Icon Services and Utilities Reference

Carbon > User Experience

2007-04-06

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

Simultaneously published in the United States and Canada.

Contents

Icon Services and Utilities Reference

Framework:	Carbon/Carbon.h
Declared in	Icons.h IconsCore.h

Overview

The Icon Utilities allow your application (and system software) to manipulate and draw icons of any standard resource type in windows and if necessary in menus or dialog boxes. You need to use these routines only if you wish to draw icons in your application's windows or to draw icons whose resource types are not recognized by the Menu Manager and Dialog Manager in menus and dialog boxes.

To display an icon most effectively at a variety of sizes and bit depths, you should provide an icon family. You can then draw the appropriate member of the family for a given size and bit depth either by passing the family's resource ID to an Icon Utilities routine or by reading the family's icon resources into memory as an icon suite and passing the suite's handle to Icon Utilities routines.

Icon Services provides icon data to multiple Mac OS clients, including the Finder, extensions and applications. Using Icon Services to obtain icon data means you can provide efficient icon caching and release memory when you don't need icon data any longer. Icon Services provides the appropriate icon for any file object (file, folder, or volume), as well as other commonly used icons such as caution, note, or help icons in alert boxes, for example. The icons provided by Icon Services support a much larger palette of colors: up to 24 bits per pixel and an eight-bit mask. Icons are Appearance-compliant and appropriate to the active theme.

Functions by Task

Converting an Icon Mask to a Region

`IconIDToRgn` (page 72) Deprecated in Mac OS X v10.5
: Converts the icon mask in an icon family to a region. (Deprecated. Use Icon Services instead.)

`IconMethodToRgn` (page 73) Deprecated in Mac OS X v10.5
: Converts, to a region, the mask for an icon that `IconMethodToRgn` obtains with the aid of your icon getter callback function. (Deprecated. Use Icon Services instead.)

`IconSuiteToRgn` (page 76) Deprecated in Mac OS X v10.5
: Converts the icon mask in an icon suite to a region. (Deprecated. Use Icon Services instead.)

Creating an Icon Suite

`AddIconToSuite` (page 61) Deprecated in Mac OS X v10.5
: Adds an icon to an icon suite. (Deprecated. Use Icon Services instead.)

`GetIconSuite` (page 69) Deprecated in Mac OS X v10.5
: Creates an icon suite in memory that contains handles to a specified icon family's resources. (Deprecated. Use Icon Services instead.)

`NewIconSuite` (page 79) Deprecated in Mac OS X v10.5
: Gets a handle to an empty icon suite. (Deprecated. Use Icon Services instead.)

Determining Whether a Point Is Within an Icon

`PtInIconID` (page 89) Deprecated in Mac OS X v10.5
: Determines whether a specified point is within an icon. (Deprecated. Use Icon Services instead.)

`PtInIconMethod` (page 89) Deprecated in Mac OS X v10.5
: Determines whether a specified point is within an icon obtained with the aid of your icon getter callback function. (Deprecated. Use Icon Services instead.)

`PtInIconSuite` (page 92) Deprecated in Mac OS X v10.5
: Determines whether a specified point is within an icon. (Deprecated. Use Icon Services instead.)

Determining Whether a Rectangle Intersects an Icon

`RectInIconID` (page 93) Deprecated in Mac OS X v10.5
: Hit-tests a rectangle against the appropriate icon mask from an icon family for a specified destination rectangle and alignment. (Deprecated. Use Icon Services instead.)

`RectInIconMethod` (page 94) Deprecated in Mac OS X v10.5
: Hit-tests a rectangle against an icon obtained by your icon getter callback function for a specified destination rectangle and alignment. (Deprecated. Use Icon Services instead.)

`RectInIconSuite` (page 96) Deprecated in Mac OS X v10.5
: Hit-tests a rectangle against the appropriate icon mask from an icon suite for a specified destination rectangle and alignment. (Deprecated. Use Icon Services instead.)

Disposing of Icon Suites

`DisposeIconSuite` (page 62) Deprecated in Mac OS X v10.5
: Releases the memory occupied by an icon suite. (Deprecated. Use Icon Services instead.)

Disposing of Icons

`DisposeCIcon` (page 62) Deprecated in Mac OS X v10.5
: Releases the memory occupied by a color icon structure. (Deprecated. Use Icon Services instead.)

Drawing Icons From an Icon Suite

`PlotIconSuite` (page 86) Deprecated in Mac OS X v10.5
: Draws the icon described by an icon suite using the most appropriate icon in the suite for the current bit depth of the display device and the rectangle in which the icon is to be drawn. (Deprecated. Use Icon Services instead.)

Drawing Icons From Resources

`PlotCIcon` (page 80) Deprecated in Mac OS X v10.5
: Draws a color icon of resource type `'cicn'` to which you have a handle. (Deprecated. Use Icon Services instead.)

`PlotCIconHandle` (page 81) Deprecated in Mac OS X v10.5
: Draws an icon of resource type `'cicn'` to which you have a handle. (Deprecated. Use Icon Services instead.)

`PlotIcon` (page 82) Deprecated in Mac OS X v10.5
: Draws an icon of resource type `'ICON'` to which you have a handle. (Deprecated. Use Icon Services instead.)

`PlotIconHandle` (page 82) Deprecated in Mac OS X v10.5
: Draws an icon of resource type `'ICON'` or `'ICN#'` to which you have a handle. (Deprecated. Use Icon Services instead.)

`PlotIconID` (page 83) Deprecated in Mac OS X v10.5
: Draws the icon described by an icon family. (Deprecated. Use Icon Services instead.)

`PlotIconMethod` (page 85) Deprecated in Mac OS X v10.5
: Draws an icon obtained with the aid of your icon getter callback function. (Deprecated. Use Icon Services instead.)

`PlotSICNHandle` (page 88) Deprecated in Mac OS X v10.5
: Draws a small icon of resource type `'SICN'` to which you have a handle. (Deprecated. Use Icon Services instead.)

Enabling and Disabling Custom Icons

`GetCustomIconsEnabled` (page 16)
: Determines whether custom icons are enabled or disabled on a specified volume.

`SetCustomIconsEnabled` (page 34)
: Enables or disables custom icons on a specified volume.

Flushing IconRef Data

`FlushIconRefs` (page 59) Deprecated in Mac OS X v10.3
: Reclaims memory used by the specified icon if the memory is purgeable. (Deprecated. There is no replacement; this function was included to facilitate porting classic applications to Carbon, but it serves no useful purpose in Mac OS X.)

`FlushIconRefsByVolume` (page 59) Deprecated in Mac OS X v10.3
On a given volume, reclaims memory used by purgeable icons. (Deprecated. There is no replacement; this function was included to facilitate porting classic applications to Carbon, but it serves no useful purpose in Mac OS X.)

Getting and Setting the Label for an Icon Suite

`GetSuiteLabel` (page 71) Deprecated in Mac OS X v10.5
Gets the default label setting associated with an icon suite. (Deprecated. Use Icon Services instead.)

`SetSuiteLabel` (page 100) Deprecated in Mac OS X v10.5
Specifies the default label associated with an icon suite. (Deprecated. Use Icon Services instead.)

Getting Label Information

`GetLabel` (page 70) Deprecated in Mac OS X v10.5
Gets the color and string used for a given label in the Label menu of the Finder and in the Labels control panel. (Deprecated. Use Icon Services instead.)

Getting Icons From an Icon Suite

`GetIconFromSuite` (page 67) Deprecated in Mac OS X v10.5
Gets an icon from an icon suite. (Deprecated. Use Icon Services instead.)

Getting Icons From Resources That Don't Belong to an Icon Family

`GetCIcon` (page 64) Deprecated in Mac OS X v10.5
Gets a handle to a color icon of resource type `'cicn'`. (Deprecated. Use Icon Services instead.)

`GetIcon` (page 65) Deprecated in Mac OS X v10.5
Gets a handle to an icon resource of type `'ICON'`. (Deprecated. Use Icon Services instead.)

IconRef Reference Counting

`AcquireIconRef` (page 14)
Increments the reference count for an `IconRef`.

