must support legacy code, you should not use functions like GetNewMBar that rely on menus and menu bars stored as resources. Instead, you should define menus in Interface Builder, store them as nib files, and then call the Interface Builder Services functions CreateMenuFromNib, CreateMenuBarFromNib, or SetMenuBarFromNibto create them.

The GetNewMBar function reads in the definition of a menu bar and its associated menus from an 'MBAR ' resource. The 'MBAR ' resource identifies the order of menus contained in its menu bar. For each menu, it also specifies the menu's resource ID. The GetNewMBar function reads in each menu from the 'MENU ' resource with the resource ID specified in the 'MBAR ' resource.

The GetNewMBar function creates a menu list for the menu bar defined by the 'MBAR ' resource and returns a handle to the menu list. GetNewMBar uses GetMenu (page 50) to read in each individual menu.

After reading in menus from an 'MBAR ' resource, use SetMenuBar to make the menu list created by GetNewMBar the current menu list. Then use DrawMenuBar (page 41) to update the menu bar.

To release the memory occupied by a menu list, use the function DisposeMenuBar (page 41)

Special Considerations

The GetNewMBar function first saves the current menu list and then clears the current menu list and your application's menu color information table. It then creates a new menu list. Before returning a handle to the new menu list, the GetNewMBar function restores the current menu list to the previously saved menu list, but GetNewMBar does not restore the previous menu color information table. To save and then restore your application's current menu color information table, call the function GetMCInfo (page 171) before GetNewMBar and call SetMCInfo (page 188) afterward.

While you supply only the resource ID of an 'MBAR' resource to the GetNewMBar function, your application often needs to use the menu IDs defined in each of your menus' 'MENU' resources. Most Menu Manager functions require either a menu ID or a handle to a menu structure to perform operations on a specific menu. For menus in the current menu list, you can use the GetMenuHandle (page 57) function to get the handle to a menu structure for a menu with a given menu ID.

Availability

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

Declared In

Menus.h

HideMenuBar

Conceals the menu bar.

```
void HideMenuBar (
   void
);
```

Discussion
The `HideMenuBar` function makes the menu bar invisible and unselectable by the user. You can use this function to enable full-screen display; however, in Mac OS X v10.2 and later, you should call the `SetSystemUIMode` function (available in `MacApplication.h`) instead.

Note that calling this function causes the `kEventMenuBarHidden` event to be sent to the application target (if your application has registered for the event).

Also see the `ShowMenuBar` (page 110) and `IsMenuBarVisible` (page 77) functions.

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

Related Sample Code
HideMenuBar

Declared In
`Menus.h`

HiliteMenu

Highlights or unhighlights menu titles.

```
void HiliteMenu (
   MenuID menuID
);
```

Parameters

`menuID`

The menu ID of the menu whose title should be highlighted. If the menu title of the specified menu is already highlighted, `HiliteMenu` does nothing. If the menu ID is 0 or the specified menu ID isn't in the current menu list, `HiliteMenu` unhighlights whichever menu title is currently highlighted (if any).

Discussion
The `IsMenuKeyEvent` (page 80), `MenuSelect` (page 86), `MenuEvent` (page 84), and `MenuKey` (page 180) functions highlight the title of the menu containing the item chosen by the user. After performing the chosen task, your application should unhighlight the menu title by calling `HiliteMenu` and passing 0 in the `menuID` parameter.

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

Related Sample Code
Simple DrawSprocket

Declared In
Menus.h

InsertMenu

Inserts an existing menu into the current menu list.

```
void InsertMenu (
    MenuRef theMenu,
    MenuID beforeID
);
```

Parameters

theMenu

The menu to insert.

beforeID

An integer that indicates where in the current menu list the menu should be inserted. InsertMenu inserts the menu into the current menu list before the menu whose menu ID is specified in the beforeID parameter. If the number in the beforeID parameter is 0 (or it isn't the ID of any menu in the menu list), InsertMenu adds the new menu after all others (except before the Help menu). If the menu is already in the current menu list or the menu list is already full, InsertMenu does nothing.

To insert a submenu into the current menu list, specify –1 or the equivalent constant kInsertHierarchicalMenu for the beforeID parameter. The submenus in the submenu portion of the menu list do not have to be currently associated with a hierarchical menu item; you can store submenus in the menu list and later specify that a menu item has a submenu if needed. However, note that during command key matching the Menu Manager scans all menus in the menu list for modifiers, including submenus that are not associated with any menu item.

You can also specify –1 for the beforeID parameter to insert a pop-up menu into the current menu list. However, if you use the standard pop-up control definition function, the pop-up control automatically inserts the menu into the current menu list according to the needs of the pop-up control.

Discussion
Menus inserted using this function are added to a menu list attached to the current root menu. To obtain or set this root menu, call AcquireRootMenu (page 21) and SetRootMenu (page 110) respectively.

Inserting a menu in the root menu (menu bar) increments its reference count; removing the menu decrements its reference count.

To change a menu title, call SetMenuItemTextWithCFString (page 107).

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

Related Sample Code
Simple DrawSprocket

Declared In
Menus.h

InsertMenuItemTextWithCFString

Inserts a new menu item with text from a CFString.

```
OSStatus InsertMenuItemTextWithCFString (
   MenuRef inMenu,
   CFStringRef inString,
   MenuItemIndex inAfterItem,
   MenuItemAttributes inAttributes,
   MenuCommand inCommandID
);
```

Parameters

menu

The menu in which to insert the new item.

inString

The text of the new item.

inAfterItem

The item after which to insert the new item. Pass zero to insert the item at the beginning of the menu. If the index value is greater than the number of items in the menu, the item is inserted at the end of the menu.

inAttributes

The attributes of the new item.

inCommandID

The command ID of the new item.

Return Value

A result code. See "Menu Manager Result Codes" (page 161).

Discussion

If the CFString is mutable, the Menu Manager will make its own copy of the CFString before returning from `InsertMenuItemWithCFString`. Modifying the string after calling `InsertMenuItemTextWithCFString` will have no effect on the menu item's actual text.

If the CFString is immutable, the Menu Manager increments the reference count of the string before returning.

The caller may release the string after calling `InsertMenuItemTextWithCFString`.

Availability

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

Declared In

`Menus.h`

InvalidateMenuEnabling

Requests that the menu's enable state be recalculated.

```
OSStatus InvalidateMenuEnabling (
   MenuRef inMenu
);
```

Parameters

inMenu

The menu whose enable states should be recalculated. Pass `NULL` to recalculate enable states for all menus in the root menu.

Return Value

A result code. See "Menu Manager Result Codes" (page 161).

Discussion

State changes in an application can cause changes in enable state of a menu's menu items. For example, selecting a block of text would usually require the Copy menu item to become enabled. Menu items are typically updated in response to a `kEventCommandUpdateStatus` Carbon event. However, the update event usually occurs only before an command key press or a click in the menu bar. To explicitly update the enable states for a menu, you can call `InvalidateMenuEnabling`.

Note that the Carbon Event Manager automatically requests recalculation of enable states for all top-level menus when

- it dispatches a user event
- the user focus changes
- the active window changes

so in many cases you may not need to explicitly call this function.

Availability

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

Declared In

`Menus.h`

InvalidateMenuItems

Invalidates a group of menu items so that they will be redrawn when `UpdateInvalidMenuItems` is next called.

```
OSStatus InvalidateMenuItems (
   MenuRef inMenu,
   MenuItemIndex inFirstItem,
   ItemCount inNumItems
);
```

Parameters

inMenu

The menu whose items to invalidate.

inFirstItem

The first item to invalidate.

inNumItems

The number of items to invalidate.

Return Value
A result code. See "Menu Manager Result Codes" (page 161).

Discussion
Menu items are automatically invalidated when their contents are changed using Menu Manager APIs while the menu is open. However, you might need to use this function if you have a custom menu definition that draws based on information not contained in the menu. For example, say you have a custom menu definition that draws a menu item using a color stored elsewhere in an application-defined data structure. If you change the color in the data structure, there is no way for the menu definition to know this has occurred. Calling `InvalidateMenuItems` after changing the color in the structure indicates that the menu should be redrawn.

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

Declared In
`Menus.h`

InvalidateMenuSize

Invalidates the menu size so that it will be recalculated when next displayed.

```
OSStatus InvalidateMenuSize (
   MenuRef inMenu
);
```

Parameters

inMenu

The menu whose size you want to invalidate.

Return Value
A result code. See "Menu Manager Result Codes" (page 161).

Discussion
Prior to Carbon, the technique for invalidating the menu size was to set the width and height to -1. Although this technique still works, for best compatibility you should call `InvalidateMenuSize` so that the Menu Manager has explicit notification that the menu is invalid.

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

Declared In
`Menus.h`

InvalMenuBar

Invalidates the menu bar.

```
void InvalMenuBar (
   void
);
```

Discussion
The `InvalMenuBar` function marks the menu bar as changed and in need of updating. When the Carbon Event Manager checks incoming events for regions that require updating, the Carbon Event Manager also checks to determine whether the menu bar requires updating (because of a call to `InvalMenuBar`). If the menu bar needs updating, the Carbon Event Manager calls the `DrawMenuBar` (page 41) function to draw the menu bar.

You can use `InvalMenuBar` instead of `DrawMenuBar` to minimize blinking in the menu bar. For example, if you have several application-defined functions that can change the enabled state of a menu and each calls `DrawMenuBar`, you can replace the calls to `DrawMenuBar` with calls to `InvalMenuBar`. In this way the menu bar is redrawn only once instead of multiple times in quick succession. If you need to make immediate changes to the menu bar, use `DrawMenuBar`. If you want to redraw the menu bar at most once each time through your event loop, use `InvalMenuBar`. Note. however, that most Menu Manager calls that affect the menu bar call `InvalMenuBar` (page 76) before returning. so you probably won't need to call it yourself.

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

Declared In
`Menus.h`

IsMenuBarInvalid

Determines if the menubar is invalid and should be redrawn.

```
Boolean IsMenuBarInvalid (
   MenuRef rootMenu
);
```

Parameters
rootMenu
The root menu for the menubar to be examined. Pass `NULL` to check the state of the current menubar.

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

Declared In
`Menus.h`

IsMenuBarVisible

Reports whether the menu bar is currently visible.

```
Boolean IsMenuBarVisible (
   void
);
```

Return Value
Returns `true` if the menu bar is currently visible; otherwise, `false`.

Discussion
Also see the `HideMenuBar` (page 72) and `ShowMenuBar` (page 110) functions.

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

Declared In
`Menus.h`

IsMenuCommandEnabled

Determines if the menu item with a specified command ID is enabled.

```
Boolean IsMenuCommandEnabled (
   MenuRef inMenu,
   MenuCommand inCommandID
);
```

Parameters

inMenu
The menu in which to begin searching for the item. Pass `NULL` to begin searching with the root menu. The search will descend into all submenus of this menu.

inCommandID
The command ID of the menu item to examine. If more than one item has this command ID, only the first will be examined.

Discussion
If you have access to the menu item index, in most cases you should use `IsMenuItemEnabled` (page 78) instead, as that function is faster and requires no searching.

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

Declared In
`Menus.h`

IsMenuItemEnabled

Reports whether a given menu or menu item is enabled.

```
Boolean IsMenuItemEnabled (
   MenuRef menu,
   MenuItemIndex item
);
```

Parameters

menu

The menu containing the item to be examined.

item

The item number of the menu item. Pass 0 to specify the menu title and determine whether the menu as a whole is enabled.

Return Value
Returns `true` if the menu item is currently enabled; otherwise, `false`.

Discussion
Your application can use the `IsMenuItemEnabled` function to determine whether specific menu items, even those with item numbers greater than 31, are currently enabled and can therefore be selected by the user.

Note that this function ignores the enable state of the menu when returning the enable state of a menu item. For example, if you call `IsMenuItemEnabled` on an enabled item while its parent menu is disabled, the function still returns `true`.

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

Declared In
`Menus.h`

IsMenuItemIconEnabled

Reports whether a given menu item icon is enabled.

```
Boolean IsMenuItemIconEnabled (
   MenuRef menu,
   MenuItemIndex item
);
```

Parameters

menu

The menu containing the icon to be examined.

item

The item number of the menu item containing the icon.

Return Value
Returns `true` if the menu item icon is currently enabled; otherwise, `false`.

Discussion
Your application can use the `IsMenuItemIconEnabled` function to determine whether a specific menu item's icon is currently enabled or dimmed.

See also the functions `DisableMenuItemIcon` (page 39) and `EnableMenuItemIcon` (page 44).

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

Declared In
`Menus.h`

IsMenuItemInvalid

Determines if a menu item is invalid and should be redrawn.

```
Boolean IsMenuItemInvalid (
   MenuRef inMenu,
   MenuItemIndex inItem
);
```

Parameters

inMenu
The menu whose item to examine.

inItem
The item to examine.

Return Value
Returns `true` if the menu is invalid, `false` otherwise.

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

Declared In
`Menus.h`

IsMenuKeyEvent

Determines if an event corresponds to a menu command key.

```
Boolean IsMenuKeyEvent (
   MenuRef inStartMenu,
   EventRef inEvent,
   MenuEventOptions inOptions,
   MenuRef *outMenu,
   MenuItemIndex *outMenuItem
);
```

Parameters

inStartMenu
The menu to search. `IsMenuKeyEvent` searches for matching menu items in this menu and all of its submenus. Pass `NULL` to search the current menu bar contents.

inEvent
The event to match against. Non-keyboard events are ignored.

inOptions

Options controlling how to search. See "Menu Event Option Constants" (page 144) for possible options. Pass `kNilOptions` for the default behavior.

outMenu

On exit, the menu containing the matching item. Set to `NULL` if no match was made.

outMenuItem

On exit, the menu item that matched. Set to `NULL` if no match was made.

Discussion

By default, `IsMenuKeyEvent` searches the menus in the current menu bar and highlights the menu title of the menu containing the selected item.

Availability

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

Declared In

`Menus.h`

IsMenuSizeInvalid

Determines if a menu's size is invalid and should be recalculated.

```
Boolean IsMenuSizeInvalid (
   MenuRef inMenu
);
```

Parameters

inMenu

The menu whose size you want to examine.

Return Value

Set to `true` if the menu size is invalid, `false` otherwise.

Discussion

Prior to Carbon, the technique for determining if a menu's size is invalid was to check if the width or height was -1. This technique is not always reliable in Carbon due to implementation changes in the Menu Manager, so you should use `IsMenuSizeInvalid` instead.

Availability

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

Declared In

`Menus.h`

IsShowContextualMenuClick

Determines whether a particular event could invoke a contextual menu.

Not recommended

```
Boolean IsShowContextualMenuClick (
   const EventRecord *inEvent
);
```

Parameters

inEvent

A pointer to the event structure that describes the event to examine.

Return Value

If `true`, the contextual menu should be displayed; if `false`, not.

Discussion

Before calling the `IsShowContextualMenuClick` function, you should call `InitContextualMenus` (page 175). If no error is returned, you can then call `IsShowContextualMenuClick`.

Unless you must support the legacy `WaitNextEvent` event model, you should use the Carbon event–based `IsShowContextualMenuEvent` (page 82) function instead. In addition, some users may choose to use a two-button mouse with their Macintosh computers, in which case a right-click does not return `true` with `IsShowContextualMenuClick` unless the mouse manufacturer's driver software deliberately returns Control-left-click in place of the right-click. If you want to properly recognize the right-click to invoke a contextual menu, you should use the `IsShowContextualMenuEvent` (page 82) function.

Applications should call `IsShowContextualMenuClick` when they receive non-null events. If `IsShowContextualMenuClick` returns `true`, your application should generate its own menu and Apple Event descriptor (`AEDesc`), and then call `ContextualMenuSelect` (page 28) to display and track the contextual menu, and then handle the user's choice.

If the mouse-down event did not invoke a contextual menu, then the application should check to see if the event occurred in the menu bar (using the `FindWindow` function) and, if so, call `MenuSelect` to allow the user to choose a command from the menu bar.

See also "Contextual Menu Gestalt Selector Constants" (page 131).

Availability

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

Declared In

`Menus.h`

IsShowContextualMenuEvent

Determines whether a particular Carbon event could invoke a contextual menu.

```
Boolean IsShowContextualMenuEvent (
   EventRef inEvent
);
```

Parameters

inEvent

The event to examine.

Return Value

Returns `true` if the application should display a contextual menu, `false` otherwise.

Discussion
If your application supports Carbon events, you should use this function in place of the older `IsShowContextualMenuClick` (page 81). This function also properly supports right-click activation of contextual menus, while `IsShowContextualMenuClick` (page 81) does not.

However, if your application uses the standard window handler, you probably don't need to use `IsShowContextualMenuEvent`, because the standard window handler automatically detects contextual menu clicks and sends the `kEventWindowContextualMenuSelect` and `kEventControlContextualMenuClick` events.

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

Declared In
`Menus.h`

IsValidMenu

Determines if a menu is valid.

```
Boolean IsValidMenu (
   MenuRef inMenu
);
```

Parameters

inMenu
The menu to check for validity.

Return Value
Indicates whether the menu is valid.

Discussion
The Menu Manager keeps a table that maps menu references to their corresponding (opaque) `MenuData` structures. A menu is considered valid if its menu reference appears in that table.

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

Declared In
`Menus.h`

LMGetTheMenu

Returns the menu ID of the currently highlighted menu in the menu bar.

```
MenuID LMGetTheMenu (
   void
);
```

Return Value
A value which contains the menu ID of the currently highlighted menu in the menu bar.

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

Declared In
`Menus.h`

MenuChoice

Returns the menu ID and index of the menu item under the cursor.

```
SInt32 MenuChoice (
   void
);
```

Return Value
The high-order word of the function result contains the menu ID of the menu, and the low-order word contains the item number of the menu item chosen by the user. The `MenuChoice` function returns 0 as the low-order word of its function result if the mouse button was released while the cursor was in the menu bar or outside the menu.

Discussion
You can use this function in the rare cases that you want to take action based upon the selection of a disabled menu item. For example, say the user is in contextual help mode (activated by pressing the Help key) and wants to get help information about a disabled menu item. You can use `MenuChoice` to determine which item the user selected and provide help for that item.

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

Declared In
`Menus.h`

MenuEvent

Maps a keyboard combination from the event structure to the keyboard equivalent of a menu item in a menu in the current menu list.

Not recommended

```
UInt32 MenuEvent (
   const EventRecord *inEvent
);
```

Parameters
`inEvent`
A pointer to the event structure containing the keyboard equivalent.

Return Value
A value that indicates the menu ID (in the high 16 bits) and menu item (in the low 16 bits) that the user chose. If the given character does not map to an enabled menu item in the current menu list, `MenuEvent` returns 0 in its high-order word and the low-order word is undefined.

Discussion
Note that unless your application uses the `WaitNextEvent` event model, you should use the Carbon event–based `IsMenuKeyEvent` (page 80) instead.

The `MenuEvent` function does not distinguish between uppercase and lowercase letters. This allows a user to invoke a keyboard equivalent command, such as the Copy command, by pressing the Command key and "c" or "C". For consistency between applications, you should define the keyboard equivalents of your commands so that they appear in uppercase in your menus.

If the given character maps to an enabled menu item in the current menu list, `MenuEvent` highlights the menu title of the chosen menu, returns the menu ID in the high-order word of its function result, and returns the chosen menu item in the low-order word of its function result. After performing the chosen task, your application should unhighlight the menu title using the `HiliteMenu` function.

You should not define menu items with identical keyboard equivalents. The `MenuEvent` function scans the menus from right to left and the items from top to bottom. If you have defined more than one menu item with identical keyboard equivalents, `MenuEvent` returns the first one it finds.

The `MenuEvent` function first searches the regular portion of the current menu list for a menu item with a keyboard equivalent matching the given key. If it doesn't find one there, it searches the submenu portion of the current menu list. If the given key maps to a menu item in a submenu, `MenuEvent` highlights the menu title in the menu bar that the user would normally pull down to begin traversing to the submenu. Your application should perform the desired command and then unhighlight the menu title.

Note that some keyboard equivalents are reserved for use by the system, such as Command–Shift–3 annd Command-Shift-4 (for taking screen shots).

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

Declared In
`Menus.h`

MenuHasEnabledItems

Determines if any items in a menu are enabled.

```
Boolean MenuHasEnabledItems (
   MenuRef theMenu
);
```

Parameters

theMenu
The menu whose items to examine.

Return Value
Returns `true` if one or more items in the menu are enabled, `false` otherwise.

Discussion
This function is equivalent to pre-Carbon code that compared the `enableFlags` field of the `MenuInfo` structure (now opaque in Carbon) with 0. It checks the enable state of all items to see if any are enabled, but ignores the state of the menu title. It will return `true` even if the menu title is disabled.

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

Declared In
`Menus.h`

MenuSelect

Allows the user to choose a menu item from a menu in the menu bar.

```
SInt32 MenuSelect (
   Point startPt
);
```

Parameters

`startPt`

The location of the cursor at the time the mouse button was first pressed, in global coordinates. Your application retrieves this point from the `kEventParamMouseLocation` parameter of a Carbon event, or the `where` field of the `EventRecord` structure.

Return Value
If the user chooses an enabled menu item (including any item from a submenu), the `MenuSelect` function returns a value as its function result that indicates which menu and menu item the user chose. The high-order word of the function result contains the menu ID of the menu, and the low-order word contains the item number of the menu item chosen by the user.

Discussion
If your application uses Carbon events, you probably don't need to use this function. The standard event handler installed by the Carbon Event Manager function `RunApplicationEventLoop` calls `MenuSelect` for you to begin tracking when the user clicks in a menu. If you do not call `RunApplicationEventLoop`, however, you will need to use this function to initiate menu tracking.

When the user presses the mouse button while the cursor is in the menu bar, your application receives a mouse-down event. To handle mouse-down events in the menu bar, pass the location of the cursor at the time of the mouse-down event as the `startPt` parameter to `MenuSelect`. The `MenuSelect` function displays and removes menus as the user moves the cursor over menu titles in the menu bar, and it handles all user interaction until the user dismisses the menu.

As the user drags the cursor through the menu bar, the `MenuSelect` function highlights the title of the menu the cursor is currently over and displays all items in that menu. If the user moves the cursor so that it is over a different menu, the `MenuSelect` function removes the previous menu and unhighlights its menu title.

The `MenuSelect` function highlights and unhighlights menu items as the user drags the cursor over the items in a menu. The `MenuSelect` function highlights a menu item if the item is enabled and the cursor is currently over it; it removes such highlighting when the user moves the cursor to another menu item. The `MenuSelect` function does not highlight disabled menu items.

If the user chooses an enabled menu item (including any item from a submenu), the `MenuSelect` function returns a value as its function result that indicates which menu and menu item the user chose. The high-order word of the function result contains the menu ID of the menu, and the low-order word contains the item number of the menu item chosen by the user.The `MenuSelect` function leaves the menu title highlighted; after performing the chosen task your application should unhighlight the menu title using the `HiliteMenu` (page 72) function.

If the user chooses an item from a submenu, `MenuSelect` returns the menu ID of the submenu in the high-order word and the item chosen by the user in the low-order word of its function result. The `MenuSelect` function also highlights the title of the menu in the menu bar that the user originally displayed in order to begin traversing to the submenu. After performing the chosen task, your application should unhighlight the menu title.

If the user releases the mouse button while the cursor is over a disabled item, in the menu bar, or outside of any menu, the `MenuSelect` function returns 0 in the high-order word of its function result and the low-order word is undefined. If it is necessary for your application to find the item number of the disabled item, your application can call `MenuChoice` (page 84) to return the menu ID and menu item.