`GetIconRefOwners` (page 22)
Provides the current reference count for an `IconRef`.

`ReleaseIconRef` (page 33)
Decrements the reference count for an `IconRef`.

Modifying IconRef Data

`OverrideIconRef` (page 30)
Replaces the bitmaps of one `IconRef` with those of another `IconRef`.

`RemoveIconRefOverride` (page 33)
Restores the original bitmaps of an overridden `IconRef`.

`UpdateIconRef` (page 36)
Forces an update of `IconRef` data.

`OverrideIconRefFromResource` (page 79) Deprecated in Mac OS X v10.5
Replaces the bitmaps in an `IconRef` with bitmaps from a specified resource file. (Deprecated. Use `OverrideIconRef` (page 30) instead.)

Obtaining Icon Data

`IsDataAvailableInIconRef` (page 27)
Indicates whether an `IconRef` has the specified data.

`IsIconRefComposite` (page 27)
Reports whether a specified `IconRef` has been composited.

`IsIconRefMaskEmpty` (page 28)
Reports whether a specified mask is empty.

`IsValidIconRef` (page 29)
Reports whether a specified `IconRef` is valid.

`GetIconSizesFromIconRef` (page 60) Deprecated in Mac OS X v10.3
Provides an `IconSelectorValue` indicating the sizes and depths of icon data available for an `IconRef`. (Deprecated. Use `IsDataAvailableInIconRef` (page 27) instead.)

Obtaining IconRef Values

`GetIconRef` (page 17)
Provides an `IconRef` object for an icon in the desktop database or for a registered icon.

`GetIconRefFromFolder` (page 19)
Provides an `IconRef` object for a folder with no custom icon.

`GetIconRefFromFileInfo` (page 18)
Provides an `IconRef` object for a file with minimal file I/O.

`GetIconRefFromTypeInfo` (page 21)
Provides an `IconRef` object with the specified type information.

`GetIconRefFromIconFamilyPtr` (page 20)
Provides an `IconRef` object from a specified icon family.

`GetIconRefFromComponent` (page 18)
Provides an `IconRef` object based on a specified component.

`GetIconRefFromFile` (page 68) Deprecated in Mac OS X v10.5
Provides an `IconRef` object for a file, folder or volume. (Deprecated. Use `GetIconRefFromFileInfo` (page 18) instead.)

Performing Operations on Icons in an Icon Suite

`ForEachIconDo` (page 63) Deprecated in Mac OS X v10.5
: Performs an action on one or more icons in an icon suite. (Deprecated. Use Icon Services instead.)

Reading, Copying, and Converting Icon Data

`GetIconFamilyData` (page 16)
: Obtains a copy of the raw icon data for an individual element in an icon family.

`IconRefToIconFamily` (page 25)
: Provides icon family data for a given `IconRef`.

`ReadIconFromFSRef` (page 31)
: Reads an icon ('`icns`') file into memory.

`SetIconFamilyData` (page 34)
: Provides new raw icon data for an individual element of an icon family.

`IconFamilyToIconSuite` (page 71) Deprecated in Mac OS X v10.5
: Provides icon suite data for a given icon family. (Deprecated. Use Icon Services instead.)

`IconSuiteToIconFamily` (page 75) Deprecated in Mac OS X v10.5
: Provides `IconFamily` data for a specified `IconSuite`. (Deprecated. Use Icon Services instead.)

`ReadIconFile` (page 92) Deprecated in Mac OS X v10.5
: Copies data from a given file into an icon family. (Deprecated. Use `ReadIconFromFSRef` (page 31) instead.)

`WriteIconFile` (page 101) Deprecated in Mac OS X v10.5
: Copies data from a given icon family into a file. (Deprecated. Use the File Manager instead.)

Registering and Unregistering IconRef Values

`RegisterIconRefFromFSRef` (page 32)
: Registers an `IconRef` from a `.icns` file and associates it with a creator and type pair.

`RegisterIconRefFromIconFamily` (page 32)
: Adds an `iconFamily`-derived `IconRef` to the Icon Services registry.

`UnregisterIconRef` (page 35)
: Removes the specified icon data from the icon registry.

`RegisterIconRefFromIconFile` (page 97) Deprecated in Mac OS X v10.5
: Adds a file-derived `IconRef` to the Icon Services registry. (Deprecated. Use `RegisterIconRefFromFSRef` (page 32) instead.)

`RegisterIconRefFromResource` (page 98) Deprecated in Mac OS X v10.5
: Adds a resource-derived `IconRef` to the Icon Services registry. (Deprecated. Use `RegisterIconRefFromFSRef` (page 32) instead.)

Using IconRef Data

`CompositeIconRef` (page 14)
Superimposes one `IconRef` onto another.

`GetIconRefVariant` (page 22)
Specifies a transformation for a given `IconRef`.

`IconRefContainsCGPoint` (page 23)
Returns a Boolean value indicating whether an icon contains a specified point.

`IconRefIntersectsCGRect` (page 24)
Returns a Boolean value indicating whether an icon intersects a specified rectangle.

`IconRefToHIShape` (page 25)
Converts an icon into an HIShape object.

`PlotIconRefInContext` (page 30)
Plots an `IconRef` using Quartz.

`IconRefToRgn` (page 74) Deprecated in Mac OS X v10.5
Converts an Icon Services icon into a QuickDraw region. (Deprecated. Use `IconRefToHIShape` (page 25) instead.)

`PlotIconRef` (page 86) Deprecated in Mac OS X v10.5
Draws an icon using appropriate size and depth data from an `IconRef`. (Deprecated. Use `PlotIconRefInContext` (page 30) instead.)

`PtInIconRef` (page 91) Deprecated in Mac OS X v10.5
Tests whether a specified point falls within an icon's mask. (Deprecated. Use `IconRefContainsCGPoint` (page 23) instead.)

`RectInIconRef` (page 95) Deprecated in Mac OS X v10.5
Tests whether a specified rectangle falls within an icon's mask. (Deprecated. Use `IconRefIntersectsCGRect` (page 24) instead.)

Working With Icon Caches

`GetIconCacheData` (page 66) Deprecated in Mac OS X v10.5
Gets the data associated with an icon cache. (Deprecated. Use Icon Services instead.)

`GetIconCacheProc` (page 66) Deprecated in Mac OS X v10.5
Gets the icon getter function associated with an icon cache. (Deprecated. Use Icon Services instead.)

`LoadIconCache` (page 77) Deprecated in Mac OS X v10.5
Loads into an icon cache a handle to the appropriate icon data for a specified destination rectangle and the current bit depth, for drawing later with a specified alignment and transform. (Deprecated. Use Icon Services instead.)

`MakeIconCache` (page 78) Deprecated in Mac OS X v10.5
Gets a handle to an empty icon cache. (Deprecated. Use Icon Services instead.)

`SetIconCacheData` (page 99) Deprecated in Mac OS X v10.5
Sets the data associated with an icon cache. (Deprecated. Use Icon Services instead.)

`SetIconCacheProc` (page 100) Deprecated in Mac OS X v10.5
Sets the icon getter callback function associated with an icon cache. (Deprecated. Use Icon Services instead.)

Creating and Managing Universal Procedure Pointers

`NewIconActionUPP` (page 29)
Creates a new universal procedure pointer (UPP) to an icon action callback function.

`NewIconGetterUPP` (page 29)
Creates a new universal procedure pointer (UPP) to an icon getter callback function.

`DisposeIconActionUPP` (page 15)
Disposes of the universal procedure pointer (UPP) to your icon action callback function.

`DisposeIconGetterUPP` (page 15)
Disposes of the universal procedure pointer (UPP) to your icon getter callback function.

`InvokeIconActionUPP` (page 26)
Calls your icon action callback function.

`InvokeIconGetterUPP` (page 26)
Calls your icon getter callback function.

Functions

AcquireIconRef

Increments the reference count for an `IconRef`.

```
OSErr AcquireIconRef (
   IconRef theIconRef
);
```

Parameters

`theIconRef`
An `IconRef` whose reference count you wish to increment.

Return Value
A result code. See "Icon Services and Utilities Result Codes" (page 57).

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

Declared In
`IconsCore.h`

CompositeIconRef

Superimposes one `IconRef` onto another.

```
OSErr CompositeIconRef (
   IconRef backgroundIconRef,
   IconRef foregroundIconRef,
   IconRef *compositeIconRef
);
```

Parameters

backgroundIconRef

A value to use as the background for the composite `IconRef`.

foregroundIconRef

A value to use as the foregound for the composite `IconRef`.

compositeIconRef

On completion, this points to an `IconRef` that is a composite of the specified background and foreground `IconRefs`.

Return Value
A result code. See "Icon Services and Utilities Result Codes" (page 57).

Discussion
This function provides an alternative to badging when you need to indicate a change of state.

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

Declared In
`IconsCore.h`

DisposeIconActionUPP

Disposes of the universal procedure pointer (UPP) to your icon action callback function.

```
void DisposeIconActionUPP (
   IconActionUPP userUPP
);
```

Parameters

userUPP

The UPP to dispose of.

Discussion
See the `IconActionProcPtr` (page 36) callback for more information.

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

Declared In
`Icons.h`

DisposeIconGetterUPP

Disposes of the universal procedure pointer (UPP) to your icon getter callback function.

```
void DisposeIconGetterUPP (
   IconGetterUPP userUPP
);
```

Parameters

userUPP

The UPP to dispose of.

Discussion

See the `IconGetterProcPtr` (page 37) callback for more information.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`Icons.h`

GetCustomIconsEnabled

Determines whether custom icons are enabled or disabled on a specified volume.

```
OSErr GetCustomIconsEnabled (
   SInt16 vRefNum,
   Boolean *customIconsEnabled
);
```

Parameters

vRefNum

The volume whose status you are querying.

customIconsEnabled

On return, `customIconsEnabled` points to the value `true` if custom icons are enabled on the volume specified or `false` if custom icons are disabled on the volume specified.

Return Value

A result code. See "Icon Services and Utilities Result Codes" (page 57).

Availability

Available in Mac OS X v10.0 and later.

Declared In

`IconsCore.h`

GetIconFamilyData

Obtains a copy of the raw icon data for an individual element in an icon family.

```
OSErr GetIconFamilyData (
   IconFamilyHandle iconFamily,
   OSType iconType,
   Handle h
);
```

Parameters

iconFamily

A handle to an `iconFamily` data structure to use as a source for icon data.

iconType

The format of the icon data you want to obtain.You may specify one of the icon types (as defined in `IconStorage.h` in the CoreServices/OSServices framework) or `'PICT'` in this parameter. For example, you can pass `kThumbnail32BitData` (`'it32'`), to obtain 65,536 bytes of raw bitmap data.

h

A handle to the icon data being returned. Icon Services resizes this handle as needed. If no data is available for the specified icon family, Icon Services sets the handle to 0.

Return Value

A result code. See "Icon Services and Utilities Result Codes" (page 57).

Availability

Available in Mac OS X v10.0 and later.

Declared In

`Icons.h`

GetIconRef

Provides an `IconRef` object for an icon in the desktop database or for a registered icon.

```
OSErr GetIconRef (
   SInt16 vRefNum,
   OSType creator,
   OSType iconType,
   IconRef *theIconRef
);
```

Parameters

vRefNum

The volume where Icon Services should start to search for the desired icon. Pass the `kOnSystemDisk` constant if you are not sure which value to specify in this parameter.

creator

The creator code of the desired icon.

iconType

The type code of the desired icon.

theIconRef

On return, a pointer to an `IconRef` object. You are responsible for releasing the object by calling `ReleaseIconRef` (page 33).

Return Value

A result code. See "Icon Services and Utilities Result Codes" (page 57).

Discussion
Icon Services defines constants for commonly-used system icons. You can pass one of these constants in the `iconType` parameter if you specify `kSystemIconsCreator` in the `creator` parameter. See "Folder Icon Constants" (page 48) for a list of these constants.

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

Declared In
`IconsCore.h`

GetIconRefFromComponent

Provides an `IconRef` object based on a specified component.

```
OSStatus GetIconRefFromComponent (
   Component inComponent,
   IconRef *outIconRef
);
```

Parameters

`inComponent`
The component whose icon data you want to obtain.

`outIconRef`
On return, a pointer to an `IconRef` object based on the `componentIconFamily` field of the specified component's `'thng'` resource. You are responsible for releasing the object by calling `ReleaseIconRef` (page 33).

Return Value
A result code. See "Icon Services and Utilities Result Codes" (page 57).

Discussion
This function obtains an `IconRef` object based on the resource ID of an icon family. A component can provide an icon family in addition to the icon provided in the `componentIcon` field. Note that members of this icon family are not used by the Finder; you supply an icon family only so that other components or applications can display your component's icon in their user interfaces if needed.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`IconsCore.h`

GetIconRefFromFileInfo

Provides an `IconRef` object for a file with minimal file I/O.

```
OSStatus GetIconRefFromFileInfo (
    const FSRef *inRef,
    UniCharCount inFileNameLength,
    const UniChar *inFileName,
    FSCatalogInfoBitmap inWhichInfo,
    const FSCatalogInfo *inCatalogInfo,
    IconServicesUsageFlags inUsageFlags,
    IconRef *outIconRef,
    SInt16 *outLabel
);
```

Parameters

inRef

A pointer to an `FSRef` for the target file.

inFileNameLength

The length of the name of the target file.

inFileName

A pointer to the name of the target file.

inWhichInfo

The mask of the file information contained in the *inCatalogInfo* parameter.

inCatalogInfo

A pointer to the catalog information.

inUsageFlags

The usage flags for this call; use `kIconServicesNormalUsageFlag`.

outIconRef

On return, a pointer to an `IconRef` object. You are responsible for releasing the object by calling `ReleaseIconRef` (page 33).

outLabel

On return, a pointer to the output label for the icon/file.

Return Value

A result code. See "Icon Services and Utilities Result Codes" (page 57).

Discussion

To minimize file operations, `FSGetCatalogInfo` should be called prior to calling this function. The information in the `FSCatalogInfo` structure should correspond to that specified by `kIconServicesCatalogInfoMask`. The name should be fetched and passed in. If either the name or the correct catalog information is not passed in, this function will do file operations for this information instead.

Availability

Available in Mac OS X v10.1 and later.

Declared In

`IconsCore.h`

GetIconRefFromFolder

Provides an `IconRef` object for a folder with no custom icon.

```
OSErr GetIconRefFromFolder (
   SInt16 vRefNum,
   SInt32 parentFolderID,
   SInt32 folderID,
   SInt8 attributes,
   SInt8 accessPrivileges,
   IconRef *theIconRef
);
```

Parameters

vRefNum

The volume where the folder is located.

parentFolderID

The ID of the desired folder's parent folder.

folderID

The ID of the desired folder.

attributes

The attributes of the desired folder. You can obtain this data from the `CInfoPBRec.dirInfo.ioFlAttrib` field of the folder's catalog information record.

accessPrivileges

The access privileges of the specified folder. You can obtain this data from the `CInfoPBRec.dirInfo.ioACUser` field of the folder's catalog information record.

theIconRef

On return, a pointer to an `IconRef` object. You are responsible for releasing the object by calling `ReleaseIconRef` (page 33).

Return Value

A result code. See "Icon Services and Utilities Result Codes" (page 57).

Discussion

If you do not have the catalog information for a folder, use the function `GetIconRefFromFileInfo` (page 18).

Availability

Available in Mac OS X v10.0 and later.

Declared In

`IconsCore.h`

GetIconRefFromIconFamilyPtr

Provides an `IconRef` object from a specified icon family.

```
OSStatus GetIconRefFromIconFamilyPtr (
    const IconFamilyResource *inIconFamilyPtr,
    Size inSize,
    IconRef *outIconRef
);
```

Parameters

inIconFamilyPtr

A pointer to an icon family. See `IconStorage.h` for more information.

inSize
The size of the resource buffer containing the icon family.

outIconRef
On return, a pointer to an `IconRef` object that matches the specified inputs. You are responsible for releasing the object by calling `ReleaseIconRef` (page 33).