Note that `MenuSelect` sends a number of Carbon events during menu tracking:

- `kEventMenuBeginTracking` when tracking begins, before displaying any menus. An event handler for this event may return `userCanceledErr` to prevent menu tracking from occurring.
- `kEventMenuEndTracking` after tracking was completed.
- `kEventMenuChangeTrackingMode` when the user switches from the mouse to the keyboard to navigate through menus, or vice versa.
- `kEventMenuOpening` is sent to each menu that the user opens, just before it opens. An event handler for this event may return `userCanceledErr` to prevent menu tracking from occurring.
- `kEventMenuClosed` is sent to each menu that the user closes.
- `kEventMenuPopulate` is sent to a menu just before it is opened in a menu tracking session. This event is sent only once per menu tracking session.
- `kEventMenuEnableItems` is sent to a menu just before it is opened in a menu tracking session. This event is sent only once per menu tracking session.
- `kEventCommandUpdateStatus` is sent to a menu just before it is opened in a menu tracking session. This event is sent only once per menu tracking session.
- `kEventMenuTargetItem` is sent to a menu when the mouse passes over a menu item or if the keyboard is used to select a menu item. This event is sent for both enabled and disabled menu items, and is also sent when the mouse is over a menu title.
- `kEventCommandProcess` is sent to a menu when the user chooses one of its menu items.

For more details about these Carbon events, see the *Carbon Event Manager Reference*.

Carbon Porting Notes

Carbon does not support desk accessories, so `MenuSelect` cannot be used in Mac OS X to pass keyboard equivalents to desk accessories.

Availability

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

Related Sample Code

Simple DrawSprocket

Declared In

`Menus.h`

PopUpMenuSelect

Displays a pop-up menu without using the standard pop-up control definition function.

```
SInt32 PopUpMenuSelect (
   MenuRef menu,
   short top,
   short left,
   MenuItemIndex popUpItem
);
```

Parameters

menu

The pop-up menu to be displayed.

top

The y-coordinate of the top-left corner of the selected menu item when the menu opens. This value should be in global coordinates.

left

The x-coordinate of the top-left corner of the selected menu item when the menu opens. This value should be in global coordinates.

popUpItem

The index of the menu item to display at the global point (`top`, `left`). If you pass zero, the first menu item is positioned at the indicated point.

Return Value

The menu ID of the chosen menu in the high 16-bit word of the function result and the chosen menu item index value in the low 16-bit word. If no item was selected, the result is zero.

Discussion

If your application uses the standard pop-up control definition function, your application does not need to use the `PopUpMenuSelect` function. `PopUpMenuSelect` uses the location specified by the `top` and `left` parameters to determine where to display the specified item of the pop-up menu. `PopUpMenuSelect` displays the pop-up menu so that the menu item specified in the `popUpItem` parameter appears highlighted at the specified location.

The `PopUpMenuSelect` function highlights and unhighlights menu items and handles all user interaction until the user releases the mouse button.

Note that `PopUpMenuSelect` sends a number of Carbon events during menu tracking (these are the same as those sent for `MenuSelect` (page 86)):

- `kEventMenuBeginTracking` when tracking begins, before displaying any menus. An event handler for this event may return `userCanceledErr` to prevent menu tracking from occurring.
- `kEventMenuEndTracking` after tracking was completed.
- `kEventMenuChangeTrackingMode` when the user switches from the mouse to the keyboard to navigate through menus, or vice versa.
- `kEventMenuOpening` is sent to each menu that the user opens, just before it opens. An event handler for this event may return `userCanceledErr` to prevent menu tracking from occurring.
- `kEventMenuClosed` is sent to each menu that the user closes.
- `kEventMenuPopulate` is sent to a menu just before it is opened in a menu tracking session. This event is sent only once per menu tracking session.

- `kEventMenuEnableItems` is sent to a menu just before it is opened in a menu tracking session. This event is sent only once per menu tracking session.
- `kEventCommandUpdateStatus` is sent to a menu just before it is opened in a menu tracking session. This event is sent only once per menu tracking session.
- `kEventMenuTargetItem` is sent to a menu when the mouse passes over a menu item or if the keyboard is used to select a menu item. This event is sent for both enabled and disabled menu items, and is also sent when the mouse is over a menu title.
- `kEventCommandProcess` is sent to a menu when the user chooses one of its menu items.

For more details about these Carbon events, see the *Carbon Event Manager Programming Guide*.

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

Declared In
`Menus.h`

RegisterMenuDefinition

Registers a binding between a resource ID and a menu definition function.

```
OSStatus RegisterMenuDefinition (
   SInt16 inResID,
   MenuDefSpecPtr inDefSpec
);
```

Parameters

`inResID`
An MDEF resource ID, as used in a `'MENU'` resource.

`inDefSpec`
Specifies the `MenuDefUPP` that should be used for menus with the given MDEF proc ID. Pass `NULL` if you want to unregister the menu definition.

Return Value
A result code. See "Menu Manager Result Codes" (page 161).

Discussion
Mac OS X does not allow you to store custom menu definitions in resources. However, some older functions such as `GetMenu` expect an `'MDEF'` resource ID to be in the `'MENU'` resource when creating menus . To work around this, you can use `RegisterMenulDefinition` to register "virtual" resource IDs for your menu definition functions. When `GetMenu` (or a similar function) attempts to access an `'MDEF'` resource, the Menu Manager uses the menu definition specified by the `MenuDefSpec` structure instead.

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

Declared In
`Menus.h`

RemoveMenuCommandProperty

Removes a property from a menu item with the specified command ID.

```
OSStatus RemoveMenuCommandProperty (
   MenuRef inMenu,
   MenuCommand inCommandID,
   OSType inPropertyCreator,
   OSType inPropertyTag
);
```

Parameters

inMenu

The menu in which to begin searching for the item. Pass `NULL` to begin searching with the root menu. The search will descend into all the submenus of this menu.

inCommandID

The command ID of the menu item whose property you want to remove. If more than one item has the same command ID, only the first will be modified.

inPropertyCreator

The four-character creator code for the application.

inPropertyTag

The four-character tag identifying the property to remove.

Return Value

A result code. See "Menu Manager Result Codes" (page 161).

Discussion

If you have access to the menu item index, in most cases you should use `RemoveMenuItemProperty` (page 90) instead, as that function is faster and requires no searching.

Availability

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

Declared In

`Menus.h`

RemoveMenuItemProperty

Removes a piece of data that has been previously associated with a menu item.

```
OSStatus RemoveMenuItemProperty (
   MenuRef menu,
   MenuItemIndex item,
   OSType propertyCreator,
   OSType propertyTag
);
```

Parameters

menu

The menu containing the item whose associated data is to be removed.

item

The index number of the menu item or 0 if the data is associated with the menu as a whole.

propertyCreator

A four-character code. Pass your program's signature, as registered through Apple Developer Technical Support. If your program is of a type that would not normally have a signature (for example, a plug-in), you should still register and use a signature in this case, even though your program's file may not have the same creator code as the signature that you register. The `'macs'` property signature is reserved for the system and may not be used.

propertyTag

A four-character code. Pass the application-defined code identifying the associated data.

Return Value
A result code. See "Menu Manager Result Codes" (page 161).

Discussion
Your application may remove data set with the `SetMenuItemProperty` (page 105) function by calling the `RemoveMenuItemProperty` function.

When a menu is destroyed, all of its properties are removed automatically.

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

Declared In
`Menus.h`

SetItemCmd

Sets the value of the keyboard equivalent field of a menu item.

Not recommended

```
void SetItemCmd (
   MenuRef theMenu,
   MenuItemIndex item,
   CharParameter cmdChar
);
```

Parameters

theMenu

The menu containing the item.

item

The menu index of the item.

cmdChar

The value to set for the item's keyboard equivalent field. The Menu Manager uses this value to map keyboard equivalents to menu s or to define special characteristics of the menu item.

To indicate that the menu item has a submenu, specify 0x1B in the `cmdChar` parameter; specify a value of 0x1C to indicate that the item has a special text encoding; specify a value of 0x1D to indicate that the Menu Manager should reduce the item's `'ICON'` resource to the size of a small icon; and specify a value of 0x1E to indicate that the item has an `'SICN'` resource.

The values 0x01 through 0x1A, as well as 0x1F and 0x21, are reserved for use by Apple. You should not use any of these reserved values in the `cmdChar` parameter.

Discussion
You should call `SetMenuItemCommandKey` (page 99) , `SetMenuItemHierarchicalID` (page 189) , and `SetMenuItemTextEncoding` (page 191) instead of `SetItemCmd` to set a menu item's keyboard equivalent and text encoding and to indicate that a menu item has a submenu.

You usually define the keyboard equivalents and other characteristics of your menu items in the `'MENU'` resource rather than using the `SetItemCmd` function. The `SetItemCmd` function sets the value in the keyboard equivalent field of the specified menu item to the value specified by the `cmdChar` parameter (you can specify 0 if the item doesn't have a keyboard equivalent, submenu, text encoding, reduced icon, or small icon). If you specify that the item has a submenu, you should provide the menu ID of the submenu as the item's marking character. If you specify that the item has a special text encoding, you must provide the text encoding in the icon field of the menu item. If you specify that the item has an `'SICN'` or a reduced `'ICON'` resource, you must provide the icon number in the icon field of the item.

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

Declared In
`Menus.h`

SetItemMark

Sets the mark of a menu item.

```
void SetItemMark (
   MenuRef theMenu,
   MenuItemIndex item,
   CharParameter markChar
);
```

Parameters

theMenu
The menu containing the item.

item
The menu index of the item.

markChar
The mark of the menu item or its submenu (if the item has a submenu). To set the submenu associated with this menu item, specify the menu ID of the submenu in the `markChar` parameter. You can pass the character marking constants defined in the Font Manager. Pass 0 (`noMark`)if the menu item has neither mark nor submenu.

Discussion
You should call `SetMenuItemHierarchicalID` (page 189) instead of `SetItemMark` to set the menu ID of a menu item's submenu. However, you can still use `SetItemMark` to set the mark of a menu item.

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

Related Sample Code
QTCarbonShell

Declared In
Menus.h

SetItemStyle

Sets a menu item's text style.

```
void SetItemStyle (
   MenuRef theMenu,
   MenuItemIndex item,
   StyleParameter chStyle
);
```

Parameters

theMenu
The menu containing the item.

item
The menu index of the item.

chStyle
An integer representing the menu item's text style. You can choose from the following constants: normal, bold, italic, underline, outline, shadow, condense, and extend.

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

Declared In
Menus.h

SetMenuBar

Sets the current menu list to a specified menu list.

```
void SetMenuBar (
   MenuBarHandle mbar
);
```

Parameters

mbar
A handle to a menu list that specifies the menus for a menu bar. You should specify a handle returned by GetMenuBar (page 52) or GetNewMBar (page 70).

Discussion
The SetMenuBar function copies the given menu list to the current menu list. As with GetMenuBar (page 52) , SetMenuBar doesn't copy the menu structures, just the menu list (which contains handles to the menu structures).

You can use SetMenuBar to restore a menu list that you previously saved using GetMenuBar or to set the current menu list to a menu list created from a nib file or from GetNewMBar (page 70).

The SetMenuBar function sets the current menu list and calls InvalMenuBar (page 76) so the menu bar can be updated the next time through the event loop; if you want to redraw the menu bar immediately, call the DrawMenuBar (page 41) function.

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

Related Sample Code
BSDLLCTest

Declared In
`Menus.h`

SetMenuCommandMark

Locates the menu item with a specified command ID and sets its mark character.

```
OSStatus SetMenuCommandMark (
   MenuRef inMenu,
   MenuCommand inCommandID,
   UniChar inMark
);
```

Parameters

inMenu
The menu in which to begin searching for the item. Pass `NULL` to begin searching with the root menu. The search will descend into all submenus of this menu.

inCommandID
The command ID of the menu item to be modified. If more than one item has this command ID, only the first will be modified.

inMark
The new mark character. While this is a Unicode character, only the low byte is currently used as the mark character, and it is interpreted using the application text encoding.

Return Value
A result code. See "Menu Manager Result Codes" (page 161).

Discussion
If you have access to the menu item index, in most cases you should use `SetItemMark` (page 92) instead, as that function is faster and requires no searching.

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

Declared In
`Menus.h`

SetMenuCommandProperty

Sets the property data for a menu item with the specified command ID.

```
OSStatus SetMenuCommandProperty (
   MenuRef inMenu,
   MenuCommand inCommandID,
   OSType inPropertyCreator,
   OSType inPropertyTag,
   ByteCount inPropertySize,
   const void *inPropertyData
);
```

Parameters

inMenu

inCommandID

The command ID of the menu item whose property you want to set. If more than one item has the same command ID, only the first item's property is set.

inPropertyCreator

The four-character creator code for the application.

inPropertyTag

The four-character tag for the property you want to set.

inPropertySize

The side of the property data, in bytes.

inPropertyData

The property data to set.

Return Value
A result code. See "Menu Manager Result Codes" (page 161).

Discussion
If you have access to the menu item index, in most cases you should use `SetMenuItemProperty` (page 105) instead, as that function is faster and requires no searching.

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

Declared In
`Menus.h`

SetMenuDefinition

Sets the menu definition structure for a menu.

```
OSStatus SetMenuDefinition (
   MenuRef menu,
   const MenuDefSpec *defSpec
);
```

Parameters

menu

The menu whose menu definition structure you want to set.

defSpec

The new menu definition structure.

Return Value
A result code. See "Menu Manager Result Codes" (page 161).

Discussion
You can use this function to change your menu definition on-the-fly.

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

Declared In
`Menus.h`

SetMenuExcludesMarkColumn

Sets whether a menu contains space for mark characters.

Not recommended

```
OSStatus SetMenuExcludesMarkColumn (
   MenuRef menu,
   Boolean excludesMark
);
```

Parameters

menu
The menu whose width is to be set.

excludesMark
Pass `true` to specify that the menu be drawn without space for mark characters; `false` to specify that the menu be drawn in its full width, with space for mark characters.

Return Value
A result code. See "Menu Manager Result Codes" (page 161).

Discussion
Your application may use the `SetMenuExcludesMarkColumn` function to set the width of an individual menu, so that no space is provided for mark characters such as checkmarks, dashes, or notification symbols (diamonds).

The `SetMenuExcludesMarkColumn` function is only recommended for use with pop-up menus, and then only in special cases. Mac OS human interface guidelines require that all standard (menu bar) menus include space for mark characters, and pop-up menus that present user-selectable attributes or commands should also contain space for marks. If a pop-up menu does not present a list of user-selectable attributes or commands, as is the case with the Mac OS 8.5 Window Manager window proxy pop-up menus that display a standard file system path, then narrowing the menu to exclude space for marks may be appropriate.

See also the `GetMenuExcludesMarkColumn` (page 55) function.

Carbon Porting Notes

You should instead set the `kMenuExcludesMarkColumn` menu attribute using the `ChangeMenuAttributes` (page 23) function.

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

Not available to 64-bit applications.

Declared In
`Menus.h`

SetMenuFont

Sets the font to be used in a menu.

```
OSStatus SetMenuFont (
   MenuRef menu,
   SInt16 inFontID,
   UInt16 inFontSize
);
```

Parameters

menu

The menu whose font is to be set.

inFontID

The font family ID for the font to be used. Pass 0 to use the current system font. Note that this is the font to be used in the menu items, not the menu title. You cannot change the menu title font.

inFontSize

The size, in points, of the font to be used. Valid font size values range from 9 to 24 points, inclusive. Pass 0 to use the font size of the current system font.

Return Value
A result code. See "Menu Manager Result Codes" (page 161).

Discussion
Your application may use the `SetMenuFont` function to set the font for an individual menu, such as a pop-up menu. This function sets the font used by all the menu items in the menu. If you want to set the font of only a particular menu item, use the `SetMenuItemFontID` (page 101) function instead.

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

Declared In
`Menus.h`

SetMenuHeight

Set the height of a menu.

```
void SetMenuHeight (
   MenuRef menu,
   SInt16 height
);
```

Parameters

menu

The menu whose height you want to set.

height

The height of the menu, in pixels.

Discussion
Calling `CalcMenuSize` (page 22) on the menu overwrites the menu height you set with this function.

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

Declared In
`Menus.h`

SetMenuID

Assigns a menu ID to a menu.

```
void SetMenuID (
   MenuRef menu,
   MenuID menuID
);
```

Parameters

menu

The menu you want to assign an ID to.

menuID

The menu ID to assign.

Discussion
In most cases you assign an ID to a menu when you first create it, but you can use this function to change it later.

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

Declared In
`Menus.h`

SetMenuItemCommandID

Sets a menu item's command ID.

```
OSErr SetMenuItemCommandID (
   MenuRef inMenu,
   MenuItemIndex inItem,
   MenuCommand inCommandID
);
```

Parameters

inMenu

The menu that contains the menu item whose command ID you want to set.

inItem
The menu index of the item.

inCommandID
An integer representing the command ID that you wish to set.

Return Value
A result code. See "Menu Manager Result Codes" (page 161).

Discussion
You can use a menu item's command ID as a position-independent method of signalling a specific action in an application. See *Carbon Event Manager Programming Guide* for more information about command IDs.

Note that Apple reserves all command IDs that contain all lowercase letters; your application is free to use any command ID containing uppercase characters.

See also the function `GetMenuItemCommandID` (page 59).

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

Declared In
`Menus.h`

SetMenuItemCommandKey

Sets the keyboard equivalent of a menu item.

```
OSStatus SetMenuItemCommandKey (
   MenuRef inMenu,
   MenuItemIndex inItem,
   Boolean inSetVirtualKey,
   UInt16 inKey
);
```

Parameters
inMenu
The menu containing the item.

inItem
The item whose keyboard equivalent you want to set.

inSetVirtualKey
Indicates whether to set the item's character code (`false`) or virtual keycode equivalent. (`true`).

inKey
The character code or virtual keycode equivalent to set.

Return Value
A result code. See "Menu Manager Result Codes" (page 161).

Discussion
A menu item's keyboard equivalent may be either a character code or a virtual keycode. The character code is always used to draw the item's keyboard equivalent in the menu, but either may be used for keyboard equivalent matching by `MenuEvent` and `IsMenuKeyEvent`, depending on whether the `kMenuItemAttrUseVirtualKey` item attribute is set. If `SetMenuItemCommandKey` is used to set the virtual

keycode equivalent for a menu item, it also automatically sets the `kMenuItemAttrUseVirtualKey` item attribute. To make the menu item stop using the virtual keycode equivalent and use the character code equivalent instead, use `ChangeMenuItemAttributes` to clear the `kMenuItemAttrUseVirtualKey` item attribute.

Version Notes
Prior to Mac OS X v10.3, passing a character code in the range 0x1A to 0x21 (the range of command key metacharacters such as `hMenuCmd`) returned an error. In Mac OS X v10.3 and later, the Menu Manager interprets codes in this range as the ASCII character for that value, and displays the appropriate command key glyph. For example, passing `hMenuCmd` would set the keyboard equivalent to be command-Escape, because the value of `hMenuCmd` (0x1B) is the character code for the Escape character.

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

Not available to 64-bit applications.

Declared In
`Menus.h`

SetMenuItemData

Sets multiple attributes of a menu item at once.

```
OSStatus SetMenuItemData (
   MenuRef inMenu,
   MenuItemID inItem,
   Boolean inIsCommandID,
   const MenuItemDataRec *inData
);
```

Parameters

inMenu

The menu whose attributes you want to set. Note that if you pass `true` for the `inIsCommandID` parameter, you can pass `NULL` here, in which case the Menu Manager searches the root menu for the first menu that matches the specified command ID.

inItem

The menu item index or the command ID of the menu item.

inIsCommandID

A Boolean value indicating whether the value passed for the `inItem` parameter is a command ID or a menu item index. Pass `true` to indicate a command ID, `false` to indicate that it is a menu item index. If you pass `true`, the Menu Manager sets the data for the first menu item that matches the specified command ID.

inData

A pointer to a `MenuItemDataRec` structure. Before calling, you should set the `whichData` field to indicate what data you want to set and fill out those fields appropriately. For more details on the types of data you can set, see "Menu Item Data Flags" (page 152).

Return Value
A result code. See "Menu Manager Result Codes" (page 161).

Discussion
This function is often more efficient than calling individual accessor functions if you want to set multiple attributes simultaneously.

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

Declared In
`Menus.h`

SetMenuItemFontID

Sets the font for a menu item.

```
OSErr SetMenuItemFontID (
   MenuRef inMenu,
   MenuItemIndex inItem,
   SInt16 inFontID
);
```

Parameters

inMenu
The menu that contains the menu item for which you wish to set the font.

inItem
The menu index of the item.

inFontID
An integer representing the font ID that you wish to set.

Return Value
A result code. See "Menu Manager Result Codes" (page 161).

Discussion
You can use this function to set up a font menu that displays item in its appropriate font. If you want to set the font for all the items in a menu, you can use the `SetMenuFont` (page 97) function.

See also the function `GetMenuItemFontID` (page 60).

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

Declared In
`Menus.h`

SetMenuItemHierarchicalMenu

Attaches a submenu to a menu item.

```
OSStatus SetMenuItemHierarchicalMenu (
   MenuRef inMenu,
   MenuItemIndex inItem,
   MenuRef inHierMenu
);
```

Parameters

inMenu

The parent menu.

inItem

The parent item.

inHierMenu

The submenu to attach. Pass `NULL` to remove an existing submenu.

Return Value

A result code. See "Menu Manager Result Codes" (page 161).

Discussion

Using `SetMenuItemHierarchicalMenu`, it is possible to directly specify the submenu for a menu item without specifying its menu ID. It is not necessary to insert the submenu into the hierarchical portion of the menubar, and it is not necessary for the submenu to have a unique menu ID. Simply use 0 as the menu ID for the submenu, and identify selections from the menu by command ID.

The Menu Manager will increment the reference count of the submenu that you specify, and the submenu's reference count will be decremented automatically when the parent menu item is deleted or the parent menu is disposed.

Availability

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

Declared In

`Menus.h`

SetMenuItemIconHandle

Sets a menu item's icon.

```
OSErr SetMenuItemIconHandle (
   MenuRef inMenu,
   MenuItemIndex inItem,
   UInt8 inIconType,
   Handle inIconHandle
);
```

Parameters

inMenu

The menu that contains the menu item for which you wish to set an icon.

inItem

The menu index of the item.

inIconType

Pass a value representing the type of icon (`'ICON'`, `'cicn'`, `'SICN'`, icon suite, or `IconRef`) you wish to attach; see "Menu Item Icon Type Constants" (page 156) for descriptions of possible values.

inIconHandle

Pass a handle to the icon you wish to attach to a menu item.

Return Value
A result code. See "Menu Manager Result Codes" (page 161).

Discussion
The `SetMenuItemIconHandle` function sets the icon of a menu item with an icon handle instead of a resource ID. `SetMenuItemIconHandle` allows you to set icons of type `'ICON'`, `'cicn'`, `'SICN'`, `IconRef`, `CGImageRef`, as well as icon suites. To set resource-based icons for a menu item, call `SetItemIcon`.

With the exception of types `IconRef` and `CGImageRef`,disposing of the menu will not dispose of the icon handles set by this function. The Menu Manager retains `IconRef`, `CGImageRef` icons and releases them when the menu is disposed or the menu item is removed. For all other icon types, your application should dispose of the icons when you dispose of the menu, to prevent memory leaks.

See also the function `GetMenuItemIconHandle` (page 62).

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

Related Sample Code
QTCarbonShell

Declared In
`Menus.h`

SetMenuItemIndent

Sets the indent level of a menu item.

```
OSStatus SetMenuItemIndent (
   MenuRef inMenu,
   MenuItemIndex inItem,
   UInt32 inIndent
);
```

Parameters

inMenu

The menu containing the item.

inItem

The item whose indent level you want to set.

inIndent

The new indent level of the item.

Return Value
A result code. See "Menu Manager Result Codes" (page 161).

Discussion
The indent level of an item is an amount of extra space added to the left of the item's icon or checkmark. The level is simply a number, starting at zero, which the Menu Manager multiplies by a constant to get the indent in pixels. The default indent level is zero.

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

Declared In
`Menus.h`

SetMenuItemKeyGlyph

Sets the command key glyph code for a menu item.

```
OSErr SetMenuItemKeyGlyph (
   MenuRef inMenu,
   MenuItemIndex inItem,
   SInt16 inGlyph
);
```

Parameters

inMenu
The menu that contains the menu item for which you wish to substitute a keyboard glyph.

inItem
The menu index of the item.

inGlyph
An integer representing the substitute glyph to display. Pass 0 to remove an existing glyph code. For a description of available keyboard glyphs, see "Menu Glyph Constants" (page 144).

Return Value
A result code. See "Menu Manager Result Codes" (page 161).

Discussion
The `SetMenuItemKeyGlyph` function overrides the character that would normally be displayed in a menu item's keyboard equivalent with a substitute keyboard glyph. This is useful if the keyboard glyph in the font doesn't match the actual character generated. For example, you might use this function to display function keys.

In addition, the glyph code you specify is used for command key matching if the menu item does not already have a command key or virtual keycode assigned to it.

See also the function `GetMenuItemKeyGlyph` (page 63).

Version Notes
In CarbonLib 1.2 and later and Mac OS X v10.0 and later, the Menu Manager automatically draws the appropriate glyph for a menu item that has a virtual keycode assigned to it; you do not have to set both virtual keycode and the glyph.

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

Declared In
`Menus.h`

SetMenuItemModifiers

Sets the modifier key(s) that must be pressed with a character key to select a particular menu item.

```
OSErr SetMenuItemModifiers (
   MenuRef inMenu,
   MenuItemIndex inItem,
   UInt8 inModifiers
);
```

Parameters

inMenu

The menu that contains the menu item for which you wish to set the modifier key(s).

inItem

The menu index of the item.

inModifiers

A value representing the modifier key(s) to be used in selecting the menu item; see "Modifier Key Mask Constants" (page 158).

Return Value

A result code. See "Menu Manager Result Codes" (page 161).

Discussion

You can call the `SetMenuItemModifiers` function to change the modifier key(s) you can include with a character key to create your keyboard equivalent. For example, you can change Command-x to Command-Option-Shift-x. By default, the Command key is always specified; you can remove it by calling `SetMenuItemModifiers` with the `kMenuNoCommandModifier` mask constant, or (if you are using a nib file) by unchecking the appropriate command checkbox in the Interface Builder menu item inspector.

See also the function `GetMenuItemModifiers` (page 64).

Availability

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

Declared In

`Menus.h`

SetMenuItemProperty

Associates data with a menu item.

```
OSStatus SetMenuItemProperty (
   MenuRef menu,
   MenuItemIndex item,
   OSType propertyCreator,
   OSType propertyTag,
   ByteCount propertySize,
   const void *propertyData
);
```

Parameters

menu

The menu containing the item with which you wish to associate data.

item
The index number of the menu item or 0 if the data is to be associated with the menu as a whole.

propertyCreator
A four-character code. Pass your program's signature, as registered through Apple Developer Technical Support. If your program is of a type that would not normally have a signature (for example, a plug-in), you should still register and use a signature in this case, even though your program's file may not have the same creator code as the signature that you register. The `'macs'` property signature is reserved for the system and may not be used.

propertyTag
A four-character code that identifies the property to set. You define the tag your application uses to identify the data; this code is not to be confused with the file type for the data, but may coincide if you wish.

propertySize
The size of the data.

propertyData
A pointer to the data.

Return Value
A result code. See "Menu Manager Result Codes" (page 161).

Discussion
You may use the `SetMenuItemProperty` function to associate arbitrary data, tagged with an identifying code, with a menu item.

See also the `GetMenuItemProperty` (page 64) and `RemoveMenuItemProperty` (page 90) functions.

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

Related Sample Code
HID Explorer
QTCarbonShell

Declared In
`Menus.h`

SetMenuItemRefCon

Sets application-specific information for a menu item.

```
OSErr SetMenuItemRefCon (
   MenuRef inMenu,
   MenuItemIndex inItem,
   URefCon inRefCon
);
```

Parameters
inMenu
The menu that contains the menu item with which you wish to associate information.

inItem

The menu index of the item. In CarbonLib 1.6 and later and Mac OS X v10.2 and later, you may pass zero to set a reference constant for the menu itself.

inRefCon

An unsigned 32-bit integer value. Pass a reference constant to associate with the menu item.

Return Value
A result code. See "Menu Manager Result Codes" (page 161).

Discussion
If you have any data you want to associate with a menu item, you can do so using the `SetMenuItemRefCon` function. Note that you can also use `SetMenuItemProperty` (page 105), which allows more flexibility in specifying application-specific data.

See also the function `GetMenuItemRefCon` (page 67).

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

Declared In
`Menus.h`

SetMenuItemTextWithCFString

Sets the text of a menu item to the text contained in a CFString.

```
OSStatus SetMenuItemTextWithCFString (
   MenuRef inMenu,
   MenuItemIndex inItem,
   CFStringRef inString
);
```

Parameters

inMenu

The menu containing the item.

inItem

The item whose text you want to set.

inString

The CFString containing the new menu item text.

Return Value
A result code. See "Menu Manager Result Codes" (page 161).

Discussion
If the CFString is mutable, the Menu Manager will make its own copy of the CFString before returning from `SetMenuItemTextWithCFString`. Modifying the string after calling `SetMenuItemTextWithCFString` will have no effect on the item's actual text.

If the CFString is immutable, the Menu Manager increments the reference count of the string before returning.

The caller may release the string after calling `SetMenuItemTextWithCFString`.

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

Related Sample Code
BSDLLCTest

Declared In
`Menus.h`

SetMenuTitleIcon

Sets the title of a menu to be an icon.

```
OSStatus SetMenuTitleIcon (
   MenuRef inMenu,
   UInt32 inType,
   void *inIcon
);
```

Parameters

inMenu
The menu whose title to set.

inType
The type of icon being used to specify the icon title; use `kMenuNoIcon` to remove the icon from the menu title. The supported types are `kMenuIconSuiteType`, `kMenuIconRefType`, and (in Mac OS X v10.3 and later) `kMenuIconCGImageRefType`.

inIcon
The icon; this parameter must be `NULL` if `inType` is `kMenuNoIcon`. The supported icon formats are `IconSuiteRef`, `IconRef`, and (in Mac OS X v10.3 and later) `CGImageRef`.

Return Value
A result code. See "Menu Manager Result Codes" (page 161).

Discussion
The Menu Manager takes ownership of the supplied icon after this call. When a menu with an title icon is disposed, the Menu Manager will dispose the icon as well. The Menu Manager will also dispose of the current title icon when a new text or icon title is supplied for a menu. If you specify an `IconRef` or `CGImageRef`, the Menu Manager will increment its reference count, so you can release your reference to the `IconRef` without invalidating the Menu Manager's copy. The menu bar is invalidated by this call and will be redrawn at the first opportunity.

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

Declared In
`Menus.h`

SetMenuTitleWithCFString

Sets the title of a menu to the text contained in a CFString.

```
OSStatus SetMenuTitleWithCFString (
   MenuRef inMenu,
   CFStringRef inString
);
```

Parameters

inMenu

The menu whose title you want to set.

inString

The string containing the new menu title text.

Return Value

A result code. See "Menu Manager Result Codes" (page 161).

Discussion

If the string is mutable, the Menu Manager will make its own copy of the CFString before returning from `SetMenuTitleWithCFString`. Modifying the string after calling `SetMenuTitleWithCFString` will then have no effect on the menu's actual title.

If the string is immutable, the Menu Manager simply increments the string's reference count.

The caller may release the string after calling `SetMenuTitleWithCFString`.

Availability

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

Declared In

`Menus.h`

SetMenuWidth

Sets the width of a menu.

```
void SetMenuWidth (
   MenuRef menu,
   SInt16 width
);
```

Parameters

menu

The menu whose width you want to set.

width

The width of the menu, in pixels.

Discussion

Calling `CalcMenuSize` (page 22) on the menu overwrites the menu width you set with this function.

Availability

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

Declared In

`Menus.h`

SetRootMenu

Sets the menu whose contents are displayed in the menubar.

```
OSStatus SetRootMenu (
   MenuRef inMenu
);
```

Parameters

inMenu

The new root menu.

Return Value

A result code. See "Menu Manager Result Codes" (page 161).

Discussion

The root menu should contain one menu item for each top-level menu to be displayed in the menu bar. Each menu item in turn should have a submenu that was specified using `SetMenuItemHierarchicalMenu` (page 101).

You can use the `SetRootMenu` function along with the `AcquireRootMenu` (page 21) function to save and restore the contents of the menu bar. Note that calling `SetRootMenu` sets the contents of the hierarchical portion of the menu list as well as the top-level menus displayed in the menu bar. Before returning the root menu, `AcquireRootMenu` calls `InsertMenu` (page 73) with the `kInsertHierarchialMenu` option to attach to the root menu a list of the menus that are currently inserted into the hierarchical portion of the menu. `SetRootMenu` reinserts any attached hierarchical menus into the hierarchical portion of the menu list. If you pass a newly-created menu to `SetRootMenu`, the hierarchical menu list is cleared and is empty.

Calling `SetRootMenu` also implicitly retains the new root menu, and you should release it at the appropriate time by calling `ReleaseMenu` (page 184).

Availability

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

Declared In

`Menus.h`

ShowMenuBar

Displays the menu bar.

```
void ShowMenuBar (
   void
);
```

Discussion

The `ShowMenuBar` function makes the menu bar visible and selectable by the user.

Note that calling this function also causes a `kEventMenuBarShown` event to be sent to the application target (if your application has registered for the event).

See also the `HideMenuBar` (page 72) and `IsMenuBarVisible` (page 77) functions.

Availability

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

Declared In
`Menus.h`

UpdateInvalidMenuItems

Redraws the invalid items of an open menu.

```
OSStatus UpdateInvalidMenuItems (
   MenuRef inMenu
);
```

Parameters

inMenu
The menu whose menu items you want to update.

Return Value
A result code. See "Menu Manager Result Codes" (page 161).

Discussion
This function redraws menu items in an open menu that has either had some visible aspect of its menu items change (such as their enable state) or has been invalidated by a call to `InvalidateMenuItems` (page 75).

It is not necessary to use `UpdateInvalidMenuItems` if you are using Carbon's built-in support for dynamic items based on modifier key state. However, if you are modifying items dynamically using your own implementation, you should call `UpdateInvalidMenuItems` after completing your modifications for a single menu. It will redraw any items that have been marked as invalid, and clear the invalid flag for those items.

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

Declared In
`Menus.h`

UpdateStandardFontMenu

Updates a standard Font menu.

```
OSStatus UpdateStandardFontMenu (
   MenuRef menu,
   ItemCount *outHierMenuCount
);
```

Parameters

menu
The menu you want to update.

outHierMenuCount
The number of hierarchical menus attached to the standard Font menu. This value may be `NULL` if the hierarchical menus count is not useful. For example, if the only submenus in your application are those created by `CreateStandardFontMenu`, then you don't need to worry about the hierarchical menu count, as any existing submenu must be a font menu.

Return Value
A result code. See "Menu Manager Result Codes" (page 161).

Discussion
`UpdateStandardFontMenu` calls the Font Manager function `FMGetFontGeneration` to determine if the fonts have changed and returns immediately without modifying the font menu if no changes occurred. As there is little overhead if no fonts have changed, you can call this function whenever you think it may be useful; for example, when your application is activated, or whenever you receive a `kEventMenuPopulate` event for the font menu.

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

Declared In
`Menus.h`

Callbacks

MenuDefProcPtr

Defines a pointer to a custom menu definition callback function. Your menu definition callback function draws the menu items in the menu, determines which item the user chose from the menu, and calculates the menu's dimensions.

Not recommended

```
typedef void (*MenuDefProcPtr) (
    SInt16 message,
    MenuRef theMenu,
    Rect *menuRect,
    Point hitPt,
    SInt16 *whichItem
);
```

If you name your function `MyMenuDefProc`, you would declare it like this:

```
void *MyMenuDefProc (
    SInt16 message,
    MenuRef theMenu,
    Rect *menuRect,
    Point hitPt,
    SInt16 *whichItem
);
```

Parameters

message

A constant that identifies the operation the menu definition function should perform; see "Custom Menu Definition Message Constants" (page 135) and "Obsolete Menu Definition Messages" (page 137) for a description of the messages that your menu definition function can receive. Your menu definition function should not respond to any value other than these defined messages. Other messages are reserved for internal use by Apple Computer, Inc.

theMenu

The menu that the operation should affect. This menu reference may refer to a regular menu or a popup menu.

menuRect

A pointer to the rectangle (in global coordinates) in which the menu is located.

hitPt

A mouse location (in global coordinates). The Menu Manager provides information in this parameter to the menu definition function when the `kMenuFindItemsMsg` or `kMenuPopUpMsg` messages are sent.

whichItem

A pointer to a value that specifies the item number of the menu item to act upon, or a pointer to a data structure related to one or more menu items. The data pointed to depends on the message sent; see the discussion for specifics.

Discussion

Carbon does not let you store menu definitions as resources. If you need to upgrade legacy code, see the *Carbon Porting Guide* for information about converting your resource-based definitions. See the Carbon Porting Notes for this callback for additional information about changes to older messages and access to low-memory global data.

On Mac OS X v10.3 and later, all standard menus are implemented as HIViews, and custom menus can be implemnented as custom HIViews. If you want to create custom menu definitions, you should subclass the HIMenuView class. See Introducing HIView in Carbon User Experience documentation for more information.

When you define your menus, you specify the menu definition function the Menu Manager should use when managing them. You'll usually want to use the standard menu definition function for your application. However, if you need a feature not provided by the standard menu definition function (for example, if you want to include more graphics in your menus), you can choose to write your own menu definition function.

The menu definition function is responsible for drawing the contents of the menu and its menu items, determining whether the cursor is in a displayed menu, highlighting and unhighlighting menu items, and calculating a menu's dimensions. To create an instance of your custom menu, call the `CreateCustomMenu` (page 33) function.

The Menu Manager calls your menu definition function whenever it needs your definition function to perform a certain action on a specific menu. The action your menu definition function should perform depends on the value of the `message` parameter. When the Menu Manager requests your menu definition function to perform an action on a menu, it provides your function with the appropriate menu reference. This allows your function to access the data in the menu structure and to use any data in the variable data portion of the menu structure to appropriately handle the menu items.

Your menu definition function should support the following messages:

- `kMenuInitMsg`

  The Menu Manager sends this message when creating a menu to give your menu definition a chance to perform any required initialization. If an error occurs during initialization, your menu definition should return a nonzero error code in the `*whichItem` parameter. This error is then returned by the function used to create the menu.

- `kMenuDisposeMsg`

  Sent when a menu is destroyed. The Menu Manager sends this message to give your menu definition a chance to release or dispose of any related data.

- `kMenuFindItemMsg`

Sent when the Menu Manager is displaying a menu and needs to know what item is under the mouse.

The `whichItem` parameter points to a `MenuTrackingData` structure. On entry, the `menu`, `virtualMenuTop`, and `virtualMenuBottom` fields of this structure are valid. Your menu definition should determine which item, if any, contains the point passed to you in the `hitPt` parameter and fill in the `itemUnderMouse`, `itemSelected`, and `itemRect` fields. If an item is found, the menu definition should always fill in the `itemUnderMouse` and `itemRect` fields. The menu definition should only fill in the `itemSelected` field if the item is available for selection; if it is unavailable (because it is disabled, or for some other reason), the menu definition should set the `itemSelected` field to zero.

The index values placed in the `itemUnderMouse` and `itemSelected` fields should be less than or equal to the number of items returned by `CountMenuItems` (page 32) on this menu. These values should also be identical if both are nonzero. The `itemUnderMouse` field should always be nonzero if the mouse is actually over an item.

The menu definition should not highlight the found item in response to this message, as the Menu Manager will send a separate `kMenuHiliteItemMsg` to request highlighting of the item.

If the menu definition supports scrolling, it should scroll the menu during this message, and update the `virtualMenuTop` and `virtualMenuBottom` fields of the MenuTrackingData to indicate the menu's new scrolled position.

If the menu definition uses QuickDraw to draw while scrolling, it should draw into the current port.

If the menu definition uses CoreGraphics to draw while scrolling, it should use the `CGContextRef` passed in the context field of the `MDEFHiliteItemData` structure. Menu definitions must use the `ScrollMenuImage` (page 185) function, if available, to scroll the menu contents. (This function is available in CarbonLib 1.5 and later, and in Mac OS X 10.1 and later.) `ScrollMenuImage` properly supports scrolling the alpha channel in the menu's image data. Use of the QuickDraw function `ScrollRect` to scroll the menu contents will result in the alpha channel being set to 0xFF (opaque) and the menu will no longer be translucent.

The menu definition should not modify the `menu` field of the `MenuTrackingData` structure.

- `kMenuHiliteItemMsg`

  Sent when the Menu Manager is displaying a menu and needs to highlight a newly selected item.

  The `whichItem` parameter points to a `MDEFHiliteItemData` structure. The menu definition should unhighlight the item in the `previousItem` field, if non-zero, and highlight the item in the `newItem` field.

  If the menu definition is using the Appearance Manager's menu drawing APIs, you should use the `EraseMenuBackground` (page 169) function to erase the old menu contents before unhighlighting a menu item. This is necessary because the background of a menu is translucent in Aqua, and if the old highlight is not erased first, it will show through the new unhighlighted menu background.

  If the menu definition uses QuickDraw to draw, it should draw into the current port. If it uses CoreGraphics to draw, it should use the `CGContextRef` passed in the context field of the `MDEFHiliteItemData` structure.

- `kMenuDrawItemsMsg`

  Sent when the Menu Manager is displaying a menu and needs to redraw a portion of the menu. This message is used by the dynamic menu item support code in the Menu Manager; for example, if items five and six in a menu are a dynamic group, the Menu Manager will send a `kMenuDrawItemsMsg` message when the group's modifier key is pressed or released to redraw the appropriate item, but no other items in the menu.

The whichItem parameter for this message points to an `MDEFDrawItemsData` structure. The menu definition should redraw the items starting with `firstItem` and ending with `lastItem`, inclusive.

If the menu definition uses QuickDraw to draw, it should draw into the current port. If it uses CoreGraphics to draw, it should use the `CGContextRef` passed in the context field of the `MDEFHiliteItemData` structure.

- `kMenuDrawMsg`

  Sent when the Menu Manager is displaying a menu and needs to redraw the entire menu.

  The `whichItem` parameter is actually a pointer to a `MenuTrackingData` structure. On entry, the menu field of this structure is valid. The menu definition should draw the menu and, if it supports scrolling, should also fill in the `virtualMenuTop` and `virtualMenuBottom` fields of the structure to indicate the menu's initial unscrolled position; typically, `virtualMenuTop` would be set to the same value as the top coordinate of the menu bounds, and `virtualMenuBottom` would be set to `virtualMenuTop` plus the virtual height of the menu.

  If the menu definition uses QuickDraw to draw, it should draw into the current port. If it uses CoreGraphics to draw, it should use the `CGContextRef` passed in the context field of the `MDEFHiliteItemData` structure.

- `kMenuSizeMsg`

  Sent when the Menu Manager needs to determine the size of a menu.

  The menu definition should calculate the width and height of the menu and store the sizes into the menu with `SetMenuWidth` (page 109) and `SetMenuHeight` (page 97). If the `gestaltMenuMgrSendsMenuBoundsToDefProc` bit is set in the Menu Manager's Gestalt value, then the `hitPt` parameter sent with this message is the maximum width (`hitPt.h`) and height (`hitPt.v`) of the menu. The menu definition should ensure that the width and height that it places in the menu do not exceed these values. If the gestalt bit is not set, the menu definition should just use the main display device's (`GDevice`) width and height as constraints on the menu's width and height.

- `kMenuPopUpMsg`

  Sent when the Menu Manager is about to display a popup menu. The Menu Manager uses the menu definition function to support pop-up menus that are not implemented as controls. If your menu definition function supports pop-up menus, it should respond appropriately to the `kMenuPopUpMsg` message.

  The menu definition should calculate the appropriate menu bounds to contain the menu based on the requested menu location and selected item. It should write the menu's bounds into the `Rect` structure passed by the `menuRect` parameter. If the `gestaltMenuMgrSendsMenuBoundsToDefProc` bit is set in the Menu Manager's Gestalt value, then the `menuRect` parameter on entry to this message contains a constraint rectangle, in global coordinates, outside of which the popup menu should not be positioned. The menu definition should take these constraint bounds into account as it calculates the menu bounds. If the gestalt bit is not set, the menu definition should use the bounds of the display device (`GDevice`) containing the menu's top left corner as a constraint on the menu's position.

  The `hitPt` parameter is the requested location for the top left corner of the menu. The coordinates of this parameter are swapped from their normal order; `hitPt.h` contains the vertical coordinate and `hitPt.v` contains the horizontal coordinate.

  On entry, the `whichItem` parameter points at a menu item index which is requested to be the initial selection when the menu is displayed. After calculating the menu's bounds, the menu definition should write the menu's virtual top coordinate into the location pointed at by the `whichItem` parameter. If displaying the menu at the requested location does not require scrolling, the virtual top will be the same as the menu bounds top; if the menu must scroll to fit in the requested location, the virtual top may be different.

- `kMenuCalcItemMsg`

  Sent when the Menu Manager needs to know the bounds of a menu item.

  The menu definition should calculate the size of the menu item specified by the `whichItem` parameter and store the bounds in the `Rect` structure specified by the `menuRect` parameter. Some sample menu definition code provided by Apple has previously shown an implementation of this message that always sets the top left corner of the item bounds to (0,0), regardless of the item's actual position in the menu. For best future compatibility, menu definitions should begin storing an item bounds that gives the item's actual position in the menu based on the menu's current virtual top. For example, if the virtual menu top starts at 20, then the menu definition would calculate an item bounds for the first item that starts at (0,20), an item bounds for the second item that starts at (0,40), and so on. The menu definition should call `GetMenuTrackingData` (page 69)to get the menu's current virtual position, and use zero for the menu top if `GetMenuTrackingData` returns an error.

- `kMenuThemeSavvyMsg`

  Sent by the Menu Manager to determine whether the menu definition uses the Appearance Manager menu-drawing functions to draw its menu. If it does, the menu definition should return `kThemeSavvyMenuResponse` in the location pointed to by `whichItem`. If the menu definition draws its own custom content without using the Appearance Manager menu-drawing functions, it should ignore this message.

The Menu Manager defines the data type `MenuDefUPP` to identify the universal procedure pointer for an application-defined menu definition function:

```
typedef UniversalProcPtr MenuDefUPP;
```

You typically use the `NewMenuDefProc` function like this:

```
MenuDefUPP myMenuDefProc;

myMenuDefProc = NewMenuDefProc(MyMenu);
```

Carbon Porting Notes

Prior to Carbon, menu definitions needed to use several low-memory globals to communicate with the Menu Manager. These globals have all been replaced or made obsolete in Carbon, as follows:

- `MenuDisable`

  `MenuDisable` is now set automatically by the Menu Manager using the value returned in the itemUnderMouse field of the MenuTrackingData structure passed to kMenuFindItemMsg.

- `TopMenuItem, AtMenuBottom`

  `TopMenuItem` and `AtMenuBottom` are now set automatically by the Menu Manager using the values returned in the `virtualMenuTop` and `virtualMenuBottom` fields of the `MenuTrackingData` structure passed to `kMenuDrawMsg` and `kMenuFindItemMsg`.

- `mbSaveLoc`

  This undocumented low-memory global was used by older menu definitions to store the bounding rect of the currently selected item and to avoid drawing glitches while the menu definition was scrolling the contents of a menu that had submenus. The Menu Manager now automatically sets the selected item bounds using the value returned in the itemRect field of the MenuTrackingData structure passed to kMenuFindItemMsg. In order to correctly support scrolling of menus with submenus, a menu definition should verify, before scrolling the menu contents, that no submenus of the scrolling menu are currently visible. A menu definition can use `GetMenuTrackingData` to verify this condition, as follows:

```
Boolean SafeToScroll( MenuRef menuBeingScrolled )
 {
        MenuTrackingData lastMenuData;
        return GetMenuTrackingData( NULL, &lastMenuData ) ==  noErr
                            && lastMenuData.menu == menuBeingScrolled;
    }
```

If `SafeToScroll` returns `false`, the menu definition should not scroll the menu.

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

Declared In
`Menus.h`

Data Types

HMenuBarHeader

Defines a list of hierarchical menus that have been inserted into a menu bar.

```
struct HMenuBarHeader {
    UInt16 lastHMenu;
    PixMapHandle menuTitleBits;
};
typedef struct HMenuBarHeader HMenuBarHeader;
```

Fields

`lastHMenu`
Offset from the start of the header to the last menu in the array of `HMenuBar` structures.

`menuTitleBits`
The saved bits behind the highlighted menu title. This value is undefined in menu bar handles returned by `GetNewMBar` (page 70) or `GetMenuBar` (page 52).