Return Value
A result code. See "Icon Services and Utilities Result Codes" (page 57).

Discussion
Typically, you do not need to use this function.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`IconsCore.h`

GetIconRefFromTypeInfo

Provides an `IconRef` object with the specified type information.

```
OSErr GetIconRefFromTypeInfo (
    OSType inCreator,
    OSType inType,
    CFStringRef inExtension,
    CFStringRef inMIMEType,
    IconServicesUsageFlags inUsageFlags,
    IconRef *outIconRef
);
```

Parameters
inCreator
The creator code of the desired `IconRef`. You may pass 0 if the creator code is unknown.

inType
The type code of the desired `IconRef`. You may pass 0 if the type code is unknown.

inExtension
The file name extension of the desired `IconRef`. You may pass `NULL` if the extension is unknown.

inMIMEType
The MIME type of the desired `IconRef`. You may pass `NULL` if the MIME type is unknown.

inUsageFlags
The usage flags; use `kIconServicesNormalUsageFlag`.

outIconRef
On return, a pointer to an `IconRef` object that most closely matches the specified inputs. You are responsible for releasing the object by calling `ReleaseIconRef` (page 33).

Return Value
A result code. See "Icon Services and Utilities Result Codes" (page 57).

Discussion
This function serves as a more versatile version of `GetIconRef` (page 17). If you specify creator and type codes and do not specify the extension and MIME type, calling this function is equivalent to calling `GetIconRef (kOnSystemDisk, inCreator, inType)`. If none of the input parameters is specified or if no match is found, this function returns the generic document icon.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`IconsCore.h`

GetIconRefOwners

Provides the current reference count for an `IconRef`.

```
OSErr GetIconRefOwners (
   IconRef theIconRef,
   UInt16 *owners
);
```

Parameters

`theIconRef`
An `IconRef` whose reference count you wish to obtain.

`owners`
On return, a pointer to the value which represents the current reference count.

Return Value
A result code. See "Icon Services and Utilities Result Codes" (page 57).

Discussion
When an `IconRef`'s reference count reaches 0, all memory allocated for the `IconRef` is marked as disposable. Any subsequent attempt to use the `IconRef` returns a result code of `-2580` (`invalidIconRefErr`).

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

Declared In
`IconsCore.h`

GetIconRefVariant

Specifies a transformation for a given `IconRef`.

```
IconRef GetIconRefVariant (
   IconRef inIconRef,
   OSType inVariant,
   IconTransformType *outTransform
);
```

Parameters

`inIconRef`
A value to be tested.

inVariant

A four-character value. You specify a variant by passing one of the following constants:

`kTileIconVariant` specifies a tiled icon.

`kRolloverIconVariant` specifies a rollover icon.

`kDropIconVariant` specifies a drop target icon.

`kOpenIconVariant` specifies an open icon.

`kOpenDropIconVariant` specifies a open drop target icon.

outTransform

On completion, this points to a transformation type that you pass to the function `PlotIconRef` (page 86) for purposes of hit-testing.

Return Value

An `IconRef` value that you pass to the function `PlotIconRef` (page 86) for purposes of hit-testing.

Discussion

Icon variants give you a simple way to indicate a temporary change of state by changing an icon's appearance. For example, if you specify the `kDropIconVariant` value when the user drags over a valid drop target, the `GetIconVariant` function provides the appropriate data for you to plot the variant with the function `PlotIconRef` (page 86).

Availability

Available in Mac OS X v10.0 and later.

Declared In

`Icons.h`

IconRefContainsCGPoint

Returns a Boolean value indicating whether an icon contains a specified point.

```
Boolean IconRefContainsCGPoint (
   const CGPoint *testPt,
   const CGRect *iconRect,
   IconAlignmentType align,
   IconServicesUsageFlags iconServicesUsageFlags,
   IconRef theIconRef
);
```

Parameters

testPt

A pointer to the point to be tested. The point should be specified in the coordinate system of the rectangle specified in the `iconRect` parameter.

iconRect

A pointer to the rectangle in which the icon appears. The rectangle you specify should be the same rectangle that you last used to draw the icon.

align

Specifies how the icon is aligned within the rectangle specified in the `iconRect` parameter. The alignment you specify should be the same alignment that you last used to draw the icon. See "`Icon Alignment Constants`" (page 41) for a description of the values you can use in this parameter.

iconServicesUsageFlags

Reserved for future use. Pass the `kIconServicesNormalUsageFlag` constant in this parameter.

theIconRef

The icon to test.

Return Value
`TRUE` if the point is in the icon and `FALSE` if it is not.

Discussion
This function uses the size of the rectangle you specify to determine the optimal icon mask to represent the icon. The function uses the alignment information you specify to adjust the position of the mask inside its bounding rectangle, and then determines whether the specified point is within the mask.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`Icons.h`

IconRefIntersectsCGRect

Returns a Boolean value indicating whether an icon intersects a specified rectangle.

```
Boolean IconRefIntersectsCGRect (
   const CGRect *testRect,
   const CGRect *iconRect,
   IconAlignmentType align,
   IconServicesUsageFlags iconServicesUsageFlags,
   IconRef theIconRef
);
```

Parameters

testRect

A pointer to the rectangle to be tested. The rectangle should be specified in the coordinate system of the rectangle specified in the `iconRect` parameter.

iconRect

A pointer to the rectangle in which the icon appears. The rectangle you specify should be the same rectangle that you last used to draw the icon.

align

Specifies how the icon is aligned within the rectangle specified by the `iconRect` parameter. The alignment you specify should be the same alignment that you last used to draw the icon. See "Icon Alignment Constants" (page 41) for a description of the values you can use in this parameter.

iconServicesUsageFlags

Reserved for future use. Pass the `kIconServicesNormalUsageFlag` constant in this parameter.

theIconRef

The icon to test.

Return Value
`TRUE` if the point is in the icon and `FALSE` if it is not.

Discussion
This function uses the size of the rectangle you specify in the `iconRect` parameter to determine the optimal icon mask to represent the icon. The function uses the alignment information you specify to adjust the position of the mask inside its bounding rectangle, and then determines whether the rectangle you specify in the `testRect` parameter intersects the mask.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`Icons.h`

IconRefToHIShape

Converts an icon into an HIShape object.

```
HIShapeRef IconRefToHIShape (
   const CGRect *iconRect,
   IconAlignmentType align,
   IconServicesUsageFlags iconServicesUsageFlags,
   IconRef theIconRef
);
```

Parameters

iconRect

A pointer to the rectangle defining the area that Icon Services uses as the bounding box of the shape.

align

A value which determines how Icon Services aligns the shape within the rectangle. For a description of possible values, see "Icon Alignment Constants" (page 41).

iconServicesUsageFlags

Reserved for future use. Pass the `kIconServicesNormalUsageFlag` constant in this parameter.

theIconRef

The icon to be converted.

Return Value
An HIShape object, or `NULL` if the icon could not be converted.

Discussion
This function uses the size of the rectangle you specify to determine the optimal icon mask to represent the icon. The function uses the alignment information you specify to adjust the position of the mask inside its bounding rectangle, and then returns the shape of the mask.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`Icons.h`

IconRefToIconFamily

Provides icon family data for a given `IconRef`.

```
OSErr IconRefToIconFamily (
   IconRef theIconRef,
   IconSelectorValue whichIcons,
   IconFamilyHandle *iconFamily
);
```

Parameters

theIconRef

An `IconRef` to use as a source for icon data.

whichIcons

The depths and sizes of icons in the `iconFamily` data structure. For a description of the possible values, see "Icon Selector Constants" (page 43).

iconFamily

On return, a pointer to a handle to the data structure which contains icon data as specified in the `IconRef` and `whichIcons` parameters. Icon Services returns `NULL` if no appropriate icon data is found. For more information on the `IconFamily` data structure, see 'icns'.

Return Value
A result code. See "Icon Services and Utilities Result Codes" (page 57).