Discussion
The hierarchical portion of the menu bar follow the nonhierarchical portion in a menu bar handle. The hierarchical section consists of the `HMenuBarHeader` structure followed by an array of `HMenuBarMenu` (page 117) structures.

You insert a hierarchical menu by specifying -1 for the `beforeID` parameter in `InsertMenu` (page 73).

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

Declared In
`Menus.h`

HMenuBarMenu

Defines a hierarchical menu.

```
struct HMenuBarMenu {
    MenuRef menu;
    SInt16 reserved;
};
typedef struct HMenuBarMenu HMenuBarMenu;
```

Fields

`menu`

The menu.

`reserved`

Currently unused.

Availability

Available in Mac OS X v10.2 and later.

Declared In

`Menus.h`

MCEntry

Specifies a menu color information table

Not recommended

```
struct MCEntry {
    MenuID mctID;
    short mctItem;
    RGBColor mctRGB1;
    RGBColor mctRGB2;
    RGBColor mctRGB3;
    RGBColor mctRGB4;
    short mctReserved;
};
typedef struct MCEntry MCEntry;
typedef MCEntry * MCEntryPtr;
typedef MCEntry MCTable[1];
```

Fields

`mctID`

Defines, along with the `mctItem` field, whether the entry is a menu bar entry, a menu title entry, or a menu item entry. The `mctID` field contains either a menu ID or 0 (for a menu bar).

`mctItem`

Defines, along with the `mctID` field, whether the entry is a menu bar entry, a menu title entry, or a menu item entry. The `mctItem` field contains either a menu item number or 0 (for a menu bar or menu title).

`mctRGB1`

Specifies color information for the entry, as follows. For a menu bar entry, this value is the default color for menu titles. For a menu title entry, this value is the title color of a specific menu. For a menu item entry, this value is the mark color for a specific item.

`mctRGB2`

Specifies color information for the entry, as follows. For a menu bar entry, this value is the default background color of a displayed menu. For a menu title entry, this value is the default color for the menu bar. For a menu item entry, this value is the color for the text of a specific item.

`mctRGB3`

Specifies color information for the entry, as follows. For a menu bar entry, this value is the default color of items in a displayed menu. For a menu title entry, this value is the default color for items in a specific menu. For a menu item entry, this value is the color for the modifier of a specific item.

`mctRGB4`

Specifies color information for the entry, as follows. For a menu bar entry, this value is the default color of the menu bar. For a menu title entry, this value is the background color of a specific menu. For a menu item entry, this value is the background color of a specific menu.

`mctReserved`

Reserved.

Discussion

The menu color information table defines the standard color for the menu bar, menu titles, menu items, and the background color of a displayed menu. If you do not add any menu color entries to this table, the Menu Manager draws your menus using the current default colors. Using the menu color information table to define custom colors for your menus is not recommended with Appearance Manager 1.0 and later.

When the Appearance Manager is available and you are using standard menus, if you do not include a menu bar entry in your menu color information table, only the menu title color and menu item text color values from menu color entries are used. If you do include a menu bar entry in your menu color information table, all menu colors are used, and the menus revert to a standard System 7 appearance.

If you are creating your own custom menu definition function, all entries in the table are used.

You can add custom colors to your menus by adding entries to your application's menu color information table, using Menu Manager functions or by defining these entries in an 'mctb' resource. Note that the menu color information table uses a format different from the standard color table format.

The value of the `mctID` field in the last entry in a menu color information table is `mctLastIDIndic`, and the rest of the fields of the last entry are reserved. The Menu Manager automatically creates the last entry in a menu color information table; your application should not use the value `mctLastIDIndic` as the menu ID of a menu if you wish to add a menu color entry for it.

The contents of a menu color table entry structure are interpreted differently, depending upon the values of the `mctID` and `mctItem` fields. Depending upon the value of these fields, the `MCEntry` structure represents a menu bar entry, a menu title entry, a menu item entry, or the last entry.

A *menu bar entry* is defined by a menu color entry structure that contains 0 in both the `mctID` and `mctItem` fields. You can define only one menu bar entry in a menu color information table. If you don't provide a menu bar entry for your application's menu color information table, the Menu Manager uses the standard menu bar colors (black text on a white background), and it uses the standard colors for the other menu elements. You can provide a menu bar entry to specify default colors for the menu title, the background of a displayed menu, the items in a menu, and the menu bar. The color information fields for a menu bar entry are interpreted as follows:

- `mctRGB1` specifies the default color for menu titles. If a menu doesn't have a menu title entry, the Menu Manager uses the value in this field as the color of the menu title.
- `mctRGB2` specifies the default color for the background of a displayed menu. If a menu doesn't have a menu title entry, the Menu Manager uses the value in this field as the color of the menu's background when it is displayed.
- `mctRGB3` specifies the default color for the items in a displayed menu. If a menu item doesn't have a menu item entry or a default color defined in a menu title entry, the Menu Manager uses the value in this field as the color of the menu item.

- `mctRGB4` specifies the default color for the menu bar. If a menu doesn't have a menu bar entry (and doesn't have any menu title entries), the Menu Manager uses the standard colors for the menu bar.

  A *menu title entry* is defined by a menu color entry structure that contains a menu ID in the `mctID` field and 0 in the `mctItem` field. You can define only one menu title entry for each menu. If you don't provide a menu title entry for a menu in your application's menu color information table, the Menu Manager uses the colors defined by the menu bar entry. If a menu bar entry doesn't exist, the Menu Manager uses the standard colors (black on white). You can provide a menu title entry to specify a color for the title and background of a specific menu and a default color for its items. The color information fields for a menu title entry are interpreted as follows:

- `mctRGB1` specifies the color for the menu title of the specified menu. If a menu doesn't have a menu title entry, the Menu Manager uses the default value defined in the menu bar entry.
- `mctRGB2` specifies the default color for the menu bar. If a menu color information table doesn't have a menu bar entry, the Menu Manager uses the value in this field as the color of the menu bar. If a menu bar entry already exists, the Menu Manager replaces the value in the mctRGB2 field of the menu title entry with the value defined in the mctRGB4 field of the menu bar entry.
- `mctRGB3` specifies the default color for the items in the menu. If a menu item doesn't have a menu item entry or a default color defined in a menu bar entry, the Menu Manager uses the value in this field as the color of the menu item.
- `mctRGB4` specifies the color for the background of the menu.

  A *menu item entry* is defined by a menu color entry structure that contains a menu ID in the `mctID` field and an item number in the `mctItem` field. You can define only one menu item entry for each menu item. If you don't provide a menu item entry for an item in your application's menu color information table, the Menu Manager uses the colors defined by the menu title entry (or by the menu bar entry if the menu containing the item doesn't have a menu title entry). If neither a menu title entry nor a menu bar entry exists, the Menu Manager draws the mark, text, and modifier in black. You can provide a menu item entry to specify a color for the mark, text, and keyboard equivalent of a specific menu item. The color information fields for a menu item entry are interpreted as follows:

- `mctRGB1` specifies the color for the mark of the menu item. If a menu item doesn't have a menu item entry, the Menu Manager uses the default value defined in the menu title entry or the menu bar entry.
- `mctRGB2` specifies the color for the text of the menu item. If a menu item doesn't have a menu item entry, the Menu Manager uses the default value defined in the menu title entry or the menu bar entry. The Menu Manager also draws a black-and-white icon of a menu item using the same color as defined by the `mctRGB2` field. (Use a `'cicn'` resource to provide a menu item with a color icon.)
- `mctRGB3` specifies the color for the modifier of the menu item. If a menu item doesn't have a menu item entry, the Menu Manager uses the default value defined in the menu title entry or the menu bar entry.
- `mctRGB4` specifies the color for the background of the menu. If the menu color information table doesn't have a menu title entry for the menu this item is in, or doesn't have a menu bar entry, the Menu Manager uses the value in this field as the background color of the menu. If a menu title entry already exists, the Menu Manager replaces the value in the `mctRGB4` field of the menu item entry with the value defined in the `mctRGB4` field of the menu title entry (or with the `mctRGB2` field of the menu bar entry).

You can use the `GetMCInfo` (page 171) function to get a copy of your application's menu color information table and the `SetMCEntries` (page 187) function to set entries of your application's menu color information table, or you can provide `'mctb'` resources that define the color entries for your menus.

The GetMenu (page 50) , GetNewMBar (page 70) , and ClearMenuBar (page 26) functions can also modify the entries in the menu color information table. The GetMenu function looks for an 'mctb ' resource with a resource ID equal to the value in the menuID parameter. If it finds one, it adds the entries to the application's menu color information table.

The GetNewMBar function builds a new menu color information table when it creates the new menu list. If you want to save the current menu color information table, call GetMCInfo before calling GetNewMBar.

The ClearMenuBar function reinitializes both the current menu list and the menu color information table.

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

Declared In
Menus.h

MDEFDrawData

Contains information needed to draw a menu.

```
struct MDEFDrawData {
    MenuTrackingData trackingData;
    void * context;
};
typedef struct MDEFDrawData MDEFDrawData;
typedef MDEFDrawData * MDEFDrawDataPtr;
```

Fields

trackingData
: A data structure containing information about the menu to be drawn. Your menu definition should fill in the virtualMenuTop and virtualMenuBottom fields of this structure while drawing the menu.

context
: The Core Graphics context that your menu definition should draw into. The Menu Manager flushes the context after returning from the menu definition.

Discussion
The Menu Manager passes this structure to your custom menu definition in the whichItem parameter of the kMenuDrawMsg message.

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

Declared In
Menus.h

MDEFDrawItemsData

Contains information about which menu items to redraw.

```
struct MDEFDrawItemsData {
    MenuItemIndex firstItem;
    MenuItemIndex lastItem;
    MenuTrackingData * trackingData;
    void * context;
};
typedef struct MDEFDrawItemsData MDEFDrawItemsData;
typedef MDEFDrawItemsData * MDEFDrawItemsDataPtr;
```

Fields

`firstItem`
: The first item to draw.

`lastItem`
: The last item to draw.

`trackingData`
: Information about the menu's tracking state. The `virtualMenuTop` and `virtualMenuBottom` fields in this structure will be the most useful in handling the `DrawItems` message.

`context`
: The Core Graphics drawing context that your menu definition should draw into. The Menu Manager flushes the context after returning from the menu definition.

Discussion

The Menu Manager passes this structure to your custom menu definition in the `whichItem` parameter of the `kMenuDrawItemsMsg` message.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`Menus.h`

MDEFFindItemData

Contains information used to determine which item the user has currently selected.

```
struct MDEFFindItemData {
    MenuTrackingData trackingData;
    void * context;
};
typedef struct MDEFFindItemData MDEFFindItemData;
typedef MDEFFindItemData * MDEFFindItemDataPtr;
```

Fields

`trackingData`
: A data structure containing information about the menu to be drawn. Your menu definition should fill in the `itemSelected`, `ItemUnderMouse`and `itemRect` fields of this structure after determining which item is under the specified point.

`context`
: The Core Graphics context the menu definition should draw into if it needs to scroll the menu during the `kMenuFindItemMsg` message. The Menu Manager flushes the context after the menu definition returns.

Discussion
The Menu Manager passes this structure to your custom menu definition in the `whichItem` parameter of the `kMenuFindItemsMsg` message.

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

Declared In
`Menus.h`

MDEFHiliteItemData

Contains information about which menu items should be highlighted and unhighlighted as the user moves through the menus. .

```
struct MDEFHiliteItemData {
    MenuItemIndex previousItem;
    MenuItemIndex newItem;
    void * context;
};
typedef struct MDEFHiliteItemData MDEFHiliteItemData;
typedef MDEFHiliteItemData * MDEFHiliteItemDataPtr;
typedef MDEFHiliteItemData HiliteMenuItemData;
typedef MDEFHiliteItemDataPtr HiliteMenuItemDataPtr;
```

Fields

`previousItem`
: The menu item that was previously selected. This item needs to be redrawn in an unhighlighted state. This parameter can be zero if no item was previously selected.

`newItem`
: The menu item that is now selected. This item needs to be redrawn in a highlighted state. This parameter can be zero if no item is currently highlighted.

`context`
: The Core Graphics context the menu definition should draw into. The Menu Manager flushes the context after the menu definition returns.

Discussion
This structure is used by menu definition functions, which receive a pointer to an `MDEFHiliteItemData` structure as the `whichItem` parameter during the `kMenuHiliteItemMsg` message.

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

Declared In
`Menus.h`

MenuBarHandle

A handle to a menu bar header.

```
typedef Handle MenuBarHandle;
```

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

Declared In
Menus.h

MenuBarHeader

Defines a list of nonhierarchical menus that that have been placed in the menu bar.

```
struct MenuBarHeader {
    UInt16 lastMenu;
    SInt16 lastRight;
    SInt16 mbResID;
};
typedef struct MenuBarHeader MenuBarHeader;
```

Fields

`lastMenu`
Offset from the start of the header to the last menu in the array of `MenuBarMenu` (page 124) structures, in bytes.

`lastRight`
The x-coordinate of the right edge of the rightmost menu, in global coordinates. This value is undefined in menu bar handles returned by `GetNewMBar` (page 70) or `GetMenuBar` (page 52).

`mbResID`
The MBDF resource ID. This value is undefined in menu bar handles returned by `GetNewMBar` (page 70) or `GetMenuBar` (page 52).

Discussion
This structure is contained within a menu bar handle (`MenuBarHandle`).

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

Declared In
Menus.h

MenuBarMenu

Defines a nonhierarchical menu.

```
struct MenuBarMenu {
    MenuRef menu;
    SInt16 menuLeft;
};
typedef struct MenuBarMenu MenuBarMenu;
```

Fields

`menu`

`menuLeft`

The x-coordinate of the left edge of the menu title, in global coordinates. This value is undefined in menu bar handles returned by `GetNewMBar` (page 70) or `GetMenuBar` (page 52).

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

Declared In
`Menus.h`

MenuCommand

Specifies a menu item's command ID.

```
typedef UInt32 MenuCommand;
```

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

Declared In
`Menus.h`

MenuCRsrc

Specifies a set of menu color table entries

Not recommended

```
struct MenuCRsrc {
    short numEntries;
    MCTable mcEntryRecs;
};
typedef struct MenuCRsrc MenuCRsrc;
typedef MenuCRsrc * MenuCRsrcPtr;
```

Fields

`numEntries`

`mcEntryRecs`

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

Declared In
`Menus.h`

MenuDefSpec

Defines the type of menu definition.

```
struct MenuDefSpec {
    MenuDefType defType
    union {
        MenuDefUPP defProc;
        struct {
                CFStringRef classID;
                EventRef initEvent;
                } view;
    } u;
};
typedef struct MenuDefSpec MenuDefSpec;
typedef MenuDefSpec * MenuDefSpecPtr;
```

Fields

`defType`

The type of menu definition. See "Menu Definition Type Constants" (page 142) for a list of possible values.

`u.defproc`

If the `defType` field is `kMenuDefProcPtr`, the menu definition is a older procedure pointer–based definition. This field then contains a UPP to the menu definition function.

`u.view.classID`

If the `defType` field is `kMenuDefClassID`, the menu definition is HIView–based. This field then contains the ID of the HIView subclass that defines this menu.

`u.view.initEvent`

If the `defType` field is `kMenuDefClassID`, the menu definition is HIView–based. This field then contains the initialization event for the HIView subclass if one exists. Otherwise, it is `NULL`.

Version Notes

HIView-based menu definitions are available in Mac OS X 10.3 and later.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`Menus.h`

MenuDefUPP

Defines a universal procedure pointer to a menu definition function.

Not recommended

```
typedef MenuDefProcPtr MenuDefUPP;
```

Discussion

For more information, see the description of the `MenuDefProcPtr` (page 112) callback.

Version Notes

In Mac OS X 10.3 and later, you should use HIView-based menu definitions instead.

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

Declared In
`Menus.h`

MenuHandle

Defines a menu reference.

```
typedef MenuRef MenuHandle;
```

Discussion
You should refer to menus using the `MenuRef` (page 130) type instead.

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

Declared In
`Menus.h`

MenuID

Defines a menu ID.

```
typedef SInt16 MenuID;
```

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

Declared In
`Menus.h`

MenuItemDataRec

Used with the `SetMenuItemData` (page 100) and `CopyMenuItemData` (page 30) functions to get or change aspects of a menu item.

```
struct MenuItemDataRec {
    MenuItemDataFlags whichData;
    StringPtr text;
    UniChar mark;
    UniChar cmdKey;
    UInt32 cmdKeyGlyph;
    UInt32 cmdKeyModifiers;
    Style style;
    Boolean enabled;
    Boolean iconEnabled;
    UInt8 filler1;
    SInt32 iconID;
    UInt32 iconType;
    Handle iconHandle;
    MenuCommand cmdID;
    TextEncoding encoding;
    MenuID submenuID;
    MenuRef submenuHandle;
    SInt32 fontID;
    UInt32 refcon;
    OptionBits attr;
    CFStringRef cfText;
    Collection properties;
    UInt32 indent;
    UInt16 cmdVirtualKey;
};
typedef struct MenuItemDataRec MenuItemDataRec;
typedef MenuItemDataRec * MenuItemDataPtr;
```

Fields

`whichData`
: The fields to be set or obtained. You pass a bit mask as specified by "Menu Item Data Flags" (page 152) to indicate which values you want to get or set. The values themselves are set or populated in the fields that follow.

`text`
: The menu item title, as an `Str255` string.

`mark`
: The menu item's mark.

`cmdKey`
: The menu item's command key. This can be either a character code or a virtual key code.

`cmdKeyGlyph`
: The menu item's command key glyph.

`cmdKeyModifiers`
: The menu item's command key modifiers.

`style`
: The menu item's QuickDraw text style.

`enabled`
: The menu item's enable state.

`iconEnabled`
: The enable state of the menu item icon.

`filler1`
: Reserved.

`iconID`
The icon resource ID of the menu item.

`iconHandle`
The icon handle of the menu item.

`cmdID`
The command ID for the menu item.

`encoding`
The text encoding of the menu item.

`submenuID`
The menu ID of the submenu associated with this menu item.

`submenuHandle`
The MenuRef of the submenu associated with this menu item.

`fontID`
The font ID for the menu item.

`refcon`
The reference constant associated with this menu item.

`attr`
The menu item's attributes.

`cfText`
The menu item's title, as a Core Foundation string.

`properties`
A collection holding the menu item's properties.

`indent`
The menu item's indent level.

`cmdVirtualKey`
The menu item's virtual key.

Discussion
When using this structure with`CopyMenuItemData` (page 30) or `SetMenuItemData` (page 100), the caller must first set the `whichData` field to a combination of `MenuItemDataFlags` indicating which specific data should be retrieved or set. Some fields also require initialization before calling `CopyMenuItemData` (page 30); see "Menu Item Data Flags" (page 152) for details.

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

Declared In
`Menus.h`

MenuItemID

Defines a menu item.

```
typedef UInt32 MenuItemID;
```

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

Declared In
Menus.h

MenuItemIndex

Specifies a particular menu item in a menu.

```
typedef UInt16 MenuItemIndex;
```

Discussion
The menu item index is one-based, so item 1 is the first menu item, item 2 is the second, and so on. Some functions allow you to pass an index of zero, which specifies the menu itself.

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

Declared In
Menus.h

MenuRef

Defines a menu.

```
typedef struct OpaqueMenuRef * MenuRef;
```

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

Declared In
Menus.h

MenuTrackingData

Contains information about the menu currently being displayed during menu tracking.

```
struct MenuTrackingData {
    MenuRef menu;
    MenuItemIndex itemSelected;
    MenuItemIndex itemUnderMouse;
    Rect itemRect;
    SInt32 virtualMenuTop;
    SInt32 virtualMenuBottom;
};
typedef struct MenuTrackingData MenuTrackingData;
typedef MenuTrackingData * MenuTrackingDataPtr;
```

Fields
menu
The menu.

`itemSelected`
: The index of the menu item that is currently selected. This field should either match the `itemUnderMouse` field, or should be zero if the item under the mouse cannot be selected (for example, if the item is disabled).

`itemUnderMouse`
: The index of the menu item that is currently under the mouse.

`itemRect`
: The `Rect` that defines the area of the menu item currently under the mouse. Note that the `itemRect` field is not supported in CarbonLib and is always set to be empty. It is, however, supported in Mac OS X.

`virtualMenuTop`
: The y-coordinate of the actual top of the menu. Because the user can scroll the menu, the menu top coordinate may be above the top of the visible screen (in which case it has a negative value).

`virtualMenuBottom`
: The y-coordinate of the actual bottom of the menu. Because the user can scroll the menu, the menu bottom coordinate may be below the bottom of the visible screen

Discussion
You can call `GetMenuTrackingData` (page 69) to obtain this structure during menu tracking.

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

Declared In
`Menus.h`

Constants

Contextual Menu Gestalt Selector Constants

Determine which contextual menu features are available.

```
enum {
    gestaltContextualMenuAttr = 'cmnu',
    gestaltContextualMenuUnusedBit = 0,
    gestaltContextualMenuTrapAvailable = 1,
    gestaltContextualMenuHasAttributeAndModifierKeys = 2,
    gestaltContextualMenuHasUnicodeSupport = 3
};
```

Constants
`gestaltContextualMenuAttr`
: The `Gestalt` selector passed to the `Gestalt` function to determine whether contextual menu functions are available. Produces a value whose bits you should test to determine whether the contextual menu functions are available.

  Available in Mac OS X v10.0 and later.

  Declared in `Menus.h`.

`gestaltContextualMenuUnusedBit`
Available in Mac OS X v10.0 and later.

Declared in `Menus.h`.

`gestaltContextualMenuTrapAvailable`
If this bit is set, the contextual menu functions are available to 68K applications. If this bit is not set, these functions are not available to 68K applications.

Available in Mac OS X v10.0 and later.

Declared in `Menus.h`.

`gestaltContextualMenuHasAttributeAndModifierKeys`
The contextual menu supports attributes and modifier keys.

Available in Mac OS X v10.2 and later.

Declared in `Menus.h`.

`gestaltContextualMenuHasUnicodeSupport`
The contextual menu supports Unicode text.

Available in Mac OS X v10.2 and later.

Declared in `Menus.h`.

Discussion
Mac OS X and all versions of Mac OS that shipped with CarbonLib support contextual menus, so you need to check only to see if certain features are available.

Contextual Menu Help Type Constants

Indicates what types of contextual menu help is available.

```
enum {
    kCMHelpItemNoHelp = 0,
    kCMHelpItemAppleGuide = 1,
    kCMHelpItemOtherHelp = 2,
    kCMHelpItemRemoveHelp = 3
};
```

Constants
`kCMHelpItemNoHelp`
The application does not support any help. The Menu Manager will put an appropriate help string into the menu and disable it.

Available in Mac OS X v10.0 and later.

Declared in `Menus.h`.

`kCMHelpItemAppleGuide`
The application supports Apple Guide help. The Menu Manager will put the name of the main Guide file into the menu and enable it.

Available in Mac OS X v10.0 and later.

Declared in `Menus.h`.

`kCMHelpItemOtherHelp`
: The application supports some other form of help. In this case, the application must also pass a valid string into the `inHelpItemString` parameter of `ContextualMenuSelect`. This string will be the text of the help item in the menu, and the help item will be enabled.

  Available in Mac OS X v10.0 and later.

  Declared in `Menus.h`.

`kCMHelpItemRemoveHelp`
: The application does not support any help. The Menu Manager will remove the Help item from the contextual menu.

  Available in CarbonLib 1.6 and Mac OS X and later. Note however, that in CarbonLib, this constant is equivalent to `kCMItemNoHelp`, which only disables the Help item.

  Available in Mac OS X v10.2 and later.

  Declared in `Menus.h`.

Discussion
You can pass these constants in the `inHelpType` parameter of the function `ContextualMenuSelect` (page 28) to specify the kind of help the application supports. Contextual menu help type constants are available with Appearance Manager 1.0 and later.

Contextual Menu Selection Type Constants

Indicates the type of item the user selected from a contextual menu.

```
enum {
    kCMNothingSelected = 0,
    kCMMenuItemSelected = 1,
    kCMShowHelpSelected = 3
};
```

Constants
`kCMNothingSelected`
: The user did not choose an item from the contextual menu and the application should do no further processing of the event.