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

Declared In
`Icons.h`

InvokeIconActionUPP

Calls your icon action callback function.

```
OSErr InvokeIconActionUPP (
   ResType theType,
   Handle *theIcon,
   void *yourDataPtr,
   IconActionUPP userUPP
);
```

Return Value
A result code. See "Icon Services and Utilities Result Codes" (page 57).

Discussion
You should not need to use the function `InvokeIconActionUPP`, as the system calls your icon action callback function for you.

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

Declared In
`Icons.h`

InvokeIconGetterUPP

Calls your icon getter callback function.

```
Handle InvokeIconGetterUPP (
   ResType theType,
   void *yourDataPtr,
   IconGetterUPP userUPP
);
```

Discussion
You should not need to use the function `InvokeIconGetterUPP`, as the system calls your icon getter callback function for you.

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

Declared In
`Icons.h`

IsDataAvailableInIconRef

Indicates whether an `IconRef` has the specified data.

```
Boolean IsDataAvailableInIconRef (
   OSType inIconKind,
   IconRef inIconRef
);
```

Parameters

`inIconKind`
The icon data kind. See `IconStorage.h` for more information.

`inIconRef`
The icon reference whose data you want to check.

Return Value
`True` if the icon reference contains the indicated data, `False` otherwise.

Discussion
This function can be used to determine the optimal icon size if you plan to cache a bitmap image of the icon.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`IconsCore.h`

IsIconRefComposite

Reports whether a specified `IconRef` has been composited.

```
OSErr IsIconRefComposite (
   IconRef compositeIconRef,
   IconRef *backgroundIconRef,
   IconRef *foregroundIconRef
);
```

Parameters

compositeIconRef

An `IconRef` that you wish to test to determine whether it has been composited.

backgroundIconRef

On return, this points to the `IconRef` value that forms the background of the `IconRef` specified in the `compositeIconRef` parameter. If the `IconRef` specified in the `compositeIconRef` parameter is not a composite, the return value is 0.

foregroundIconRef

On return, this points to the `IconRef` value that forms the foreground of the `IconRef` specified in the `compositeIconRef` parameter. If the `IconRef` specified in the `compositeIconRef` parameter is not a composite, the return value is 0.

Return Value

A result code. See "Icon Services and Utilities Result Codes" (page 57).

Discussion

The function `CompositeIconRef` (page 14) allows the creation of a composite `IconRef` from a given background `IconRef` and a given foreground `IconRef`. The `IsIconRefComposite` function checks a specified `IconRef` to determine whether it is a composite and, if so, provides the background and foreground `IconRef` values.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`IconsCore.h`

IsIconRefMaskEmpty

Reports whether a specified mask is empty.

```
Boolean IsIconRefMaskEmpty (
   IconRef iconRef
);
```

Parameters

iconRef

An `IconRef` whose mask you wish to test.

Return Value

`true` if the mask associated with the given `IconRef` is empty, `false` otherwise.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`Icons.h`

IsValidIconRef

Reports whether a specified `IconRef` is valid.

```
Boolean IsValidIconRef (
   IconRef theIconRef
);
```

Parameters

`theIconRef`
An `IconRef`.

Return Value

`true` if the specified `IconRef` is valid, `false` otherwise.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`IconsCore.h`

NewIconActionUPP

Creates a new universal procedure pointer (UPP) to an icon action callback function.

```
IconActionUPP NewIconActionUPP (
   IconActionProcPtr userRoutine
);
```

Parameters

`userRoutine`
A pointer to your icon action function.

Return Value

A UPP to the icon action function.

Discussion

See the `IconActionProcPtr` (page 36) callback for more information.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`Icons.h`

NewIconGetterUPP

Creates a new universal procedure pointer (UPP) to an icon getter callback function.

```
IconGetterUPP NewIconGetterUPP (
   IconGetterProcPtr userRoutine
);
```

Parameters

userRoutine

A pointer to your icon getter function.

Return Value
A UPP to the icon getter function.

Discussion
See the `IconGetterProcPtr` (page 37) callback for more information.

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

Declared In
`Icons.h`

OverrideIconRef

Replaces the bitmaps of one `IconRef` with those of another `IconRef`.

```
OSErr OverrideIconRef (
   IconRef oldIconRef,
   IconRef newIconRef
);
```

Parameters

oldIconRef

A pointer to a value of type `IconRef` whose bitmaps are to be replaced.

newIconRef

A pointer to a value of type`IconRef` containing the replacement bitmaps.

Return Value
A result code. See "Icon Services and Utilities Result Codes" (page 57).

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

Declared In
`IconsCore.h`

PlotIconRefInContext

Plots an `IconRef` using Quartz.

```
OSStatus PlotIconRefInContext (
   CGContextRef inContext,
   const CGRect *inRect,
   IconAlignmentType inAlign,
   IconTransformType inTransform,
   const RGBColor *inLabelColor,
   PlotIconRefFlags inFlags,
   IconRef inIconRef
);
```

Parameters

inContext

The graphics context to use.

inRect

A pointer to the rectangle to plot the icon in.

inAlign

The icon alignment. See "Icon Alignment Constants" (page 41).

inTransform

The icon transform. See "Icon Transformation Constants" (page 43).

inLabelColor

A pointer to the icon label color.

inFlags

The drawing flags to use; this is usually `kPlotIconRefNormalFlags`.

inIconRef

The `IconRef` to plot.

Return Value

A result code. See "Icon Services and Utilities Result Codes" (page 57).

Availability

Available in Mac OS X v10.1 and later.

Declared In

`Icons.h`

ReadIconFromFSRef

Reads an icon ('`icns`') file into memory.

```
OSStatus ReadIconFromFSRef (
   const FSRef *ref,
   IconFamilyHandle *iconFamily
);
```

Parameters

ref

A pointer to the `FSRef` for the icon file.

iconFamily

A pointer to the handle for the icon family.

Return Value

A result code. See "Icon Services and Utilities Result Codes" (page 57).

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

Declared In
`IconsCore.h`

RegisterIconRefFromFSRef

Registers an `IconRef` from a `.icns` file and associates it with a creator and type pair.

```
OSStatus RegisterIconRefFromFSRef (
   OSType creator,
   OSType iconType,
   const FSRef *iconFile,
   IconRef *theIconRef
);
```

Parameters

creator
The creator code for the `.icns` file.

iconType
The type code for the `.icns` file.

iconFile
A pointer to the `FSRef` of the `.icns` file.

theIconRef
A pointer to an `IconRef`. On return, this contains the registered `IconRef`.

Return Value
A result code. See "Icon Services and Utilities Result Codes" (page 57).

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

Declared In
`IconsCore.h`

RegisterIconRefFromIconFamily

Adds an `iconFamily`-derived `IconRef` to the Icon Services registry.

```
OSErr RegisterIconRefFromIconFamily (
   OSType creator,
   OSType iconType,
   IconFamilyHandle iconFamily,
   IconRef *theIconRef
);
```

Parameters

creator
The creator code of the desired icon. You can use your application's creator code, for example. Lower-case creator codes are reserved for the System.

iconType
The type code of the desired icon.

iconFamily
A handle to the `iconFamily` data structure to register.

theIconRef
On return, a pointer to the desired icon data.

Return Value
A result code. See "Icon Services and Utilities Result Codes" (page 57).

Discussion
Consider using the function `RegisterIconRefFromIconFile` (page 97), since the data registered using the `RegisterIconRefFromIconFamily` function cannot be purged. You are responsible for disposing of the `IconRef` by using the function `ReleaseIconRef` (page 33).

Calling this function increments the reference count of the `IconRef`.

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

Declared In
`IconsCore.h`

ReleaseIconRef

Decrements the reference count for an `IconRef`.

```
OSErr ReleaseIconRef (
   IconRef theIconRef
);
```

Parameters

theIconRef
An `IconRef` whose reference count you wish to decrement.

Return Value
A result code. See "Icon Services and Utilities Result Codes" (page 57).

Discussion
When an `IconRef`'s reference count reaches 0, all memory allocated for the `IconRef` is marked as disposable. Any subsequent attempt to use the `IconRef` returns a result code of - `2580` (`invalidIconRefErr`).