  Available in Mac OS X v10.0 and later.

  Declared in `Menus.h`.

`kCMMenuItemSelected`
: The user chose one of the application's items from the menu. The application can examine the `outMenuID` and `outMenuItem` parameters of `ContextualMenuSelect` to see what the menu selection was, and it should then handle the selection appropriately.

  Available in Mac OS X v10.0 and later.

  Declared in `Menus.h`.

`kCMShowHelpSelected`
: The user chose the Help item from the menu. The application should open an Apple Help database to a section appropriate for the selection. If the application supports some other form of help, it should be presented instead.

  Available in Mac OS X v10.0 and later.

  Declared in `Menus.h`.

Discussion
These constants are returned in the `outUserSelectionType` parameter of the function `ContextualMenuSelect` (page 28) to specify what the user selected from the contextual menu. Contextual menu selection type constants are available with Appearance Manager 1.0 and later.

Contextual Menu Item Content Constants

Specify contents of menu items in contextual menus.

```
enum {
    keyContextualMenuName = 'pnam',
    keyContextualMenuCommandID = 'cmcd',
    keyContextualMenuSubmenu = 'cmsb',
    keyContextualMenuAttributes = 'cmat',
    keyContextualMenuModifiers = 'cmmd'
};
```

Constants

`keyContextualMenuName`
The menu item text. In Mac OS X v10.1 and earlier, the data format must be either `typeChar` or `typeIntlText`. In Mac OS X v10.2 and later, you can also specify `typeStyledText`, `typeAEText`, `typeUnicodeText`, and `typeCFStringRef`. Note that if you specify `typeCFStringRef`, the Menu Manager releases the CFString reference after displaying the menu. If you need to hold onto the CFString reference, you should retain it before inserting it into the Apple event record (`AERecord`).

Available in Mac OS X v10.2 and later.

Declared in `Menus.h`.

`keyContextualMenuCommandID`
The menu item command ID. The data format for this parameter must be `typeLongInteger`.

Available in Mac OS X v10.2 and later.

Declared in `Menus.h`.

`keyContextualMenuSubmenu`
The submenu of the menu item. You typically use this with the Apple Event Manager function `AEPutDesc` to add an entire `AEDesc` record (which contains the submenu) as the parameter data.

Available in Mac OS X v10.2 and later.

Declared in `Menus.h`.

`keyContextualMenuAttributes`
Specifies the menu item attributes. The data format for this parameter must be `typeLongInteger`.

Available in Mac OS X v10.2 and later.

Declared in `Menus.h`.

`keyContextualMenuModifiers`
Specifies the modifer keys for the menu item. The data format for this parameter must be `typeLongInteger`. By using this parameter along with the `keyContextualMenuAttributes` parameter, you can create dynamic contextual menu items that change according to the state of the modifier keys.

Declared in `Menus.h`.

Available in Mac OS X v10.2 and later.

Discussion
You use these keyword constants to specify parameters in an Apple Event record (`AERecord`) that defines a contextual menu item. Typically you assign these values in your ExamineContext method of a contextual menu plugin.

Custom Menu Definition Message Constants

Indicate messages used for non-HIView–based custom menu definitions.

```
enum {
    kMenuDrawMsg = 0,
    kMenuSizeMsg = 2,
    kMenuPopUpMsg = 3,
    kMenuCalcItemMsg = 5,
    kMenuThemeSavvyMsg = 7,
    kMenuInitMsg = 8,
    kMenuDisposeMsg = 9,
    kMenuFindItemMsg = 10,
    kMenuHiliteItemMsg = 11,
    kMenuDrawItemsMsg = 12,
    mDrawMsg = kMenuDrawMsg,
    mSizeMsg = kMenuSizeMsg,
    mPopUpMsg = kMenuPopUpMsg,
    mCalcItemMsg = kMenuCalcItemMsg
};
```

Constants

`kMenuDrawMsg`
Draw the menu in the specified rectangle.
Available in Mac OS X v10.0 and later.
Declared in `Menus.h`.

`kMenuSizeMsg`
Calculate the dimensions of the menu rectangle and store them in the menu structure.
Available in Mac OS X v10.0 and later.
Declared in `Menus.h`.

`kMenuPopUpMsg`
Calculate the dimensions of the pop-up menu.
Available in Mac OS X v10.0 and later.
Declared in `Menus.h`.

`kMenuCalcItemMsg`
Calculate the dimensions of the specified menu item.
Available in Mac OS X v10.0 and later.
Declared in `Menus.h`.

`kMenuThemeSavvyMsg`
Identify whether your menu definition function is theme-compliant. If so, your menu definition function should respond by passing back `kThemeSavvyMenuResponse` in the `whichItem` parameter. The Menu Manager then draws the menu background as appropriate for the current theme.
Available in Mac OS X v10.0 and later.
Declared in `Menus.h`.

`kMenuInitMsg`
Perform any initializations required for the menu. Return an error code in `*whichItem` to indicate success or failure.

Available in Mac OS X v10.0 and later.

Declared in `Menus.h`.

`kMenuDisposeMsg`
Dispose of the menu.

Available in Mac OS X v10.0 and later.

Declared in `Menus.h`.

`kMenuFindItemMsg`
Determine the item underneath the mouse.

Available in Mac OS X v10.0 and later.

Declared in `Menus.h`.

`kMenuHiliteItemMsg`
Highlight the specified menu item.

Available in Mac OS X v10.0 and later.

Declared in `Menus.h`.

`kMenuDrawItemsMsg`
Draw the specified menu items.

Available in Mac OS X v10.0 and later.

Declared in `Menus.h`.

`mDrawMsg`
Same as `kMenuDrawMsg`. Obsolete.

Available in Mac OS X v10.0 and later.

Declared in `Menus.h`.

`mSizeMsg`
Same as `kMenuSizeMsg`. Obsolete.

Available in Mac OS X v10.0 and later.

Declared in `Menus.h`.

`mPopUpMsg`
Same as `kMenuPopUpMsg`. Obsolete.

Available in Mac OS X v10.0 and later.

Declared in `Menus.h`.

`mCalcItemMsg`
Same as `kMenuCalcItemMsg`. Obsolete.

Available in Mac OS X v10.0 and later.

Declared in `Menus.h`.

Discussion
For applications running on Mac OS X v10.3 and later, Apple recommends creating custom menu definitions using HIView subclasses rather than MDEF messages. See Introducing HIView for more information.

The Menu Manager passes a value defined by one of these constants in the `message` parameter of your menu definition function specifying what action your function must perform. Other messages are reserved for internal use by Apple Computer, Inc. For more information on how to respond to the various messages, see `MenuDefProcPtr` (page 112).

Obsolete Menu Definition Messages

Older MDEF messages.

```
enum {
    mChooseMsg = 1,
    mDrawItemMsg = 4,
    kMenuChooseMsg = mChooseMsg,
    kMenuDrawItemMsg = mDrawItemMsg
};
```

Constants

`mChooseMsg`

Determine whether the specified mouse location is in an enabled menu item, and highlight or unhighlight the menu item appropriately. Carbon `MDEF`s must replace `mChooseMsg` with the new messages `kMenuFindItemMsg` and `kMenuHiliteItemMsg`.

Available in Mac OS X v10.0 and later.

Declared in `Menus.h`.

`mDrawItemMsg`

Draw the specified menu item in the specified rectangle. `mDrawItemMsg` was used by the popup menu control in versions of the Mac OS prior to Mac OS 8.5, but is no longer used.

Available in Mac OS X v10.0 and later.

Declared in `Menus.h`.

`kMenuChooseMsg`

Same as `mChooseMsg`.

Available in Mac OS X v10.0 and later.

Declared in `Menus.h`.

`kMenuDrawItemMsg`

Same as `mDrawItemMsg`.

Available in Mac OS X v10.0 and later.

Declared in `Menus.h`.

Hierarchical Font Menu Option Constant

Indicates that the font menu should be hierarchical.

```
enum {
    kHierarchicalFontMenuOption = 0x00000001
};
```

Constants

`kHierarchicalFontMenuOption`

The parent menu displays the font families, with font variations (plain, bold, and so on) displayed in submenus.

Available in Mac OS X v10.0 and later.

Declared in `Menus.h`.

Discussion

Use this option constant when calling the `CreateStandardFontMenu` (page 35) function.

Menu Attribute Constants

Specify menu attributes.

```
typedef UInt32 MenuAttributes;
enum {
    kMenuAttrExcludesMarkColumn = (1 << 0),
    kMenuAttrAutoDisable = (1 << 2),
    kMenuAttrUsePencilGlyph = (1 << 3),
    kMenuAttrHidden = (1 << 4),
    kMenuAttrCondenseSeparators = (1 << 5),
    kMenuAttrDoNotCacheImage = (1 << 6),
    kMenuAttrDoNotUseUserCommandKeys = (1 << 7)
};
```

Constants

`kMenuAttrExcludesMarkColumn`

No column space is allocated for the mark character when this menu is drawn.

Available in Mac OS X v10.0 and later.

Declared in `Menus.h`.

`kMenuAttrAutoDisable`

The menu title is automatically disabled when all of its menu items are disabled.

Available in Mac OS X v10.0 and later.

Declared in `Menus.h`.

`kMenuAttrUsePencilGlyph`

Use the pencil glyph from the Keyboard font (`kMenuPencilGlyph`) to draw the control modifier keys when drawing keyboard equivalents. Typically used only for Japanese input method menus.

Available in Mac OS X v10.0 and later.

Declared in `Menus.h`.

`kMenuAttrHidden`

Do not draw the menu title, even when the menu is inserted in the menu bar. This attribute is useful for specifying keyboard equivalent commands that don't correspond with visible menu items. That is, you can add command key equivalents to to menu items and keep the menu itself from appearing in the menu.

Declared in `Menus.h`.

Available in Mac OS X v10.2 and later.

`kMenuAttrCondenseSeparators`

Hides extra separators to avoid blank spaces in a menu. That is, if separators exists at the beginning or end of a menu, or if multiple contiguous separators exist, the Menu Manager marks the extra separator items as hidden. The Menu Manager checks for extra separators whenever it recalculates the menu size.

Declared in `Menus.h`.

Available in Mac OS X v10.3 and later.

`kMenuAttrDoNotCacheImage`

Disables automatic caching of the menu image. Normally, the Menu Manager caches images of all HIView–based menus. (All standard menus are drawn using HIViews in Mac OS X v10.3 and later.) If you specify this attribute, the Menu Manager draws the menu each time it is displayed.

Declared in `Menus.h`.

Available in Mac OS X v10.3 and later.

`kMenuAttrDoNotUseUserCommandKeys`

Disables substitution of command key equivalents from the `NSUserKeysEquivalents` dictionary. By default, the Menu Manager checks for matches in the dictionary for every menu item. Note that this attribute is effective only if you set it when you create the menu; After the Menu Manager searches the dictionary and sets the user command keys (which occurs in the `CalcMenuSize` (page 22), `GetMenuItemCommandKey` (page 59), `GetItemCmd` (page 48) and before command key matching), you cannot retrieve the original command keys. Similarly, clearing this attribute does not restore the original command keys.

Declared in `Menus.h`.

Available in Mac OS X v10.3 and later.

Discussion

Menu attributes control behavior of the entire menu. They are used with the `ChangeMenuAttributes` (page 23) and `GetMenuAttributes` (page 51).

Menu Item Attribute Constants

Specify attributes for menu items.

```
typedef UInt32 MenuItemAttributes;
enum {
    kMenuItemAttrDisabled = (1 << 0),
    kMenuItemAttrIconDisabled = (1 << 1),
    kMenuItemAttrSubmenuParentChoosable = (1 << 2),
    kMenuItemAttrDynamic = (1 << 3),
    kMenuItemAttrNotPreviousAlternate = (1 << 4),
    kMenuItemAttrHidden = (1 << 5),
    kMenuItemAttrSeparator = (1 << 6),
    kMenuItemAttrSectionHeader = (1 << 7),
    kMenuItemAttrIgnoreMeta = (1 << 8),
    kMenuItemAttrAutoRepeat = (1 << 9),
    kMenuItemAttrUseVirtualKey = (1 << 10),
    kMenuItemAttrCustomDraw = (1 << 11),
    kMenuItemAttrIncludeInCmdKeyMatching = (1 << 12),
    kMenuItemAttrAutoDisable = (1 << 13),
    kMenuItemAttrUpdateSingleItem = (1 << 14)
};
```

Constants

`kMenuItemAttrDisabled`

This menu item is disabled.

Available in Mac OS X v10.0 and later.

Declared in `Menus.h`.

`kMenuItemAttrIconDisabled`

This menu item's icon is disabled.

Available in Mac OS X v10.0 and later.

Declared in `Menus.h`.

`kMenuItemAttrSubmenuParentChoosable`

The user can select the parent item of a submenu.

Available in Mac OS X v10.0 and later.

Declared in `Menus.h`.

`kMenuItemAttrDynamic`

This menu item changes dynamically based on the state of the modifier keys. For example, holding down the command key might change the menu item from "Select widget" to "Select all widgets."

When a menu item has alternate dynamic states, you should group them together sequentially in the menu and assign them the same command key. A collection of menu item alternates is called a dynamic group.

Available in Mac OS X v10.0 and later.

Declared in `Menus.h`.

`kMenuItemAttrNotPreviousAlternate`

This item is not part of the same dynamic group as the previous item. The Menu Manager determines which menu items belong to a dynamic group by examining the command keys of each item; if a menu item has the same command key as the previous item, the Menu Manager considers it to be part of the same dynamic group.

However, in some cases you may have sequential items with the same command key (or no command key at all) that should not be considered part of the same dynamic group. To distinguish the separation, you should set this flag for the first menu item in the new group.

Available in Mac OS X v10.0 and later.

Declared in `Menus.h`.

`kMenuItemAttrHidden`
The menu item is not drawn when displaying the menu. The item is also not included in command-key matching unless the `kMenuItemAttrDynamic` or `kMenuItemIncludeInCmdKeyMatching` attribute is set.

Available in Mac OS X v10.0 and later.

Declared in `Menus.h`.

`kMenuItemAttrSeparator`
The menu item is a separator; any text in the item is ignored.

Available in Mac OS X v10.0 and later.

Declared in `Menus.h`.

`kMenuItemAttrSectionHeader`
The menu item is a menu section header; this item is disabled and not selectable.

Available in Mac OS X v10.0 and later.

Declared in `Menus.h`.

`kMenuItemAttrIgnoreMeta`
Ignore the dash (-) metacharacter in this menu item. Dashes at the beginning of a menu item title traditionally signify that the menu item is a separator. However, in some cases you might want to display the dash in the title (for example, if you wanted the menu item to read "-40 degrees F.")

Available in Mac OS X v10.0 and later.

Declared in `Menus.h`.

`kMenuItemAttrAutoRepeat`
The `IsMenuKeyEvent` (page 80) event function recognizes this menu item when it receives an autorepeat keyboard event.

Available in Mac OS X v10.0 and later.

Declared in `Menus.h`.

`kMenuItemAttrUseVirtualKey`
When `MenuEvent` (page 84) and `IsMenuKeyEvent` (page 80) compare this menu item's keyboard equivalent against a keyboard event, they use the item's virtual keycode equivalent rather than its character code equivalent.

Available in Mac OS X v10.0 and later.

Declared in `Menus.h`.

`kMenuItemAttrCustomDraw`
This is a custom menu item. Setting this attribute causes custom menu item drawing Carbon events to be sent to your application. Available in CarbonLib 1.4 and Mac OS X v10.1 and later.

Available in Mac OS X v10.1 and later.

Declared in `Menus.h`.

`kMenuItemAttrIncludeInCmdKeyMatching`
If this attribute is set, functions such as `MenuKey` (page 180), `MenuEvent` (page 84) and `IsMenuKeyEvent` (page 80) examine this menu item during command key matching. Typically, visible items are examined and hidden items (unless they have the `kMenuItemAttrDynamic` attribute set) are ignored during command key matching. However, by setting this attribute, you can force hidden items to be included in the matching. Available in CarbonLib 1.6 and Mac OS X v10.2 and later.

Available in Mac OS X v10.2 and later.

Declared in `Menus.h`.

`kMenuItemAttrAutoDisable`

Disables the menu item if it does not respond to the `kEventCommandUpdateStatus` event . That is, if no `kEventCommandUpdateStatus` handler is installed on this item, or if all the handlers installed for the update event return `eventNotHandledErr`, this item is automatically disabled. This attribute is useful if your application uses the `kEventCommandUpdateStatus` event to enable menu items; for example you no longer have to install an update status handler on the application target to disable menu items when there are no document windows open.

Available in Mac OS X v10.3 and later.

Declared in `Menus.h`.

`kMenuItemAttrUpdateSingleItem`

Update only the menu item that matches when searching available command keys. Normally when the Menu Manager does command key matching, it sends a `kEventMenuEnableItems` event to the menu containing the matching item and then sends a `kEventCommandUpdateStatus` to each item in the menu. Doing so can be inefficient, since in most cases only the item that matches needs to be updated. By setting this attribute, only the matching item receives the update event and `kEventMenuEnableItems` is not sent to the menu. If your application enables menu items solely through `kEventCommandUpdateStatus` event handlers, you should set this attribute for your menu items.

Declared in `Menus.h`.

Available in Mac OS X v10.3 and later.

Discussion

Menu item attributes control behavior of individual menu items. They are used with the `GetMenuItemAttributes` (page 58) and `ChangeMenuItemAttributes` (page 24) APIs.

Menu Definition Type Constants

Indicate the type of menu definition being used.

```
enum {
    kMenuDefProcPtr = 0,
    kMenuDefClassID = 1
};
typedef UInt32 MenuDefType;
```

Constants

`kMenuDefProcPtr`

A custom menu definition using the older MDEF messaging model.

Available in Mac OS X v10.0 and later.

Declared in `Menus.h`.

`kMenuDefClassID`

A custom menu definition using an HIView subclass.

Available in Mac OS X v10.3 and later.

Declared in `Menus.h`.

Menu Definition Feature Constants

Indicate menu definition features.

```
enum {
    kThemeSavvyMenuResponse = 0x7473
};
```

Constants

`kThemeSavvyMenuResponse`

Indicates that the menu is Appearance theme–savvy.

Available in Mac OS X v10.0 and later.

Declared in `Menus.h`.

Discussion

The Menu Manager may pass the `kMenuThemeSavvyMsg` constant in the `message` parameter of your menu definition function to determine if your custom menu is theme-savvy (that is, whether it draws using Appearance Manager functions). In response, your menu definition function may respond with this flag in the `whichItem` parameter.

Menu Definition IDs

Specify options used in menu item functions.

```
enum {
    textMenuProc = 0,
    hMenuCmd = 27,
    hierMenu = -1,
    kInsertHierarchicalMenu = -1,
};
```

Constants

`textMenuProc`

The menu definition ID for menus that are not Appearance-compliant. When mapping is enabled, this constant is mapped to `kMenuStdMenuProc`, its Appearance-compliant equivalent. Not normally used.

Available in Mac OS X v10.0 and later.

Declared in `Menus.h`.

`hMenuCmd`

Deprecated. Use `SetMenuItemHierarchicalMenu` (page 101) or `SetMenuItemHierarchicalID` (page 189) to specify a hierarchical menu instead.

Available in Mac OS X v10.0 and later.

Declared in `Menus.h`.

`hierMenu`

Deprecated. Use `kInsertHierarchicalMenu` instead.

Available in Mac OS X v10.0 and later.

Declared in `Menus.h`.

`kInsertHierarchicalMenu`

Used with `InsertMenu` (page 73) to insert a submenu or pop-up menu into the submenu portion of the current menu list.

Available in Mac OS X v10.0 and later.

Declared in `Menus.h`.

`kHIMenuAppendItem`
Pass to `InsertMenuItem` (page 176), `InsertMenuItemText` (page 177), or `InsertMenuItemTextWithCFString` (page 74) to indicate that the new menu item should be added to the end of the menu. Note that you can simply call `AppendMenu` (page 163), `AppendMenuItemText` (page 164), or `AppendMenuItemTextWithCFString` (page 21) instead.

Available in Mac OS X v10.3 and later.

Declared in `Menus.h`.

Discussion
A menu definition ID is supplied to the menu resource or a menu-creation function such as `NewMenu` to specify which menu definition function to use in creating the menu. The menu definition ID contains the resource ID of the menu definition function.

Menu Event Option Constants

Specify options when attempting to match a keyboard event to a menu item.

```
typedef UInt32 MenuEventOptions;
enum {
    kMenuEventIncludeDisabledItems = 0x0001,
    kMenuEventQueryOnly = 0x0002,
    kMenuEventDontCheckSubmenus = 0x0004
};
```

Constants
`kMenuEventIncludeDisabledItems`
Disabled items are examined for a match.

Available in Mac OS X v10.0 and later.

Declared in `Menus.h`.

`kMenuEventQueryOnly`
Don't highlight the menu title if a match is found.

Available in Mac OS X v10.0 and later.

Declared in `Menus.h`.

`kMenuEventDontCheckSubmenus`
Don't search the submenus of the starting menu when looking for a match.

Available in Mac OS X v10.0 and later.

Declared in `Menus.h`.

Discussion
Menu event options control how the menus are searched for an item matching a particular keyboard event. They are used with the `IsMenuKeyEvent` (page 80) API.

Menu Glyph Constants

Specify menu glyphs.

```
enum {
     kMenuNullGlyph = 0x00,
     kMenuTabRightGlyph = 0x02,
     kMenuTabLeftGlyph = 0x03,
     kMenuEnterGlyph = 0x04,
     kMenuShiftGlyph = 0x05,
     kMenuControlGlyph = 0x06,
     kMenuOptionGlyph = 0x07,
     kMenuSpaceGlyph = 0x09,
     kMenuDeleteRightGlyph = 0x0A,
     kMenuReturnGlyph = 0x0B,
     kMenuReturnR2LGlyph = 0x0C,
     kMenuNonmarkingReturnGlyph = 0x0D,
     kMenuPencilGlyph = 0x0F,
     kMenuDownwardArrowDashedGlyph = 0x10,
     kMenuCommandGlyph = 0x11,
     kMenuCheckmarkGlyph = 0x12,
     kMenuDiamondGlyph = 0x13,
     kMenuAppleLogoFilledGlyph = 0x14,
     kMenuParagraphKoreanGlyph = 0x15,
     kMenuDeleteLeftGlyph = 0x17,
     kMenuLeftArrowDashedGlyph = 0x18,
     kMenuUpArrowDashedGlyph = 0x19,
     kMenuRightArrowDashedGlyph = 0x1A,
     kMenuEscapeGlyph = 0x1B,
     kMenuClearGlyph = 0x1C,
     kMenuLeftDoubleQuotesJapaneseGlyph = 0x1D,
     kMenuRightDoubleQuotesJapaneseGlyph = 0x1E,
     kMenuTrademarkJapaneseGlyph = 0x1F,
     kMenuBlankGlyph = 0x61,
     kMenuPageUpGlyph = 0x62,
     kMenuCapsLockGlyph = 0x63,
     kMenuLeftArrowGlyph = 0x64,
     kMenuRightArrowGlyph = 0x65,
     kMenuNorthwestArrowGlyph = 0x66,
     kMenuHelpGlyph = 0x67,
     kMenuUpArrowGlyph = 0x68,
     kMenuSoutheastArrowGlyph = 0x69,
     kMenuDownArrowGlyph = 0x6A,
     kMenuPageDownGlyph = 0x6B,
     kMenuAppleLogoOutlineGlyph = 0x6C,
     kMenuContextualMenuGlyph = 0x6D,
     kMenuPowerGlyph = 0x6E,
     kMenuF1Glyph = 0x6F,
     kMenuF2Glyph = 0x70,
     kMenuF3Glyph = 0x71,
     kMenuF4Glyph = 0x72,
     kMenuF5Glyph = 0x73,
     kMenuF6Glyph = 0x74,
     kMenuF7Glyph = 0x75,
     kMenuF8Glyph = 0x76,
     kMenuF9Glyph = 0x77,
     kMenuF10Glyph = 0x78,
     kMenuF11Glyph = 0x79,
     kMenuF12Glyph = 0x7A,
     kMenuF13Glyph = 0x87,
     kMenuF14Glyph = 0x88,
     kMenuF15Glyph = 0x89,
```

```
    kMenuControlISOGlyph = 0x8A,
    kMenuEjectGlyph = 0x8C
};
```

Constants

`kMenuNullGlyph`
The null character. Note that this glyph has no visible representation (that is, nothing appears in the menu).