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

Declared In
`IconsCore.h`

RemoveIconRefOverride

Restores the original bitmaps of an overridden `IconRef`.

```
OSErr RemoveIconRefOverride (
   IconRef theIconRef
);
```

Parameters

theIconRef

A pointer to a value of type `IconRef` whose bitmaps are to be restored.

Return Value

A result code. See "Icon Services and Utilities Result Codes" (page 57).

Availability

Available in Mac OS X v10.0 and later.

Declared In

`IconsCore.h`

SetCustomIconsEnabled

Enables or disables custom icons on a specified volume.

```
OSErr SetCustomIconsEnabled (
   SInt16 vRefNum,
   Boolean enableCustomIcons
);
```

Parameters

vRefNum

The volume where custom icons are to be enabled or disabled.

enableCustomIcons

If you pass `true`, custom icons are enabled on the volume specified. Passing `false` disables custom icons on the volume specified.

Return Value

A result code. See "Icon Services and Utilities Result Codes" (page 57).

Discussion

If you use the `SetCustomIconsEnabled` function to enable or disable custom icons, the setting remains in effect only as long as the specified volume remains mounted during the current session.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`IconsCore.h`

SetIconFamilyData

Provides new raw icon data for an individual element of an icon family.

```
OSErr SetIconFamilyData (
   IconFamilyHandle iconFamily,
   OSType iconType,
   Handle h
);
```

Parameters

iconFamily

A handle to an `iconFamily` data structure to be used as the target.

iconType

The format of the icon data you provide.You may specify one of the icon types (as defined in `IconStorage.h` in the CoreServices/OSServices framework) or `'PICT'` in this parameter. For a thumbnail icon, for example, you specify `kThumbnail32BitData` in this parameter. For a thumbnail mask, you specify `kThumbnail8BitMask`.

h

A handle to the icon data you provide. For a thumbnail icon, the handle contains raw image data in the form of 128x128, four bytes per pixel, RGB data. For a thumbnail mask, the data is in the same format except that it is one byte per pixel.

Return Value
A result code. See "Icon Services and Utilities Result Codes" (page 57).

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

Declared In
`Icons.h`

UnregisterIconRef

Removes the specified icon data from the icon registry.

```
OSErr UnregisterIconRef (
   OSType creator,
   OSType iconType
);
```

Parameters

creator

The creator code of the icon data to be unregistered.

iconType

The type code of the icon data to be unregistered.

Return Value
A result code. See "Icon Services and Utilities Result Codes" (page 57).

Discussion
The specified icon data is not unregistered until all its users have called the function `ReleaseIconRef` (page 33).

You should not unregister an icon that you have not registered.

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

Declared In
`IconsCore.h`

UpdateIconRef

Forces an update of `IconRef` data.

```
OSErr UpdateIconRef (
   IconRef theIconRef
);
```

Parameters

`theIconRef`

An `IconRef` to be updated.

Return Value
A result code. See "Icon Services and Utilities Result Codes" (page 57).

Discussion
This function is useful after you have changed a file or folder's custom icon, for example. Do not call the `UpdateIconRef` function if you have not already obtained an `IconRef` for a particular icon; call the function `GetIconRefFromFile` (page 68) instead.

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

Declared In
`IconsCore.h`

Callbacks

IconActionProcPtr

Defines a pointer to an icon action callback function, which performs an action on a single icon.

```
typedef OSErr (*IconActionProcPtr) (
    ResType theType,
    Handle *theIcon,
    void *yourDataPtr
);
```

If you name your function `MyIconActionProc`, you would declare it like this:

```
OSErr MyIconActionProc (
    ResType theType,
    Handle *theIcon,
    void *yourDataPtr
);
```

Parameters

theType

The resource type of the icon.

theIcon

A pointer to the handle to the icon on which to perform the operation.

yourDataPtr

A pointer to data as specified in the `yourDataPtr` parameter of the `ForEachIconDo` function. When your application calls `ForEachIconDo`, it typically provides in the `yourDataPtr` parameter a value that identifies the action your function should perform.

Return Value

A result code. See "Icon Services and Utilities Result Codes" (page 57).

Discussion

You can perform operations on every icon in an icon suite by providing a pointer to an icon action function as a parameter to the `ForEachIconDo` (page 63) function. The `ForEachIconDo` function calls your icon action function for specified icon resource types. Your icon action function should return a result code indicating whether it successfully performed the action on the icon.

Before using your icon action function, you must first create a new universal procedure pointer to it, using the `NewIconActionUPP` (page 29) function, as shown here:

```
IconActionUPP MyIconActionUPP;
MyIconActionUPP = NewIconActionUPP(&MyIconActionProc)
```

You then pass `MyIconActionUPP` to the `ForEachIconDo` function. When you are finished with your icon action callback function, you should dispose of the universal procedure pointer associated with it:

```
DisposeIconActionUPP(MyIconActionUPP);
```

Availability

Available in Mac OS X v10.0 and later.

Declared In

`Icons.h`

IconGetterProcPtr

Defines a pointer to an icon getter callback function, which retrieves a handle to an icon of the requested type.

```
typedef Handle (*IconGetterProcPtr) (
    ResType theType,
    void *yourDataPtr
);
```

If you name your function `MyIconGetterProc`, you would declare it like this:

```
Handle MyIconGetterProc (
    ResType theType,
    void *yourDataPtr
);
```

Parameters

theType

The resource type of the icon. In general, you should specify your icon resources as purgeable.

yourDataPtr

If your icon getter function was called by an icon cache function, this parameter contains, on return, a pointer to the data associated with the icon cache. Otherwise, this parameter contains the value your application specified in the `yourDataPtr` parameter. For icon caches, you initially set this value when you first create a cache using the `MakeIconCache` (page 78) function. You can change this value using the `SetIconCacheData` (page 99) function. The icon getter function can use this data as needed.

Return Value

An icon getter function should return as its function result a handle to the requested icon's data.

Discussion

If you use icon caches, you must provide at least one icon getter function. The `MakeIconCache` function takes a pointer to an icon getter function for use with a new icon cache. Subsequent calls to Icon Utilities functions that use icon types not present in the icon cache use the icon getter function associated with the icon cache to return a handle to the icon data. To get and set an existing icon cache's icon getter function, use the `GetIconCacheProc` (page 66) and `SetIconCacheProc` (page 100) functions.

You can also specify an icon getter function for use by the `PlotIconMethod` (page 85), `IconMethodToRgn` (page 73), `PtInIconMethod` (page 89), and `RectInIconMethod` (page 94) functions. Like Icon Utilities functions that work with icon caches, the icon getter function that you provide as a parameter to `PlotIconMethod` should return a handle to the requested icon's data. Note that the icon getter function that you provide as a parameter to `IconMethodToRgn`, `PtInIconMethod`, and `RectInIconMethod` should also return a handle to the requested icon; these three functions then extract the icon mask from the icon data your icon getter function returns.

Before using your icon getter function, you must first create a new universal procedure pointer to it, using the `NewIconGetterUPP` (page 29) function, as shown here:

```
IconGetterUPP MyIconGetterUPP;
MyIconGetterUPP = NewIconGetterUPP(&MyIconGetterProc)
```

You can then pass `MyIconGetterUPP` to any of the Icon Utilities functions which use custom icon getter functions. When you are finished with your icon getter callback function, you should dispose of the universal procedure pointer associated with it, using the `DisposeIconGetterUPP` (page 15) function:

```
DisposeIconGetterUPP(MyIconGetterUPP);
```

Availability

Available in Mac OS X v10.0 and later.

Declared In

`Icons.h`

Data Types

CIcon

Defines a color icon structure.

```
struct CIcon {
    PixMap iconPMap;
    BitMap iconMask;
    BitMap iconBMap;
    Handle iconData;
    SInt16 iconMaskData[1];
};
typedef struct CIcon CIcon;
typedef CIcon * CIconPtr;
```

Fields

`iconPMap`
: The pixel map describing the icon. Note that this is a pixel map record, not a handle to a pixel map record.

`iconMask`
: A bitmap of the icon's mask.

`iconBMap`
: A bitmap of the icon.

`iconData`
: A handle to the icon's pixel image.