Available in Mac OS X v10.0 and later.

Declared in `Menus.h`.

`kMenuTabRightGlyph`
The Tab-to-the-right key. Used in left to right script systems.

Available in Mac OS X v10.0 and later.

Declared in `Menus.h`.

`kMenuTabLeftGlyph`
The Tab-to-the-left key. Used in right to left script systems.

Available in Mac OS X v10.0 and later.

Declared in `Menus.h`.

`kMenuEnterGlyph`
The Enter key.

Available in Mac OS X v10.0 and later.

Declared in `Menus.h`.

`kMenuShiftGlyph`
The Shift key.

Available in Mac OS X v10.0 and later.

Declared in `Menus.h`.

`kMenuControlGlyph`
The Control key.

Available in Mac OS X v10.0 and later.

Declared in `Menus.h`.

`kMenuOptionGlyph`
The Option key.

Available in Mac OS X v10.0 and later.

Declared in `Menus.h`.

`kMenuSpaceGlyph`
The Space bar. Note that this glyph has no visible representation (that is, nothing appears in the menu).

Available in Mac OS X v10.0 and later.

Declared in `Menus.h`.

`kMenuDeleteRightGlyph`
The Delete-to-the-right key. Used in right-to-left script systems.

Available in Mac OS X v10.0 and later.

Declared in `Menus.h`.

`kMenuReturnGlyph`
The Return key for left-to-right script systems.

Available in Mac OS X v10.0 and later.

Declared in `Menus.h`.

`kMenuReturnR2LGlyph`
The Return key for right-to-left script systems.

Available in Mac OS X v10.0 and later.

Declared in `Menus.h`.

`kMenuNonmarkingReturnGlyph`
The nonmarking Return key. Note that this glyph has no visible representation (that is, nothing appears in the menu).

Available in Mac OS X v10.0 and later.

Declared in `Menus.h`.

`kMenuPencilGlyph`
The Pencil key.

Available in Mac OS X v10.0 and later.

Declared in `Menus.h`.

`kMenuDownwardArrowDashedGlyph`
The downward dashed arrow key.

Available in Mac OS X v10.0 and later.

Declared in `Menus.h`.

`kMenuCommandGlyph`
The Command key.

Available in Mac OS X v10.0 and later.

Declared in `Menus.h`.

`kMenuCheckmarkGlyph`
The Check mark key.

Available in Mac OS X v10.0 and later.

Declared in `Menus.h`.

`kMenuDiamondGlyph`
The diamond mark.

Available in Mac OS X v10.0 and later.

Declared in `Menus.h`.

`kMenuAppleLogoFilledGlyph`
The filled Apple logo.

Available in Mac OS X v10.0 and later.

Declared in `Menus.h`.

`kMenuParagraphKoreanGlyph`
Unassigned. (Paragraph glyph in Korean)

Available in Mac OS X v10.0 and later.

Declared in `Menus.h`.

`kMenuDeleteLeftGlyph`
The Delete-to-the-left key.
Available in Mac OS X v10.0 and later.
Declared in `Menus.h`.

`kMenuLeftArrowDashedGlyph`
The dashed left arrow key.
Available in Mac OS X v10.0 and later.
Declared in `Menus.h`.

`kMenuUpArrowDashedGlyph`
The dashed up arrow key.
Available in Mac OS X v10.0 and later.
Declared in `Menus.h`.

`kMenuRightArrowDashedGlyph`
The dashed right arrow key.
Available in Mac OS X v10.0 and later.
Declared in `Menus.h`.

`kMenuEscapeGlyph`
The Escape key.
Available in Mac OS X v10.0 and later.
Declared in `Menus.h`.

`kMenuClearGlyph`
The Clear key.
Available in Mac OS X v10.0 and later.
Declared in `Menus.h`.

`kMenuLeftDoubleQuotesJapaneseGlyph`
Unassigned. (Left double quotation marks in Japanese)
Available in Mac OS X v10.0 and later.
Declared in `Menus.h`.

`kMenuRightDoubleQuotesJapaneseGlyph`
Unassigned (Right double quotation marks in Japanese)
Available in Mac OS X v10.0 and later.
Declared in `Menus.h`.

`kMenuTrademarkJapaneseGlyph`
Unassigned. (Trademark in Japanese)
Available in Mac OS X v10.0 and later.
Declared in `Menus.h`.

`kMenuBlankGlyph`
The blank key.
Available in Mac OS X v10.0 and later.
Declared in `Menus.h`.

`kMenuPageUpGlyph`
The Page Up key.
Available in Mac OS X v10.0 and later.
Declared in `Menus.h`.

`kMenuCapsLockGlyph`
The Caps Lock key.
Available in Mac OS X v10.0 and later.
Declared in `Menus.h`.

`kMenuLeftArrowGlyph`
The left arrow key.
Available in Mac OS X v10.0 and later.
Declared in `Menus.h`.

`kMenuRightArrowGlyph`
The right arrow key.
Available in Mac OS X v10.0 and later.
Declared in `Menus.h`.

`kMenuNorthwestArrowGlyph`
The northwest arrow key
Available in Mac OS X v10.0 and later.
Declared in `Menus.h`.

`kMenuHelpGlyph`
The Help key.
Available in Mac OS X v10.0 and later.
Declared in `Menus.h`.

`kMenuUpArrowGlyph`
The up arrow key.
Available in Mac OS X v10.0 and later.
Declared in `Menus.h`.

`kMenuSoutheastArrowGlyph`
The southeast arrow key.
Available in Mac OS X v10.0 and later.
Declared in `Menus.h`.

`kMenuDownArrowGlyph`
The down arrow key.
Available in Mac OS X v10.0 and later.
Declared in `Menus.h`.

`kMenuPageDownGlyph`
The Page Down key.
Available in Mac OS X v10.0 and later.
Declared in `Menus.h`.

`kMenuAppleLogoOutlineGlyph`
The outlined Apple logo.
Available in Mac OS X v10.0 and later.
Declared in `Menus.h`.

`kMenuContextualMenuGlyph`
The contextual menu key
Available in Mac OS X v10.0 and later.
Declared in `Menus.h`.

`kMenuPowerGlyph`
The power key (that is, the startup key).
Available in Mac OS X v10.0 and later.
Declared in `Menus.h`.

`kMenuF1Glyph`
The F1 key.
Available in Mac OS X v10.0 and later.
Declared in `Menus.h`.

`kMenuF2Glyph`
The F2 key.
Available in Mac OS X v10.0 and later.
Declared in `Menus.h`.

`kMenuF3Glyph`
The F3 key.
Available in Mac OS X v10.0 and later.
Declared in `Menus.h`.

`kMenuF4Glyph`
The F4 key.
Available in Mac OS X v10.0 and later.
Declared in `Menus.h`.

`kMenuF5Glyph`
The F5 key.
Available in Mac OS X v10.0 and later.
Declared in `Menus.h`.

`kMenuF6Glyph`
The F6 key.
Available in Mac OS X v10.0 and later.
Declared in `Menus.h`.

`kMenuF7Glyph`
The F7 key.
Available in Mac OS X v10.0 and later.
Declared in `Menus.h`.

`kMenuF8Glyph`
The F8 key.
Available in Mac OS X v10.0 and later.
Declared in `Menus.h`.

`kMenuF9Glyph`
The F9 key.
Available in Mac OS X v10.0 and later.
Declared in `Menus.h`.

`kMenuF10Glyph`
The F10 key.
Available in Mac OS X v10.0 and later.
Declared in `Menus.h`.

`kMenuF11Glyph`
The F11 key.
Available in Mac OS X v10.0 and later.
Declared in `Menus.h`.

`kMenuF12Glyph`
The F12 key.
Available in Mac OS X v10.0 and later.
Declared in `Menus.h`.

`kMenuF13Glyph`
The F13 key.
Available in Mac OS X v10.0 and later.
Declared in `Menus.h`.

`kMenuF14Glyph`
The F14 key.
Available in Mac OS X v10.0 and later.
Declared in `Menus.h`.

`kMenuF15Glyph`
The F15 key.
Available in Mac OS X v10.0 and later.
Declared in `Menus.h`.

`kMenuControlISOGlyph`
The ISO standard control key.
Available in Mac OS X v10.0 and later.
Declared in `Menus.h`.

`kMenuEjectGlyph`
The Eject key (available in Mac OS X v10.2 and later).
Available in Mac OS X v10.2 and later.
Declared in `Menus.h`.

Discussion
Use these constants with `GetMenuItemKeyGlyph` (page 63) and `SetMenuItemKeyGlyph` (page 104).

Menu Item Data Flags

Indicate which fields of a `MenuItemDataRec` structure are to be copied or set.

```
enum {
    kMenuItemDataText = (1 << 0),
    kMenuItemDataMark = (1 << 1),
    kMenuItemDataCmdKey = (1 << 2),
    kMenuItemDataCmdKeyGlyph = (1 << 3),
    kMenuItemDataCmdKeyModifiers = (1 << 4),
    kMenuItemDataStyle = (1 << 5),
    kMenuItemDataEnabled = (1 << 6),
    kMenuItemDataIconEnabled = (1 << 7),
    kMenuItemDataIconID = (1 << 8),
    kMenuItemDataIconHandle = (1 << 9),
    kMenuItemDataCommandID = (1 << 10),
    kMenuItemDataTextEncoding = (1 << 11),
    kMenuItemDataSubmenuID = (1 << 12),
    kMenuItemDataSubmenuHandle = (1 << 13),
    kMenuItemDataFontID = (1 << 14),
    kMenuItemDataRefcon = (1 << 15),
    kMenuItemDataAttributes = (1 << 16),
    kMenuItemDataCFString = (1 << 17),
    kMenuItemDataProperties = (1 << 18),
    kMenuItemDataIndent = (1 << 19),
    kMenuItemDataCmdVirtualKey = (1 << 20),
    kMenuItemDataAllDataVersionOne = 0x000FFFFF,
    kMenuItemDataAllDataVersionTwo = kMenuItemDataAllDataVersionOne
| kMenuItemDataCmdVirtualKey
};
enum {
    kMenuItemDataAllData = kMenuItemDataAllDataVersionTwo
};
typedef UInt64 MenuItemDataFlags;
```

Constants

`kMenuItemDataText`

Set or return the `Str255` text of a menu using the `MenuItemDataRec.text` field. If getting the text, the text field must be initialized with a pointer to a `Str255` variable before calling `CopyMenuItemData`. If both `kMenuItemDataText` and `kMenuItemCFString` are set on entry to `CopyMenuItemData`, the API will determine whether the menu text was most recently set using a `Str255` or CFString, and return only that text format; the flags value for the other format will be cleared. Valid for both menu items and the menu title (if item number is 0).

Available in Mac OS X v10.0 and later.

Declared in `Menus.h`.

`kMenuItemDataMark`

Set or return the mark character of a menu item using the `MenuItemDataRec.mark` field. Valid only for menu items.

Available in Mac OS X v10.0 and later.

Declared in `Menus.h`.

`kMenuItemDataCmdKey`
: Set or return the command key of a menu using the `MenuItemDataRec.cmdKey` field. Valid only for menu items.

  Available in Mac OS X v10.0 and later.

  Declared in `Menus.h`.

`kMenuItemDataCmdKeyGlyph`
: Set or return the command key glyph of a menu using the `MenuItemDataRec.cmdKeyGlyph` field. Valid only for menu items.

  Available in Mac OS X v10.0 and later.

  Declared in `Menus.h`.

`kMenuItemDataCmdKeyModifiers`
: Set or return the command key modifiers of a menu using the `MenuItemDataRec.cmdKeyModifiers` field. Valid only for menu items.

  Available in Mac OS X v10.0 and later.

  Declared in `Menus.h`.

`kMenuItemDataStyle`
: Set or return the QuickDraw text style of a menu item using the `MenuItemDataRec.style` field. Valid only for menu items.

  Available in Mac OS X v10.0 and later.

  Declared in `Menus.h`.

`kMenuItemDataEnabled`
: Set or return the enable state of a menu using the `MenuItemDataRec.enabled` field. Valid for both menu items and the menu itself (if the item number is zero).

  Available in Mac OS X v10.0 and later.

  Declared in `Menus.h`.

`kMenuItemDataIconEnabled`
: Set or return the enable state of the menu item icon using the `MenuItemDataRec.iconEnabled` field. Valid only for menu items.

  Available in Mac OS X v10.0 and later.

  Declared in `Menus.h`.

`kMenuItemDataIconID`
: Set or return the icon resource ID of the menu item icon using the `MenuItemDataRec.iconID` field. Valid only for menu items.

  Available in Mac OS X v10.0 and later.

  Declared in `Menus.h`.

`kMenuItemDataIconHandle`

Set or return the icon handle of a menu item using the `MenuItemDataRec.iconType` and `MenuItemDataRec.iconHandle` field. You must initialize both fields if you are setting the handle; both fields are returned when obtaining the handle.

The `iconType` field can contain one of the following constants: `kMenuIconType`, `kMenuShrinkIconType`, `kMenuSmallIconType`, `kMenuColorIcontype`, `kMenuIconSuiteType`, or `kMenuIconRefType`. The icon handle may be a handle to an `'ICON'` resource, a `'SICN'` resource, a `'cicn'` resource, an icon suite, or an icon reference. Valid only for menu items.

In Mac OS X v10.0 and later, the `iconType` field can also contain `kMenuCGImageType`, with the icon handle being of type `CGImageRef`.

In Mac OS X v10.1 and later, the `iconType` field can also contain `kMenuSystemIconSelectorType` or `kMenuIconResource`, which have icon handles of type `OSType` and `CFStringRef` respectively.

Available in Mac OS X v10.0 and later.

Declared in `Menus.h`.

`kMenuItemDataCommandID`

Set or return the command ID of a menu using the `MenuItemDataRec.cmdID` field. Valid only for menu items.

Available in Mac OS X v10.0 and later.

Declared in `Menus.h`.

`kMenuItemDataTextEncoding`

Set or return the text encoding of a menu item using the `MenuItemDataRec.encoding` field. Valid only for menu items.

Available in Mac OS X v10.0 and later.

Declared in `Menus.h`.

`kMenuItemDataSubmenuID`

Set or return the menu ID of the submenu associated with this menu item using the `MenuItemDataRec.submenuID` field. Valid only for menu items.

Available in Mac OS X v10.0 and later.

Declared in `Menus.h`.

`kMenuItemDataSubmenuHandle`

Set or return the menu reference (`MenuRef`) of the submenu associated with this menu using the `MenuItemDataRec.submenuHandle` field. Valid only for menu items.

Available in Mac OS X v10.0 and later.

Declared in `Menus.h`.

`kMenuItemDataFontID`

Set or return the font ID associated with this menu item using the `MenuItemDataRec.fomtID` field. Valid only for menu items.

Available in Mac OS X v10.0 and later.

Declared in `Menus.h`.

`kMenuItemDataRefcon`

Set or return the reference constant associated with this menu item using the `MenuItemDataRec.refcon` field. If you specified a menu item index of 0, you can set or obtain the menu reference constant.

Available in Mac OS X v10.0 and later.

Declared in `Menus.h`.

`kMenuItemDataAttributes`
Set or return the attribute bits associated with this menu item using the `MenuItemDataRec.attr` field. If you specified a menu item index of 0, you can set or obtain a `MenuAttributes` bit field, not a `MenuItemAttributes` bit field.

Available in Mac OS X v10.0 and later.

Declared in `Menus.h`.

`kMenuItemDataCFString`
Set or return the title of the menu item (as a Core Foundation string) using the `MenuItemDataRec.cFText` field. If you specified a menu item index of 0, you can set or obtain the menu title.

Available in Mac OS X v10.0 and later.

Declared in `Menus.h`.

`kMenuItemDataProperties`
Set or return the properties of a menu using the `MenuItemDataRec.properties` field.

If setting properties, the `properties` field should contain a collection with the new properties; note that this will overwrite any existing properties with the same collection creator and tag.

If getting properties, you should set the `properties` field to either a valid collection or `NULL`. A valid collection is overwritten by the new properties. If you pass `NULL`, the `CopyMenuItemData` (page 30) function allocates a new collection and returns it in the `properties` field.

You can set this flag for both menu items and the menu itself (if the item number is zero).

Available in Mac OS X v10.0 and later.

Declared in `Menus.h`.

`kMenuItemDataIndent`
Set or return the indent level of a menu item using the `MenuItemDataRec.indent` field. Valid only for menu items.

Available in Mac OS X v10.0 and later.

Declared in `Menus.h`.

`kMenuItemDataCmdVirtualKey`
Set or return the virtual key code for this menu item using the `MenuItemDataRec.cmdVirtualKey` field. Valid only for menu items.

Available in Mac OS X v10.0 and later.

Declared in `Menus.h`.

`kMenuItemDataAllDataVersionOne`
Sets all flags, except for `kMenuItemDataCmdVirtualKey`.

Available in Mac OS X v10.0 and later.

Declared in `Menus.h`.

`kMenuItemDataAllDataVersionTwo`
Sets all flags, including `kMenuItemDataCmdVirtualKey`.

Available in Mac OS X v10.0 and later.

Declared in `Menus.h`.

Discussion
A `MenuItemDataFlags` value indicates which fields of a "MenuItemDataRec" (page 127) structure should be used by the `CopyMenuItemData` (page 30) or `SetMenuItemData` (page 100) functions. All menu item data flags may be used when getting or setting the contents of a menu item; some may also be used when getting or setting information about the menu itself, if the item index given to `CopyMenuItemData` (page 30) or `SetMenuItemData` (page 100) is 0.

Menu Item Icon Type Constants

Specify types of icons to attach to menu items.

```
enum {
    kMenuNoIcon = 0,
    kMenuIconType = 1,
    kMenuShrinkIconType = 2,
    kMenuSmallIconType = 3,
    kMenuColorIconType = 4,
    kMenuIconSuiteType = 5,
    kMenuIconRefType = 6,
    kMenuCGImageRefType = 7,
    kMenuSystemIconSelectorType = 8,
    kMenuIconResourceType = 9
};
```

Constants

`kMenuNoIcon`
No icon.
Available in Mac OS X v10.0 and later.
Declared in `Menus.h`.

`kMenuIconType`
Identifies an icon of type `'ICON'`.
Available in Mac OS X v10.0 and later.
Declared in `Menus.h`.

`kMenuShrinkIconType`
Identifies a 32-by-32-pixel icon of type `'ICON'`, shrunk (at display time) to 16-by-16.
Available in Mac OS X v10.0 and later.
Declared in `Menus.h`.

`kMenuSmallIconType`
Identifies an icon of type `'SICN'`.
Available in Mac OS X v10.0 and later.
Declared in `Menus.h`.

`kMenuColorIconType`
Identifies an icon of type 'cicn'.
Available in Mac OS X v10.0 and later.
Declared in `Menus.h`.

`kMenuIconSuiteType`
Identifies an icon suite.

Available in Mac OS X v10.0 and later.

Declared in `Menus.h`.

`kMenuIconRefType`
Identifies an icon of type `IconRef`. This value is supported under Mac OS 8.5 and later.

Available in Mac OS X v10.0 and later.

Declared in `Menus.h`.

`kMenuCGImageRefType`
Identifies an icon of type `CGImageRef`.

Available in Mac OS X v10.0 and later.

Declared in `Menus.h`.

`kMenuSystemIconSelectorType`
Identifies an `OSType` value that corresponds to an icon (type `IconRef`)registered with Icon Services under `kSystemIconsCreator`.

Available in Mac OS X v10.1 and later.

Declared in `Menus.h`.

`kMenuIconResourceType`
Identifies a CFString that names an icon resource in the main bundle of the application.

Available in Mac OS X v10.1 and later.

Declared in `Menus.h`.

Discussion
These constants specify the type of an icon attached to a menu item. They are passed in `SetMenuItemIconHandle` (page 102) and obtained by `GetMenuItemIconHandle` (page 62). Menu item icon type constants are available with Appearance Manager 1.0 and later.

Menu Item Property Attribute Constant

Define attributes to associate with menu item properties.

```
enum {
    kMenuPropertyPersistent = 0x00000001
};
```

Constants
`kMenuPropertyPersistent`
If this bit is set, the menu item property is saved when the menu is flattened.

Available in Mac OS X v10.0 and later.

Declared in `Menus.h`.

Menu Tracking Mode Constants

Indicates how the menu is being tracked.

```
typedef UInt32 MenuTrackingMode;
enum {
    kMenuTrackingModeMouse = 1,
    kMenuTrackingModeKeyboard = 2
};
```

Constants

`kMenuTrackingModeMouse`

Menus are being tracked using the mouse.

Available in Mac OS X v10.0 and later.

Declared in `Menus.h`.

`kMenuTrackingModeKeyboard`

Menus are being tracked using the keyboard.

Available in Mac OS X v10.0 and later.

Declared in `Menus.h`.

Discussion

A menu tracking mode constant is part of the `kEventMenuBeginTracking` and `kEventMenuChangeTrackingMode` Carbon events. It indicates whether a menu is being tracked by mouse movement or by directional keyboard input.

Modifier Key Mask Constants

Specify modifier keys used with menu item selections.

```
enum {
    kMenuNoModifiers = 0,
    kMenuShiftModifier = (1 << 0),
    kMenuOptionModifier = (1 << 1),
    kMenuControlModifier = (1 << 2),
    kMenuNoCommandModifier = (1 << 3)
};
```

Constants

`kMenuNoModifiers`

If no bit is set, only the Command key is used in the keyboard equivalent.

Available in Mac OS X v10.0 and later.

Declared in `Menus.h`.

`kMenuShiftModifier`

If this bit (bit 0) is set, the Shift key is used in the keyboard equivalent.

Available in Mac OS X v10.0 and later.

Declared in `Menus.h`.

`kMenuOptionModifier`

If this bit (bit 1) is set, the Option key is used in the keyboard equivalent.

Available in Mac OS X v10.0 and later.

Declared in `Menus.h`.

`kMenuControlModifier`
If this bit (bit 2) is set, the Control key is used in the keyboard equivalent.

Available in Mac OS X v10.0 and later.

Declared in `Menus.h`.

`kMenuNoCommandModifier`
If this bit (bit 3) is set, the Command key is not used in the keyboard equivalent.

Available in Mac OS X v10.0 and later.

Declared in `Menus.h`.

Discussion
You can use one or more of these mask constants to determine which modifier key(s) must be pressed along with a character key to create a keyboard equivalent for selecting a menu item. You set and obtain these constants by calling `SetMenuItemModifiers` (page 105) and `GetMenuItemModifiers` (page 64), respectively.

No Mark Marking Character Constant

Indicates that a menu item contains no marking characters.

```
enum {
    noMark = 0
};
```

Constants
`noMark`
No marking character to be associated with a menu or submenu item.

Available in Mac OS X v10.0 and later.

Declared in `Menus.h`.

Discussion
You can pass this constant, as well as those character marking constants defined in the Font Manager, in the `markChar` parameter of the function `SetItemMark` (page 92) and the marking character field of the menu resource (of type '`MENU`') and return these constants in the `markChar` parameter of the function `GetItemMark` (page 48) to specify the mark of a specific menu item or the menu ID of the submenu associated with the menu item.

Menu Dismissal Constants

Specify reasons why menu tracking ended.

```
enum {
kHIMenuDismissedBySelection   = 1,
kHIMenuDismissedByUserCancel  = 2,
kHIMenuDismissedByMouseDown   = 3,
kHIMenuDismissedByMouseUp     = 4,
kHIMenuDismissedByKeyEvent    = 5,
kHIMenuDismissedByAppSwitch   = 6,
kHIMenuDismissedByTimeout     = 7,
kHIMenuDismissedByCancelMenuTracking = 8,
kHIMenuDismissedByActivationChange = 9,
kHIMenuDismissedByFocusChange = 10
};
```

Constants

`kHIMenuDismissedBySelection`
The user selected a menu item.

Available in Mac OS X v10.3 and later.

Declared in `Menus.h`.

`kHIMenuDismissedByUserCancel`
The user cancelled menu tracking.

Available in Mac OS X v10.3 and later.

Declared in `Menus.h`.

`kHIMenuDismissedByMouseDown`
The user pressed the mouse someplace that did not result in a menu item selection.

Available in Mac OS X v10.3 and later.

Declared in `Menus.h`.

`kHIMenuDismissedByMouseUp`
The user released the mouse someplace that did not result in a menu item selection.

Available in Mac OS X v10.3 and later.

Declared in `Menus.h`.

`kHIMenuDismissedByKeyEvent`
A keyboard event occurred.

Available in Mac OS X v10.3 and later.

Declared in `Menus.h`.

`kHIMenuDismissedByAppSwitch`
The application with the menu is no longer frontmost.

Available in Mac OS X v10.3 and later.

Declared in `Menus.h`.

`kHIMenuDismissedByTimeout`
The menu tracking mode timed out.

Available in Mac OS X v10.3 and later.

Declared in `Menus.h`.

`kHIMenuDismissedByCancelMenuTracking`
The application called `CancelMenuTracking`.

Available in Mac OS X v10.3 and later.

Declared in `Menus.h`.

`kHIMenuDismissedByActivationChange`
The active window changed.

Available in Mac OS X v10.3 and later.

Declared in `Menus.h`.

`kHIMenuDismissedByFocusChange`
The user focus window changed, or the keyboard focus was removed from the current process.

Available in Mac OS X v10.3 and later.

Declared in `Menus.h`.

Discussion
The Carbon Event Manager passes these constants in the `kEventMenuEndTrackingEvent` to indicate why menu tracking ended.

Standard Menu Definition Constants

Specify the menu definitions for standard menus and menu bars.

```
enum {
    kMenuStdMenuProc = 63,
    kMenuStdMenuBarProc = 63
};
```

Constants
`kMenuStdMenuProc`
The menu definition ID for Appearance-compliant menus.

Available with Appearance Manager 1.0 and later.

Declared in `Menus.h`.

`kMenuStdMenuBarProc`
The menu bar definition ID for Appearance-compliant menu bars.

Available with Appearance Manager 1.0 and later.

Declared in `Menus.h`.

Result Codes

This table lists result codes defined for the Menu Manager.

Result Code	Value	Description
`menuPropertyInvalidErr`	-5603	You specified an Apple-reserved creator type in a menu property function. Available in Mac OS X v10.0 and later.
`menuPropertyNotFoundErr`	-5604	The specified property creator/ID combination was not found. Available in Mac OS X v10.0 and later.

Result Code	Value	Description
`menuNotFoundErr`	-5620	The specified menu or menu ID wasn't found. Available in Mac OS X v10.0 and later.
`menuUsesSystemDefErr`	-5621	`GetMenuDefinition` failed because the menu uses the system menu definition. Available in Mac OS X v10.0 and later.
`menuItemNotFoundErr`	-5622	The specified menu item wasn't found. Available in Mac OS X v10.0 and later.
`menuInvalidErr`	-5623	The menu reference passed to the function was invalid. Available in Mac OS X v10.0 and later.

Deprecated Menu Manager Functions

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

Deprecated in Mac OS X v10.5

AppendMenu

Appends one or more items to a menu previously created. (Deprecated in Mac OS X v10.5.)

Not recommended

```
void AppendMenu (
    MenuRef menu,
    ConstStr255Param data
);
```

Parameters

menu

The menu to which you wish to append the menu item or items.

data

A Pascal string that defines the characteristics of the new menu item or items. Note that in most cases you should store the text of a menu item in a resource, so that your menu items can be more easily localized. The `AppendMenu` function appends the menu items in the order in which they are listed in the data parameter.

Discussion

Note that unless you are supporting legacy code, you should use the `AppendMenuItemTextWithCFString` (page 21) function instead.

The `AppendMenu` function appends any defined menu items to a previously-created menu. The menu items are added to the end of the menu. You specify the text of any menu items and their characteristics in the `data` parameter. You can embed metacharacters in the string to define various characteristics of a menu item.

Table A-1 (page 163) lists the metacharacters that you can specify in the `data` parameter:

Table A-1 Metacharacters available to pass in AppendMenu

Metacharacter	Description
; or Return	Separates menu items.
^	When followed by an icon number, defines the icon for the item. If the keyboard equivalent field contains 0x1C, this number is interpreted as a text encoding.

Metacharacter	Description
!	When followed by a character, defines the mark for the item. If the keyboard equivalent field contains 0x1B or the equivalent constant `hMenuCmd`, this value is interpreted as the menu ID of a submenu of this menu item.
<	When followed by one or more of the characters B, I, U, O, and S, defines the character style of the item to Bold, Italic, Underline, Outline, or Shadow, respectively.
/	When followed by a character, defines the keyboard equivalent for the item. When followed by 0x1B or the equivalent constant `hMenuCmd`, specifies that this menu item has a submenu. To specify that the menu item has a text encoding, small icon, or reduced icon, use the `SetItemCmd` function to set the keyboard equivalent field to 0x1C, 0x1D, or 0x1E, respectively.
(	Defines the menu item as disabled.

You can specify any, all, or none of these metacharacters in the text string. The metacharacters that you specify aren't displayed in the menu item. (To use any of these metacharacters in the text of a menu item, first use `AppendMenu`, specifying at least one character as the item's text, and then use the function `SetMenuItemText` (page 190) to set the item's text to the desired string.)

If you add menu items using the `AppendMenu` function, you should define the text and any marks or keyboard equivalents in resources for easier localization.

You can specify the first character that defines the text of a menu item as a hyphen to create a divider line. The string in the data parameter can be blank (containing one or more spaces), but it should not be an empty string.

If you do not define a specific characteristic of a menu item, the `AppendMenu` function assigns the default characteristic to the menu item. If you do not define any characteristic other than the text for a menu item, the `AppendMenu` function inserts the menu item so that it appears in the menu as an enabled item, without an icon or a mark, in the plain character style, and without a keyboard equivalent.

You can use `AppendMenu` to append items to a menu regardless of whether the menu is in the current menu list.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.5.
Not available to 64-bit applications.

Related Sample Code
Simple DrawSprocket

Declared In
`Menus.h`

AppendMenuItemText

Appends a menu item to a menu. (Deprecated in Mac OS X v10.5.)

Not recommended

```
OSStatus AppendMenuItemText (
   MenuRef menu,
   ConstStr255Param inString
);
```

Parameters

menu

The menu to which the item is to be appended.

inString

A Pascal string containing the text of the menu item to append. You can pass a string containing any characters, and these characters will be presented verbatim in the menu item.

Return Value
A result code. See "Menu Manager Result Codes" (page 161).

Discussion
Note that unless you are supporting legacy code, you should use the `AppendMenuItemTextWithCFString` (page 21) function instead.

The `AppendMenuItemText` function appends the menu item containing the specified string to a menu, without evaluating the string for metacharacters, as the pre–Mac OS 8.5 Menu Manager function `AppendMenu` does. You may wish to use `AppendMenuItemText` if you need to present non-alphanumeric characters in a menu item.

The appended menu item appears at the end of the menu as an enabled item. If you wish to place the menu item elsewhere than at the end of the menu you should use the function `InsertMenuItemText` (page 177).

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.5.
Not available to 64-bit applications.

Declared In
`Menus.h`

AppendResMenu

Searches all resource files open to your application for a given resource type and appends the names of any resources it finds to a specified menu. (Deprecated in Mac OS X v10.5.)

Not recommended

```
void AppendResMenu (
   MenuRef theMenu,
   ResType theType
);
```

Parameters

theMenu

The menu to which to append the names of any resources of a given type that `AppendResMenu` finds.

theType

A four-character code that identifies the resource type for which to search.

Discussion
Unless you must support legacy code, you should not use functions like `AppendResMenu` ,which assumes that menu items to append are stored in resources.

If you want to insert fonts into a menu, you should call `CreateStandardFontMenu` (page 35) instead.

The `AppendResMenu` function searches all resource files open to your application for resources of the type defined by the parameter `theType`. It appends the names of any resources it finds of the given type to the end of the specified menu. `AppendResMenu` appends the names of found resources in alphabetical order; it does not alphabetize items already in the menu. The `AppendResMenu` function does not add resources with names that begin with a period (.) or a percent sign (%) to the menu.

`AppendResMenu` assigns default characteristics to each menu item. Each appended menu item appears in the menu as an enabled item, without an icon or a mark, in the plain character style, and without a keyboard equivalent.

Note that for applications using CarbonLib, you no longer need to call `AppendResMenu` add resources of type `'DRVR '` to your Apple menu; CarbonLib does this for you automatically.

If you specify that `AppendResMenu` append resources of type `'FONT '` or `'FOND '`, the Menu Manager performs special processing for any resources it finds that have font numbers greater than 0x4000. If the script system associated with the font name is installed in the system, `AppendResMenu` stores information in the `itemDefinitions` array (in the `itemIcon` and `itemCmd` fields for that item) in the menu structure. This allows the Menu Manager to display the font name in the correct script.

Special Considerations

The `AppendResMenu` function calls the Resource Manager function `SetResLoad` (specifying `true` in the `load` parameter) before returning. The `AppendResMenu` function reads the resource data of the resources it finds into memory. If your application does not want the Resource Manager to read resource data into memory when your application calls other functions that read resources, you need to call `SetResLoad` and specify `false` in the `load` parameter after `AppendResMenu` returns.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.5.
Not available to 64-bit applications.

Related Sample Code
Simple DrawSprocket

Declared In
`Menus.h`

DeleteMCEntries

Deletes a menu item entry, a menu title entry, the menu bar entry, or all menu item entries of a specific menu from your application's menu color information table. (Deprecated in Mac OS X v10.5.)

Not recommended

```
void DeleteMCEntries (
   MenuID menuID,
   short menuItem
);
```

Parameters

menuID

The menu ID that the `DeleteMCEntries` function should use to determine which entry to delete from the menu color information table. Specify 0 in the `menuID` parameter (and the `menuItem` parameter) to delete the menu bar entry. Specify the menu ID of a menu in the current menu list in the `menuID` parameter and 0 in the `menuItem` parameter to delete a specific menu title entry. Specify the menu ID of a menu in the current menu list in the `menuID` parameter and an item number in the `menuItem` parameter to delete a specific menu item entry.

menuItem

The menu item that the `DeleteMCEntries` function should use to determine which entry to delete from the menu color information table. If you specify 0 in this parameter, `DeleteMCEntries` deletes either the menu bar entry or menu title entry, depending on the value of the `menuID` parameter. If you specify the item number of a menu item in this parameter and the menu ID of a menu in the current menu list in the `menuID` parameter, `DeleteMCEntries` deletes a specific menu item entry. You can also delete all menu item entries for a specific menu from your application's menu color information table by specifying the constant `mctAllItems`.

Discussion

If the `DeleteMCEntries` function does not find the specified entry in your application's menu color information table, it does not delete the entry. Your application should not delete the last entry in your application's menu color information table.

If any of the deleted entries changes the menu bar color or a menu title color, your application should call `DrawMenuBar` (page 41) to update the menu bar.

Carbon Porting Notes

`DeleteMCEntries` does nothing, because the Appearance Manager doesn't use color tables.

Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.5.

Not available to 64-bit applications.

Declared In

`Menus.h`

DisposeMCInfo

Disposes of a menu color information table. (Deprecated in Mac OS X v10.5.)

Not recommended

```
void DisposeMCInfo (
   MCTableHandle menuCTbl
);
```

Parameters

menuCTbl

The handle to the menu color information table you want to remove.

Discussion

The `DisposeMCInfo` function disposes of the menu color information table referred to by the `menuCTbl` parameter.

Carbon Porting Notes

`DisposeMCInfo` does nothing, because Appearance Manager doesn't use color tables.

Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.5.

Not available to 64-bit applications.

Declared In

`Menus.h`

DisposeMenuDefUPP

Disposes of universal procedure pointer to a custom menu definition. (Deprecated in Mac OS X v10.5.)

Not recommended

```
void DisposeMenuDefUPP (
   MenuDefUPP userUPP
);
```

Parameters

userUPP

Carbon Porting Notes

Apple discourages you from writing and using your own menu definition functions and encourages you to use the system-supplied menu definition function instead. New features that have previously been missing are now available in the system-supplied menu definition function. Since Appearance Manager 1.0 (in Mac OS 8.0), for example, the system-supplied menu definition function has supported extended menu item command key modifiers and glyphs. And in Carbon, the system-supplied menu definition function supports dynamic items, which allow the contents of a menu item to be redrawn while the menu is displayed in response to the user pressing a modifier key on the keyboard.

Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.5.

Not available to 64-bit applications.

Declared In

`Menus.h`

EraseMenuBackground

Erases the menu background to prepare for additional drawing. (Deprecated in Mac OS X v10.5.)

```
OSStatus EraseMenuBackground (
   MenuRef inMenu,
   const Rect *inEraseRect,
   CGContextRef inContext
);
```

Parameters

inMenu

The menu whose background you want to erase.

inEraseRect

The bounds of the area to erase, in the local coordinates of the current port.

inContext

The Core Graphics context to erase. If set to `NULL`, this function creates a new context based on the current port.

Return Value

A result code. See "Menu Manager Result Codes" (page 161).

Discussion

Typically you use this function only if you are implementing message-based custom menu definition functions. HIView-based custom menus and normal application code do not need to call `EraseMenuBackground`.

Before calling the Appearance Manager function `DrawThemeMenuBackground`, you must erase the current menu background. Themes such as Aqua draw the menu background using an alpha channel, so if the old background is not erased, portions of the old image will show through the menu background.

Availability

Available in Mac OS X v10.1 and later.

Deprecated in Mac OS X v10.5.

Not available to 64-bit applications.

Declared In

`Menus.h`

GetItemIcon

Returns a menu item's icon or text encoding. (Deprecated in Mac OS X v10.5.)

```
void GetItemIcon (
   MenuRef theMenu,
   MenuItemIndex item,
   short *iconIndex
);
```

Parameters

theMenu

The menu containing the item.

item

The menu index of the item.

iconIndex

On output, an integer representing the menu item's icon or text encoding. For menu items that do not specify 0x1C in the keyboard equivalent field, a value from 1 through 255 if the menu item has an icon associated with it and is 0 otherwise representing the item's icon number.

Discussion

In most cases, you should use `GetMenuItemTextEncoding` (page 173) rather than `GetItemIcon` to get the menu item's text encoding.

The `GetItemIcon` function returns the icon number or text encoding of the specified menu item through the `iconIndex` parameter (or 0 if the item doesn't have an icon or a text encoding). If the menu item's keyboard equivalent field contains 0x1C, the returned number represents the text encoding of the menu item. Otherwise, the returned number represents the item's icon number.

In the `iconIndex` parameter, you can add 256 to the icon number to generate the resource ID of the `'cicn'`, `'ICON '`, or `'SICN '` resource that describes the icon of the menu item. For example, if the `GetItemIcon` function returns 5 as the icon number, then the icon of the menu item is described by an icon resource with resource ID 261.

For menu items that contain 0x1C in the keyboard equivalent field, the `GetItemIcon` function returns the text encoding of the menu item. The Menu Manager displays the menu item using this text encoding if the corresponding script system is installed.

Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.5.

Not available to 64-bit applications.

Declared In

`Menus.h`

GetMCEntry

Gets information about an entry in an application's menu color information table. (Deprecated in Mac OS X v10.5.)

Not recommended

```
MCEntryPtr GetMCEntry (
   MenuID menuID,
   short menuItem
);
```

Parameters

menuID

The menu ID that the `GetMCEntry` function should use to return information about the menu color information table. Specify 0 in the `menuID` parameter (and the `menuItem` parameter) to get the menu bar entry. Specify the menu ID of a menu in the current menu list in the `menuID` parameter and 0 in the `menuItem` parameter to get a specific menu title entry. Specify the menu ID of a menu in the current menu list in the `menuID` parameter and an item number in the `menuItem` parameter to get a specific menu item entry.

menuItem

The menu item that the `GetMCEntry` function should use to return information about the menu color information table. If you specify 0 in this parameter, `GetMCEntry` returns either the menu bar entry or the menu title entry, depending on the value of the `menuID` parameter. If you specify the item number of a menu item in this parameter and the menu ID of a menu in the current menu list in the `menuID` parameter, `GetMCEntry` returns a specific menu item entry.

Return Value
A menu bar entry, a menu title entry, or a menu item entry according to the values specified in the `menuID` and `menuItem` parameters. If the `GetMCEntry` function finds the specified entry in your application's menu color information table, it returns a pointer to a structure of data type `MCEntry`. If the specified entry is not found, `GetMCEntry` returns `null`. The menu color information table is relocatable, so the pointer returned by the `GetMCEntry` function may not be valid across functions that may move or purge memory. Your application should make a copy of the menu color entry structure if necessary. See page for a description of the `MCEntry` data structure.

Carbon Porting Notes

`GetMCEntry` does nothing, because Appearance Manager doesn't use color tables.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.5.
Not available to 64-bit applications.

Declared In
`Menus.h`

GetMCInfo

Returns a handle to a copy of your application's menu color information table. (Deprecated in Mac OS X v10.5.)

Not recommended

```
MCTableHandle GetMCInfo (
   void
);
```

Return Value
A handle to a copy of your application's menu color information table. If the copy fails, `GetMCInfo` returns `null`.

Carbon Porting Notes

The Menu Manager ignores color tables in Mac OS X, so there is no reason to call `GetMCInfo`.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.5.
Not available to 64-bit applications.

Declared In
`Menus.h`

GetMenuItemHierarchicalID

Obtains the menu ID of a specified submenu. (Deprecated in Mac OS X v10.5.)

```
OSErr GetMenuItemHierarchicalID (
   MenuRef inMenu,
   MenuItemIndex inItem,
   MenuID *outHierID
);
```

Parameters

inMenu

The menu that contains the menu item for which you wish to get the submenu's menu ID.

inItem

The menu index of the item.

outHierID

Pass a pointer to a signed 16-bit integer value. On return, the value is set to the menu ID of the submenu.

Return Value

A result code. See "Menu Manager Result Codes" (page 161).

Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.5.

Not available to 64-bit applications.

Declared In

`Menus.h`

GetMenuItemText

Obtains the text of a menu item. (Deprecated in Mac OS X v10.5.)

Not recommended

```
void GetMenuItemText (
   MenuRef theMenu,
   MenuItemIndex item,
   Str255 itemString
);
```

Parameters

theMenu

The menu containing the item.

item

The menu index of the item.

itemString

On output, the menu item's text string.

Discussion

Unless you need to support legacy code, you should use the `CopyMenuItemTextAsCFString` (page 31) instead.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.5.
Not available to 64-bit applications.

Related Sample Code
CarbonSketch

Declared In
`Menus.h`

GetMenuItemTextEncoding

Obtains the text encoding used for a menu item's text. (Deprecated in Mac OS X v10.5.)

```
OSErr GetMenuItemTextEncoding (
   MenuRef inMenu,
   MenuItemIndex inItem,
   TextEncoding *outScriptID
);
```

Parameters

inMenu
The menu containing the menu item whose text encoding you wish to obtain.

inItem
The menu index of the item.

outScriptID
A pointer to a `TextEncoding` value. On return, the value is set to the script code of the text encoding used in the menu item's text.

Return Value
A result code. See "Menu Manager Result Codes" (page 161).

Discussion
If a menu item has a command code of 0x1C when `GetMenuItemTextEncoding` is called, `GetMenuItemTextEncoding` gets the value in the icon field of the menu item.

See also the function `SetMenuItemTextEncoding` (page 191).

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.5.
Not available to 64-bit applications.

Declared In
`Menus.h`

GetMenuRetainCount

Returns the retain count of this menu. (Deprecated in Mac OS X v10.5.)

```
ItemCount GetMenuRetainCount (
   MenuRef inMenu
);
```

Parameters

inMenu

The menu whose retain count you want to obtain.

Return Value
The retain count for the menu.

Version Notes
In Mac OS X v10.2 and later, all menus are Core Foundation CFTypes, so you can also call `CFGetRetainCount` instead of `GetMenuRetainCount`.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.5.
Not available to 64-bit applications.

Declared In
`Menus.h`

GetMenuTitle

Obtains the title of the menu (Deprecated in Mac OS X v10.5.)

Not recommended

```
StringPtr GetMenuTitle (
   MenuRef menu,
   Str255 title
);
```

Parameters

menu

title

The menu title, as a Str255 string.

Return Value
A pointer to the menu title.

Discussion
Unless you need to support legacy code, you should use the `CopyMenuTitleAsCFString` (page 32) function instead.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.5.
Not available to 64-bit applications.

Declared In
`Menus.h`

InitContextualMenus

Adds a program to the system registry of contextual menu clients. (Deprecated in Mac OS X v10.5.)

Not recommended

```
OSStatus InitContextualMenus (
   void
);
```

Return Value
A result code. See "Menu Manager Result Codes" (page 161).

Discussion
Your program should call the `InitContextualMenus` function early in your startup code to register your application as a contextual menu client. If you do not register your program, some system-level functions may respond as though your program does not use contextual menus. Not registering your program may also cause `ProcessIsContextualMenuClient` (page 183) to return an incorrect value.

Carbon Porting Notes

You do not need to call this function before using contextual menus in Mac OS X.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.5.
Not available to 64-bit applications.

Declared In
`Menus.h`

InsertFontResMenu

Inserts menu items from a font resource. (Deprecated in Mac OS X v10.5.)

Not recommended

```
void InsertFontResMenu (
   MenuRef theMenu,
   MenuItemIndex afterItem,
   short scriptFilter
);
```

Parameters

theMenu
The menu to add the fonts to.

afterItem
The menu item after which you want to add the fonts.

scriptFilter
The script filter you want to use.

Discussion
Unless you need to do script filtering, you should use `CreateStandardFontMenu` (page 35)instead.

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

Deprecated in Mac OS X v10.5.
Not available to 64-bit applications.

Declared In
`Menus.h`

InsertIntlResMenu

Inserts menu items from an internationalized resource. (Deprecated in Mac OS X v10.5.)

Not recommended

```
void InsertIntlResMenu (
   MenuRef theMenu,
   ResType theType,
   MenuItemIndex afterItem,
   short scriptFilter
);
```

Parameters

theMenu
The menu to add the menu items to.

theType
The resource to retrieve the menu items from.

afterItem
The menu item after which you want to add the new items.

scriptFilter
The script filter you want to use.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.5.
Not available to 64-bit applications.

Declared In
`Menus.h`

InsertMenuItem

Inserts one or more items into a menu previously created. (Deprecated in Mac OS X v10.5.)

Not recommended

```
void InsertMenuItem (
    MenuRef theMenu,
    ConstStr255Param itemString,
    short afterItem
);
```

Parameters

theMenu
The menu to which you wish to add the menu item or items.

itemString

A string that defines the characteristics of the new menu items. Note that in most cases you should store the text of a menu item in a nib file or resource, so that your menu items can be more easily localized. You can specify the contents of the `itemString` parameter using metacharacters; the function `InsertMenuItem` accepts the same metacharacters as the `AppendMenu` (page 163) function. However, if you specify multiple items, the `InsertMenuItem` function inserts the items in the reverse of their order in the `itemString` parameter.

afterItem

The item number of the menu item after which the new menu items are to be added. Specify 0 in the `afterItem` parameter to insert the new items before the first menu item; specify the item number of a current menu item to insert the new menu items after it; specify a number greater than or equal to the last item in the menu to append the new items to the end of the menu.

Discussion

Note that unless you are supporting legacy code, you should use the `InsertMenuItemTextWithCFString` (page 74) function instead.

If you do not define a specific characteristic of a menu item, the `InsertMenuItem` function assigns the default characteristic to the menu item. If you do not define any characteristic other than the text for a menu item, the `InsertMenuItem` function inserts the menu item so that it appears in the menu as an enabled item, without an icon or a mark, in the plain character style, and without a keyboard equivalent.

You can use `InsertMenuItem` to insert items into a menu regardless of whether the menu is in the current menu list.

Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.5.

Not available to 64-bit applications.

Declared In

`Menus.h`

InsertMenuItemText

Inserts a menu item into a menu. (Deprecated in Mac OS X v10.5.)