`iconMaskData`
: An array containing the icon's mask data followed by the icon's bitmap data. This is used only when the icon is stored as a resource.

Discussion

The `PlotCIcon` (page 80), `PlotCIconHandle` (page 81), `GetCIcon` (page 64), and `DisposeCIcon` (page 62) functions all use the `CIconHandle` data type to refer to a color icon structure. A color icon structure contains information about a color icon.

All color icon resources should be marked purgeable. You can use icons of resource type `'cicn'` in menus the same way that you use resources of type `'ICON'`. If a menu item specifies an icon number, the menu definition function first tries to load in a `'cicn'` resource with the specified resource ID. If it doesn't find one, the menu definition function tries to load in an `'ICON'` resource with the same ID. The Dialog Manager also uses a `'cicn'` resource instead of an `'ICON'` resource if it finds one with the same resource ID.

Availability

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

Declared In

`Icons.h`

IconRef

Defines an icon reference.

```
typedef struct OpaqueIconRef * IconRef;
```

Discussion

An `IconRef` is a 32–bit values identifying cached icon data. `IconRef` 0 is invalid.

Availability

Available in Mac OS X v10.0 and later.

Declared In
IconsCore.h

IconActionUPP

Defines a universal procedure pointer (UPP) to an icon action callback function.

```
typedef IconActionProcPtr IconActionUPP;
```

Discussion
For more information, see the description of the IconActionProcPtr (page 36) callback function.

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

Declared In
Icons.h

IconGetterUPP

Defines a universal procedure pointer to an icon getter callback function.

```
typedef IconGetterProcPtr IconGetterUPP;
```

Discussion
For more information, see the description of the IconGetterProcPtr (page 37) callback function.

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

Declared In
Icons.h

IconCacheRef

Defines a reference to an icon cache.

```
typedef Handle IconCacheRef;
```

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

Declared In
Icons.h

IconSuiteRef

Defines a reference to an icon suite.

```
typedef Handle IconSuiteRef;
```

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

Declared In
`Icons.h`

Constants

Icon Alignment Constants

Define constants that allow you to specify how to align an icon within its rectangle.

```
enum {
    kAlignNone = 0x00,
    kAlignVerticalCenter = 0x01,
    kAlignTop = 0x02,
    kAlignBottom = 0x03,
    kAlignHorizontalCenter = 0x04,
    kAlignAbsoluteCenter = kAlignVerticalCenter | kAlignHorizontalCenter,
    kAlignCenterTop = kAlignTop | kAlignHorizontalCenter,
    kAlignCenterBottom = kAlignBottom | kAlignHorizontalCenter,
    kAlignLeft = 0x08,
    kAlignCenterLeft = kAlignVerticalCenter | kAlignLeft,
    kAlignTopLeft = kAlignTop | kAlignLeft,
    kAlignBottomLeft = kAlignBottom | kAlignLeft,
    kAlignRight = 0x0C,
    kAlignCenterRight = kAlignVerticalCenter | kAlignRight,
    kAlignTopRight = kAlignTop | kAlignRight,
    kAlignBottomRight = kAlignBottom | kAlignRight
};
typedef SInt16 IconAlignmentType;
```

Constants

`kAlignNone`
Use this value if you do not wish to specify a particular alignment.
Available in Mac OS X v10.0 and later.
Declared in `Icons.h`.

`kAlignVerticalCenter`
Use this value to center the icon vertically within the rectangle.
Available in Mac OS X v10.0 and later.
Declared in `Icons.h`.

`kAlignTop`
Use this value to top align the icon within the rectangle.
Available in Mac OS X v10.0 and later.
Declared in `Icons.h`.

`kAlignBottom`
Use this value to bottom align the icon within the rectangle.
Available in Mac OS X v10.0 and later.
Declared in `Icons.h`.

`kAlignHorizontalCenter`
Use this value to center the icon horizontally within the rectangle.
Available in Mac OS X v10.0 and later.
Declared in `Icons.h`.

`kAlignAbsoluteCenter`
Use this value to center the icon horizontally and vertically within the rectangle.
Available in Mac OS X v10.0 and later.
Declared in `Icons.h`.

`kAlignCenterTop`
Use this value to top align the icon and center it horizontally within the rectangle.
Available in Mac OS X v10.0 and later.
Declared in `Icons.h`.

`kAlignCenterBottom`
Use this value to bottom align the icon and center it horizontally within the rectangle.
Available in Mac OS X v10.0 and later.
Declared in `Icons.h`.

`kAlignLeft`
Use this value to left align the icon within the rectangle.
Available in Mac OS X v10.0 and later.
Declared in `Icons.h`.

`kAlignCenterLeft`
Use this value to left align the icon and center it vertically within the rectangle.
Available in Mac OS X v10.0 and later.
Declared in `Icons.h`.

`kAlignTopLeft`
Use this value to left and top align the icon within the rectangle.
Available in Mac OS X v10.0 and later.
Declared in `Icons.h`.

`kAlignBottomLeft`
Use this value to left and bottom align the icon within the rectangle.
Available in Mac OS X v10.0 and later.
Declared in `Icons.h`.

`kAlignRight`
Use this value to right align the icon within the rectangle.
Available in Mac OS X v10.0 and later.
Declared in `Icons.h`.

`kAlignCenterRight`
Use this value to right align the icon and center it vertically within the rectangle.

Available in Mac OS X v10.0 and later.

Declared in `Icons.h`.

`kAlignTopRight`
Use this value to right and top align the icon within the rectangle.

Available in Mac OS X v10.0 and later.

Declared in `Icons.h`.

`kAlignBottomRight`
Use this value to right and bottom align the icon within the rectangle.

Available in Mac OS X v10.0 and later.

Declared in `Icons.h`.

Discussion
Icon Services and Utilities functions use the `IconAlignmentType` constants to determine how an icon is aligned within its bounding rectangle.

Icon Transformation Constants

Define values that Icon Services uses to report how an icon has been transformed after you call the function `GetIconRefVariant`.

```
enum {
    kTransformNone = 0x00,
    kTransformDisabled = 0x01,
    kTransformOffline = 0x02,
    kTransformOpen = 0x03,
    kTransformLabel1 = 0x0100,
    kTransformLabel2 = 0x0200,
    kTransformLabel3 = 0x0300,
    kTransformLabel4 = 0x0400,
    kTransformLabel5 = 0x0500,
    kTransformLabel6 = 0x0600,
    kTransformLabel7 = 0x0700,
    kTransformSelected = 0x4000,
    kTransformSelectedDisabled = kTransformSelected | kTransformDisabled,
    kTransformSelectedOffline = kTransformSelected | kTransformOffline,
    kTransformSelectedOpen = kTransformSelected | kTransformOpen
};
typedef SInt16 IconTransformType;
```

Discussion
The functions `PlotIconID` (page 83)`PlotIconMethod` (page 85), `PlotIconHandle` (page 82), `PlotCIconHandle` (page 81), `PlotIconSuite` (page 86), `LoadIconCache` (page 77) and `PlotSICNHandle` (page 88) use these constants to specify how an icon should be modified, if at all, when plotted.