Not recommended

```
OSStatus InsertMenuItemText (
   MenuRef menu,
   ConstStr255Param inString,
   MenuItemIndex afterItem
);
```

Parameters

menu

The menu into which the item is to be inserted.

inString

A Pascal string containing the text of the menu item to insert. You can pass a string containing any characters, and these characters will be presented verbatim in the menu item.

afterItem

The item number of the menu item after which the new menu item is to be inserted. Specify 0 to insert the new menu item at the top of the menu, before the first menu item. Specify a value greater than or equal to the last menu item to append the new item to the end of the menu.

Return Value
A result code. See "Menu Manager Result Codes" (page 161).

Discussion

Note that unless you are supporting legacy code, you should use the `InsertMenuItemTextWithCFString` (page 74) function instead.

The `InsertMenuItemText` function inserts an enabled menu item containing the specified string into a menu, without evaluating the string for metacharacters, as the pre–Mac OS 8.5 Menu Manager function `InsertMenuItem` does. You may wish to use `InsertMenuItemText` if you have a need to present non-alphanumeric characters in a menu item.

See also the `AppendMenuItemText` (page 164) function.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.5.
Not available to 64-bit applications.

Declared In
`Menus.h`

InsertResMenu

Searches all resource files open to your application for a given resource type and inserts the names of any resources it finds in the specified menu. (Deprecated in Mac OS X v10.5.)

Not recommended

```
void InsertResMenu (
   MenuRef theMenu,
   ResType theType,
   MenuItemIndex afterItem
);
```

Parameters

theMenu

The menu to which to add the names of any resources of a given type that `InsertResMenu` finds.

theType

A four-character code that identifies the resource type for which to search.

afterItem

A number that indicates where in the menu to insert the names of any resources of the given type that `InsertResMenu` finds. Specify 0 in the `afterItem` parameter to insert the items before the first menu item; specify the item number of a menu item already in the menu to insert the items after the specified item number. If you specify a number greater than or equal to the last item in the menu, the items are inserted at the end of the menu.

Discussion
Unless you must support legacy code, you should not use functions like `InsertResMenu` that rely on searching the resource chain. Prior to Carbon, you used `InsertResMenu` primarily to create an Apple menu (by passing `'DRVR'` for the resource type or to create a font menu (by passing `'FONT'`). In Carbon, the Apple menu is created for you automatically, and you should call `CreateStandardFontMenu` (page 35) to create a font menu.

The `InsertResMenu` function searches all resource files open to your application for resources of the type defined by the parameter `theType`. The specified menu must have been previously created using `NewMenu` (page 182) , `GetMenu` (page 50) , or `GetNewMBar` (page 70). `InsertResMenu` adds the names of found resources after the specified menu item in alphabetical order; it does not alphabetize items already in the menu.

The `InsertResMenu` function does not add resources with names that begin with a period (.) or a percent sign (%) to the menu.

The `InsertResMenu` function assigns default characteristics to each menu item. Each appended menu item appears in the menu as an enabled item, without an icon or a mark, in the plain character style, and without a keyboard equivalent.

If you specify that `InsertResMenu` add resources of type `'DRVR '` to your Apple menu, `InsertResMenu` adds the name (and icon) of each item in the Apple Menu Items folder to the menu.

If you specify that `InsertResMenu` add resources of type `'FONT '` or `'FOND '`, the Menu Manager performs special processing for any resources it finds that have font numbers greater than 0x4000. If the script associated with the font name is currently active, `InsertResMenu` stores information in the `itemDefinitions` array (in the `itemIcon` and `itemCmd` fields for that item) in the menu structure that allows the Menu Manager to display the font name in the correct script.

Special Considerations

The `InsertResMenu` function calls the Resource Manager function `SetResLoad` (specifying `true` in the `load` parameter) before returning. The `InsertResMenu` function reads the resource data of the resources it finds into memory. If your application does not want the Resource Manager to read resource data into memory when your application calls other functions that read resources, you need to call the Resource Manager function `SetResLoad` and specify `false` in the `load` parameter after `InsertResMenu` returns.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.5.
Not available to 64-bit applications.

Declared In
`Menus.h`

InvokeMenuDefUPP

Calls your custom menu definition through a universal procedure pointer. (Deprecated in Mac OS X v10.5.)

Not recommended

```
void InvokeMenuDefUPP (
   short message,
   MenuRef theMenu,
   Rect *menuRect,
   Point hitPt,
   short *whichItem,
   MenuDefUPP userUPP
);
```

Parameters

message

theMenu

menuRect

hitPt

whichItem

userUPP

Carbon Porting Notes

Apple discourages you from writing and using your own menu definition functions and encourages you to use the system-supplied menu definition function instead. New features that have previously been missing are now available in the system-supplied menu definition function. Since Appearance Manager 1.0 (in Mac OS 8.0), for example, the system-supplied menu definition function has supported extended menu item command key modifiers and glyphs. And in Carbon, the system-supplied menu definition function supports dynamic items, which allow the contents of a menu item to be redrawn while the menu is displayed in response to the user pressing a modifier key on the keyboard.

Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.5.

Not available to 64-bit applications.

Declared In

`Menus.h`

MenuKey

Maps a character key with the command key to determine the keyboard equivalent of a menu item in a menu in the current menu list. (Deprecated in Mac OS X v10.5.)

Not recommended

```
SInt32 MenuKey (
   CharParameter ch
);
```

Parameters

ch

The 1-byte character that represents the key pressed by the user in combination with the Command key.

Return Value

A value containing the menu ID and menu item that corresponds to the given character.If the given character does not map to an enabled menu item in the current menu list, `MenuKey` returns 0 in its high-order word and the low-order word is undefined.

Discussion
Note that unless your application uses the `WaitNextEvent` event model, you should use the Carbon event–based `IsMenuKeyEvent` (page 80) instead. Even when using the `WaitNextEvent` model, you should probably use `MenuEvent` (page 84), as that function supports the Shift, Option, and Control modifier keys in addition to the Command key

The `MenuKey` function determines whether the key combination maps to a current menu item when the user presses another key while holding down the Command key.

`MenuKey` does not distinguish between uppercase and lowercase letters. This allows a user to invoke a keyboard equivalent command, such as the Copy command, by pressing the Command key and "c" or "C". For consistency between applications, you should define the keyboard equivalents of your commands so that they appear in uppercase in your menus.

You should not define menu items with identical keyboard equivalents. The `MenuKey` function scans the menus from right to left and the items from top to bottom. If you have defined more than one menu item with identical keyboard equivalents, `MenuKey` returns the first one it finds.

If the given character maps to an enabled menu item in the current menu list, `MenuKey` highlights the menu title of the chosen menu, returns the menu ID in the high-order word of its function result, and returns the chosen menu item in the low-order word of its function result. After performing the chosen task, your application should unhighlight the menu title using the `HiliteMenu` function.

The `MenuKey` function first searches the regular portion of the current menu list for a menu item with a keyboard equivalent matching the given key. If it doesn't find one there, it searches the submenu portion of the current menu list. If the given key maps to a menu item in a submenu, `MenuKey` highlights the menu title in the menu bar that the user would normally pull down to begin traversing to the submenu. Your application should perform the desired command and then unhighlight the menu title.

You shouldn't assign a Command–Shift–number key sequence to a menu item as its keyboard equivalent Command–Shift–number key sequences are reserved for use as `'FKEY '` resources. Command–Shift–number key sequences are not returned to your application, but instead are processed by the Event Manager. The Event Manager invokes the `'FKEY '` resource with a resource ID that corresponds to the number that activates it.

Apple reserves the Command-key codes 0x1B through 0x20 to indicate meanings other than keyboard equivalents. `MenuKey` ignores these character codes and returns a function result of 0 if you specify any of these values in the `ch` parameter. Your application should not use these character codes for its own use

Carbon Porting Notes

Carbon does not support desk accessories, so `MenuKey` cannot be used in OS X to pass keyboard equivalents to desk accessories.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.5.
Not available to 64-bit applications.

Related Sample Code
Simple DrawSprocket

Declared In
`Menus.h`

NewMenu

Creates an empty menu with a specified title and menu ID. (Deprecated in Mac OS X v10.5.)

Not recommended

```
MenuRef NewMenu (
   MenuID menuID,
   ConstStr255Param menuTitle
);
```

Parameters

menuID

The ID for the menu. The menu ID is a number that identifies the menu. Menu IDs in Carbon can be any value, but Apple recommends that the ID be either zero or positive. A menu ID of zero is a valid ID. IDs of submenus should similarly be zero or a positive value.

menuTitle

The title of the new menu. Note that in most cases you should store the titles of menus in resources, so that your menu titles can be more easily localized.

Return Value

A menu reference. If the `NewMenu` function is unable to create the menu structure, it returns `NULL`. See the description of the `MenuRef` data type.

Discussion

Unless you must support legacy code, you should not use functions like `NewMenu`. If you need to create menus programmatically, you can call `CreateNewMenu` (page 34); otherwise you should define menus in Interface Builder, store them as nib files, and then call the Interface Builder Services function `CreateMenuFromNib` to create them.

The `NewMenu` function creates a menu with the specified title, assigns it the specified menu ID, and returns a handle to the menu structure. It sets up the menu structure to use the standard menu definition function (and it reads the standard menu definition function into memory if it isn't already there). `NewMenu` does not insert the newly created menu into the current menu list.

After creating a menu with `NewMenu`, use `AppendMenu` (page 163) , `InsertMenuItem` (page 176) , `AppendResMenu` (page 165) , or `InsertResMenu` (page 178) to add menu items to the menu. To add a menu created by `NewMenu` to the current menu list, use `InsertMenu` (page 73). In Carbon, you do not need to call `DrawMenuBar` (page 41) to update the menu bar, as the Menu Manager automatically invalidates andredraws the menu bar.

Menus in a resource must not be purgeable nor should the resource lock bit be set. Do not define a "circular" hierarchical menu—that is, a hierarchical menu in which a submenu has a submenu whose submenu is a hierarchical menu higher in the chain.

Special Considerations

To release the memory associated with a menu that you created using `NewMenu`, first call `DeleteMenu` (page 36) to remove the menu from the current menu list and to remove any entries for this menu in your application's menu color information table then call `DisposeMenu` (page 40) to dispose of the menu structure. Note that in Carbon, the Menu Manager automatically invalidates and redraws the menu bar after disposing of a menu.

Version Notes

Note that if you are running on Mac OS 8.1 and earlier, the menu ID of a submenu must be within the range 0 to 255.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.5.
Not available to 64-bit applications.

Related Sample Code
Simple DrawSprocket

Declared In
`Menus.h`

NewMenuDefUPP

Creates a new univeral procedure pointer to your custom menu definition. (Deprecated in Mac OS X v10.5.)

Not recommended

```
MenuDefUPP NewMenuDefUPP (
   MenuDefProcPtr userRoutine
);
```

Parameters
userRoutine

Return Value
See the description of the `MenuDefUPP` data type.

Carbon Porting Notes

Apple discourages you from writing and using your own menu definition functions and encourages you to use the system-supplied menu definition function instead. New features that have previously been missing are now available in the system-supplied menu definition function. Since Appearance Manager 1.0 (in Mac OS 8.0), for example, the system-supplied menu definition function has supported extended menu item command key modifiers and glyphs. And in Carbon, the system-supplied menu definition function supports dynamic items, which allow the contents of a menu item to be redrawn while the menu is displayed in response to the user pressing a modifier key on the keyboard.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.5.
Not available to 64-bit applications.

Declared In
`Menus.h`

ProcessIsContextualMenuClient

Determines whether a given program is a contextual menu client. (Deprecated in Mac OS X v10.5.)

Not recommended

```
Boolean ProcessIsContextualMenuClient (
   ProcessSerialNumber *inPSN
);
```

Parameters

inPSN

A pointer to the ID of the process containing the program.

Return Value

`true` if the program in the process uses contextual menus; otherwise, `false`.

Discussion

In Mac OS X, you do not need to call `InitContextualMenus` (page 175) to register your application, so this function call is unnecessary.

The `ProcessIsContextualMenuClient` function checks the system registry of contextual menu clients and returns `true` if the program in the given process supports contextual menus. However, the program must have been registered as a client using `InitContextualMenus` (page 175).

See also "Contextual Menu Gestalt Selector Constants" (page 131).

Version Notes

This function is available with Appearance Manager 1.0 and later.

Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.5.

Not available to 64-bit applications.

Declared In

`Menus.h`

ReleaseMenu

Decrements the retain count of a menu. (Deprecated in Mac OS X v10.5.)

```
OSStatus ReleaseMenu (
   MenuRef inMenu
);
```

Parameters

inMenu

The menu whose retain count to decrement. If the retain count falls to zero, the menu is destroyed.

Return Value

A result code. See "Menu Manager Result Codes" (page 161).

Discussion

The reference that you pass in the `theMenu` parameter is not valid after `DisposeMenu` returns. This function is identical to `DisposeMenu` (page 40).

Version Notes

In Mac OS X v10.2 and later, all menus are Core Foundation CFTypes, so you can optionally call `CFRelease` instead of `ReleaseMenu`.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.5.
Not available to 64-bit applications.

Related Sample Code
QTCarbonShell

Declared In
`Menus.h`

RetainMenu

Increments the reference count of a menu. (Deprecated in Mac OS X v10.5.)

```
OSStatus RetainMenu (
   MenuRef inMenu
);
```

Parameters

inMenu

The menu whose reference count you want to increment.

Return Value
A result code. See "Menu Manager Result Codes" (page 161).

Discussion
`RetainMenu` does not create a new menu. It simply adds one to the reference count.

Version Notes
In Mac OS X v10.2 and later, all menus are Core Foundation CFTypes, so you can optionally call `CFRetain` instead of `RetainMenu`.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.5.
Not available to 64-bit applications.

Declared In
`Menus.h`

ScrollMenuImage

Scrolls a portion of the menu image. (Deprecated in Mac OS X v10.5.)

```
OSStatus ScrollMenuImage (
   MenuRef inMenu,
   const Rect *inScrollRect,
   int inHScroll,
   int inVScroll,
   CGContextRef inContext
);
```

Parameters

inMenu

The menu to scroll.

inScrollRect

The bounds of the area to scroll.

inHScroll

The distance to scroll horizontally, in pixels.

inVScroll

The distance to scroll vertically, in pixels.

inContext

The Core Graphics context to scroll. If you pass `NULL`, the function creates a context based on the current port.

Return Value

A result code. See "Menu Manager Result Codes" (page 161).

Discussion

Scrolling menus on Mac OS X using `ScrollRect` or other QuickDraw functions destroys the alpha channel data, so you should use `ScrollMenuImage` instead.

Availability

Available in Mac OS X v10.1 and later.

Deprecated in Mac OS X v10.5.

Not available to 64-bit applications.

Declared In

`Menus.h`

SetItemIcon

Sets a menu item's icon or text encoding. (Deprecated in Mac OS X v10.5.)

```
void SetItemIcon (
   MenuRef theMenu,
   MenuItemIndex item,
   short iconIndex
);
```

Parameters

theMenu

The menu containing the item.

item

The menu index of the item.

iconIndex

An integer representing the icon number or text encoding for the specified menu item. If the menu item's keyboard equivalent field does not contain 0x1C, the `SetItemIcon` function sets the icon number of the item's icon to the number defined in this parameter. The icon number you specify should be a value from 1 through 255 (or from 1 through 254 if the item has a small or reduced icon) or 0 if the item does not have an icon.

The Menu Manager adds 256 to the icon number to generate the resource ID of the '`cicn`' or '`ICON`' resource that describes the icon of the menu item. For example, if you specify 5 as the value of the `iconIndex` parameter, when the Menu Manager needs to draw the item, it looks for an icon resource with resource ID 261.

If the menu item's keyboard equivalent field contains 0x1C, the `SetItemIcon` function sets the text encoding of the menu item to the number defined in the `iconIndex` parameter. The Menu Manager displays the menu item using the specified text encoding if the corresponding script system is installed.

You can specify 0 in the `iconIndex` parameter to indicate that the item uses the current system script and does not have an icon number.

Discussion

In most cases, you should use `SetMenuItemTextEncoding` (page 191) rather than `SetItemIcon` to set the menu item's text encoding.

The `SetItemIcon` function sets the icon number (for resource-based icons) or text encoding of the specified menu item to the value in the `iconIndex` parameter. To use handle-based icons, call `SetMenuItemIconHandle` (page 102). Usually you display menu items in the current system script; however, if needed, you can use the `SetItemIcon` function to set the text encoding of a menu item. For an item's text encoding to be set, the keyboard equivalent field of the item must contain 0x1C. If the keyboard equivalent field contains any other value, the `SetItemIcon` function interprets the specified number as the item's icon number.

Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.5.

Not available to 64-bit applications.

Declared In

`Menus.h`

SetMCEntries

Sets entries in an application's menu color information table. (Deprecated in Mac OS X v10.5.)

Not recommended

```
void SetMCEntries (
   short numEntries,
   MCTablePtr menuCEntries
);
```

Parameters

numEntries

The number of entries contained in the array of menu color entry structures.

menuCEntries

A pointer to an array of menu color entry structures. Specify the number of structures in the array in the `numEntries` parameter.

Discussion

The `SetMCEntries` function sets any specified menu bar entry, menu title entry, or menu item entry according to the values specified in the menu color entry structures. If an entry already exists for a specified menu color entry, the `SetMCEntries` function updates the entry in your application's menu color information table with the new values. If the entry doesn't exist, it is added to your application's menu color information table.

If any of the added entries specify a new menu bar color or new menu title colors, your application should call `DrawMenuBar` (page 41) to update the menu bar with the new colors.

Special Considerations

The `SetMCEntries` function may move or purge memory. Your application should make sure that the array specified by the `menuCEntries` parameter is nonrelocatable before calling `SetMCEntries`.

Carbon Porting Notes

`SetMCEntries` does nothing, because Appearance Manager doesn't use color tables.

Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.5.

Not available to 64-bit applications.

Declared In

`Menus.h`

SetMCInfo

Makes a copy of your application's menu color information table. (Deprecated in Mac OS X v10.5.)

Not recommended

```
void SetMCInfo (
   MCTableHandle menuCTbl
);
```

Parameters

menuCTbl

A handle to a copy of your application's menu color information table.

Discussion

The `SetMCInfo` function copies the table specified by the `menuCTbl` parameter to your application's menu color information table. If successful, the `SetMCInfo` function is responsible for disposing of your application's current menu color information table, so your application does not need to explicitly dispose of the current table.

Your application should call the Memory Manager function `MemError` to determine whether the `SetMCInfo` function successfully copied the table. If the `SetMCInfo` function cannot successfully copy the table, it does not dispose of the current menu color information table and the `MemError` function returns a nonzero result code. If the `SetMCInfo` function is able to successfully copy the table, it disposes of the current menu color information table and the `MemError` function returns the `noErr` result code.

If the menu color information table specifies a new menu bar color or new menu title colors, your application should call `DrawMenuBar` (page 41) after calling `SetMCInfo`.

Note that `GetNewMBar` (page 70) does not save your application's current menu color information table. If your application changes menu bars, you can save and restore your application's current menu color information table by calling `GetMCInfo` (page 171) before `GetNewMBar` and calling `SetMCInfo` afterward.

Carbon Porting Notes

`SetMCInfo` does nothing, because Appearance Manager doesn't use color tables.

Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.5.

Not available to 64-bit applications.

Declared In

`Menus.h`

SetMenuFlashCount

Specifies whether a menu should fade slowly or immediately disappear when closing. (Deprecated in Mac OS X v10.5.)

Modified

```
void SetMenuFlashCount (
   short count
);
```

Parameters

count

See the Discussion.

Discussion

Unlike the classic Mac OS version, `SetMenuFlashCount` no longer lets you set the flash count for a menu. In Mac OS X, the flash count is always 1, and it not user adjustable.

Passing zero for the `count` parameter causes menus to disappear immediately when closing; passing any nonzero parameter causes the menus to fade out when closing (the default).

Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.5.

Not available to 64-bit applications.

Declared In

`Menus.h`

SetMenuItemHierarchicalID

Attaches a submenu to a menu item. (Deprecated in Mac OS X v10.5.)

```
OSErr SetMenuItemHierarchicalID (
   MenuRef inMenu,
   MenuItemIndex inItem,
   MenuID inHierID
);
```

Parameters

inMenu

The menu that contains the menu item to which you wish to attach a submenu.

inItem

The menu index of the item.

inHierID

An integer representing the menu ID of the submenu you wish to attach. This menu should be inserted into the menu list by calling `InsertMenu` with the `kInsertHierarchicalMenu` constant.

Return Value
A result code. See "Menu Manager Result Codes" (page 161).

Discussion
The `SetMenuItemHierarchicalID` function should be called instead of setting the keyboard equivalent to 0x1B. You should call `SetMenuItemHierarchicalID` instead of `SetItemMark` to set the menu ID of a menu item's submenu. However, you can still use `SetItemMark` to set the mark of a menu item.

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

Deprecated in Mac OS X v10.5.

Not available to 64-bit applications.

Declared In
`Menus.h`

SetMenuItemText

Sets menu item text to a specified string. (Deprecated in Mac OS X v10.5.)

Not recommended

```
void SetMenuItemText (
   MenuRef theMenu,
   MenuItemIndex item,
   ConstStr255Param itemString
);
```

Parameters

theMenu

The menu containing the item.

item

The menu index of the item.

itemString

The menu item's text string. This parameter must not be `NULL` or an empty (zero-length) string. Do not use meta–font characters in this parameter.

Your menu item text string must be limited to 250 bytes. 251 or more bytes will cause the Menu Manager to crash.

Discussion
The `SetMenuItemText` function does not recognize any metacharacters used by `AppendMenu` (page 163) and `InsertMenuItem` (page 176).

If you set the text of a menu item using the `SetMenuItemText` function, you should store the text in a string resource so that your application can be more easily localized.

Carbon Porting Notes

n Carbon, you should use the `SetMenuItemTextWithCFString` (page 107) function instead.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.5.
Not available to 64-bit applications.

Declared In
`Menus.h`

SetMenuItemTextEncoding

Sets the text encoding for a menu item's text. (Deprecated in Mac OS X v10.5.)

```
OSErr SetMenuItemTextEncoding (
   MenuRef inMenu,
   MenuItemIndex inItem,
   TextEncoding inScriptID
);
```

Parameters

inMenu

The menu containing the menu item whose text encoding you wish to set.

inItem

The menu index of the item.

inScriptID

The script code that corresponds to the text encoding you wish to set.

Return Value
A result code. See "Menu Manager Result Codes" (page 161).

Discussion
If a menu item has a command code of 0x1C when `SetMenuItemTextEncoding` is called, the values in the menu item's command key and icon ID fields are cleared and replaced with the value in the `inScriptID` parameter of `SetMenuItemTextEncoding`.

See also the function `GetMenuItemTextEncoding` (page 173).

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

Deprecated in Mac OS X v10.5.
Not available to 64-bit applications.

Declared In
`Menus.h`

SetMenuTitle

Sets the title of a menu. (Deprecated in Mac OS X v10.5.)

Not Recommended

```
OSStatus SetMenuTitle (
   MenuRef menu,
   ConstStr255Param title
);
```

Parameters

menu
The menu whose title you want to set.

title
A string containing the menu title to set.

Return Value
A result code. See "Menu Manager Result Codes" (page 161).

Discussion
Unless you need to support legacy code, you should use the `SetMenuTitleWithCFString` (page 108) function instead.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.5.
Not available to 64-bit applications.

Declared In
`Menus.h`

Document Revision History

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

Date	Notes
2006-09-15	Updated for Mac OS X v10.5.
2005-07-07	Fixed incorrect cross-references.
2004-02-26	Major revision
	Updated documentation for all functions, types, and constants. Removed older information not relevant to Mac OS X applications.
	Renamed and reorganized function groups. Moved superseded or otherwise not recommended functions into a separate group.
	Custom message-based menu definitions are now "Not recommended" in favor of HIView-based menu definitions.
2003-07-01	Expanded discussion for `SetMenuItemHierarchicalMenu` (page 101) function.
2003-02-01	Updated formatting and linking. Moved Porting Notes for unsupported functions to an appendix.

Document Revision History

Index

H

I

K

L

M

N

O

P

R

S

T

U