Icon Selector Constants

Describe values that you can use to obtain information about the sizes and depths of icons available in a given icon family.

```
enum {
    kSelectorLarge1Bit = 0x00000001,
    kSelectorLarge4Bit = 0x00000002,
    kSelectorLarge8Bit = 0x00000004,
    kSelectorLarge32Bit = 0x00000008,
    kSelectorLarge8BitMask = 0x00000010,
    kSelectorSmall1Bit = 0x00000100,
    kSelectorSmall4Bit = 0x00000200,
    kSelectorSmall8Bit = 0x00000400,
    kSelectorSmall32Bit = 0x00000800,
    kSelectorSmall8BitMask = 0x00001000,
    kSelectorMini1Bit = 0x00010000,
    kSelectorMini4Bit = 0x00020000,
    kSelectorMini8Bit = 0x00040000,
    kSelectorHuge1Bit = 0x01000000,
    kSelectorHuge4Bit = 0x02000000,
    kSelectorHuge8Bit = 0x04000000,
    kSelectorHuge32Bit = 0x08000000,
    kSelectorHuge8BitMask = 0x10000000,
    kSelectorAllLargeData = 0x000000FF,
    kSelectorAllSmallData = 0x0000FF00,
    kSelectorAllMiniData = 0x00FF0000,
    kSelectorAllHugeData = 0xFF000000,
    kSelectorAll1BitData = kSelectorLarge1Bit | kSelectorSmall1Bit
| kSelectorMini1Bit | kSelectorHuge1Bit,
    kSelectorAll4BitData = kSelectorLarge4Bit | kSelectorSmall4Bit
| kSelectorMini4Bit | kSelectorHuge4Bit,
    kSelectorAll8BitData = kSelectorLarge8Bit | kSelectorSmall8Bit
| kSelectorMini8Bit | kSelectorHuge8Bit,
    kSelectorAll32BitData = kSelectorLarge32Bit | kSelectorSmall32Bit
| kSelectorHuge32Bit,
    kSelectorAllAvailableData = 0xFFFFFFFF
};
typedef UInt32 IconSelectorValue;
```

Constants

`kSelectorLarge1Bit`

Specify to include `'ICN#'` resource.

Available in Mac OS X v10.0 and later.

Declared in `Icons.h`.

`kSelectorLarge4Bit`

Specify to include `'icl4'` resource.

Available in Mac OS X v10.0 and later.

Declared in `Icons.h`.

`kSelectorLarge8Bit`

Specify to include `'icl8'` resource.

Available in Mac OS X v10.0 and later.

Declared in `Icons.h`.

`kSelectorLarge32Bit`

Available in Mac OS X v10.0 and later.

Declared in `Icons.h`.

`kSelectorLarge8BitMask`
Available in Mac OS X v10.0 and later.
Declared in `Icons.h`.

`kSelectorSmall1Bit`
Specify to include `'ics#'` resource.
Available in Mac OS X v10.0 and later.
Declared in `Icons.h`.

`kSelectorSmall4Bit`
Specify to include `'ics4'` resource.
Available in Mac OS X v10.0 and later.
Declared in `Icons.h`.

`kSelectorSmall8Bit`
Specify to include `'ics8'` resource.
Available in Mac OS X v10.0 and later.
Declared in `Icons.h`.

`kSelectorSmall32Bit`
Available in Mac OS X v10.0 and later.
Declared in `Icons.h`.

`kSelectorSmall8BitMask`
Available in Mac OS X v10.0 and later.
Declared in `Icons.h`.

`kSelectorMini1Bit`
Specify to include `'icm#'` resource.
Available in Mac OS X v10.0 and later.
Declared in `Icons.h`.

`kSelectorMini4Bit`
Specify to include `'icm4'` resource.
Available in Mac OS X v10.0 and later.
Declared in `Icons.h`.

`kSelectorMini8Bit`
Specify to include `'icm8'` resource.
Available in Mac OS X v10.0 and later.
Declared in `Icons.h`.

`kSelectorHuge1Bit`
Available in Mac OS X v10.0 and later.
Declared in `Icons.h`.

`kSelectorHuge4Bit`
Available in Mac OS X v10.0 and later.
Declared in `Icons.h`.

`kSelectorHuge8Bit`
Available in Mac OS X v10.0 and later.
Declared in `Icons.h`.

`kSelectorHuge32Bit`
Available in Mac OS X v10.0 and later.
Declared in `Icons.h`.

`kSelectorHuge8BitMask`
Available in Mac OS X v10.0 and later.
Declared in `Icons.h`.

`kSelectorAllLargeData`
Specify to include `'ICN#'`, `'icl4'`, and `'icl8'` resources.
Available in Mac OS X v10.0 and later.
Declared in `Icons.h`.

`kSelectorAllSmallData`
Specify to include `'ics#'`, `'ics4'`, and `'ics8'` resources.
Available in Mac OS X v10.0 and later.
Declared in `Icons.h`.

`kSelectorAllMiniData`
Specify to include `'icm#'`, `'icm4'`, and `'icm8'` resources.
Available in Mac OS X v10.0 and later.
Declared in `Icons.h`.

`kSelectorAllHugeData`
Available in Mac OS X v10.0 and later.
Declared in `Icons.h`.

`kSelectorAll1BitData`
Specify to include `'ICN#'`, `'ics#'`, and `'icm#'` resources.
Available in Mac OS X v10.0 and later.
Declared in `Icons.h`.

`kSelectorAll4BitData`
Specify to include `'icl4'`, `'ics4'`, and `'icm4'` resources.
Available in Mac OS X v10.0 and later.
Declared in `Icons.h`.

`kSelectorAll8BitData`
Specify to include `'icl8'`, `'ics8'`, and `'icm8'` resources.
Available in Mac OS X v10.0 and later.
Declared in `Icons.h`.

`kSelectorAll32BitData`
Available in Mac OS X v10.0 and later.
Declared in `Icons.h`.

`kSelectorAllAvailableData`
Specify to include all resources of given ID.
Available in Mac OS X v10.0 and later.
Declared in `Icons.h`.

Discussion
The functions `GetIconSuite` (page 69) and `ForEachIconDo` (page 63) use these constants in the `selector` parameter to specify which members of the family to include in the icon suite.

Catalog Information Bitmask

Defines a minimal bitmask for use with the `GetIconRefFromFileInfo` function.

```
enum {
    kIconServicesCatalogInfoMask =
            (kFSCatInfoNodeID | kFSCatInfoParentDirID | kFSCatInfoVolume
            | kFSCatInfoNodeFlags | kFSCatInfoFinderInfo |
            kFSCatInfoFinderXInfo | kFSCatInfoUserAccess)
};
```

Constants

`kIconServicesCatalogInfoMask`

Use this mask with the File Manager function `FSGetCatalogInfo` before calling `GetIconRefFromFileInfo`.

Available in Mac OS X v10.1 and later.

Declared in `IconsCore.h`.

System Icon Constant

Defines a creator type for all system–defined icons.

```
enum {
    kSystemIconsCreator = 'macs'
};
```

Discussion

You can use the `kSystemIconsCreator` constant to obtain System icons that are not associated with a file, such as the help icon.

Icon Services Usage Flag

```
typedef UInt32 IconServicesUsageFlags;
enum {
    kIconServicesNormalUsageFlag = 0
};
```

Alert Icon Constants

Specify standard alert icons.

```
enum {
    kAlertNoteIcon = 'note',
    kAlertCautionIcon = 'caut',
    kAlertStopIcon = 'stop'
};
```

Discussion

Icon Services defines constants for a number of standard alert icons. You can pass one of these constants in the `iconType` parameter of the function `GetIconRef` (page 17), for example.

Filesharing Privilege Icon Constants

Identify standard filesharing privilege icons.

```
enum {
    kSharingPrivsNotApplicableIcon = 'shna',
    kSharingPrivsReadOnlyIcon = 'shro',
    kSharingPrivsReadWriteIcon = 'shrw',
    kSharingPrivsUnknownIcon = 'shuk',
    kSharingPrivsWritableIcon = 'writ'
};
```

Discussion
Icon Services defines constants for a number of standard filesharing privilege icons. You can pass one of these constants in the `iconType` parameter of the function `GetIconRef` (page 17), for example.

Folder Icon Constants

Identify standard folder icons.

```
enum {
    kGenericFolderIcon = 'fldr',
    kDropFolderIcon = 'dbox',
    kMountedFolderIcon = 'mntd',
    kOpenFolderIcon = 'ofld',
    kOwnedFolderIcon = 'ownd',
    kPrivateFolderIcon = 'prvf',
    kSharedFolderIcon = 'shfl'
};
```

Discussion
Icon Services defines constants for a number of standard folder icons. You can pass one of these constants in the `iconType` parameter of the function `GetIconRef` (page 17), for example.

Internet Icon Constants

Identify standard Internet icons.

```
enum {
    kInternetLocationHTTPIcon = 'ilht',
    kInternetLocationFTPIcon = 'ilft',
    kInternetLocationAppleShareIcon = 'ilaf',
    kInternetLocationAppleTalkZoneIcon = 'ilat',
    kInternetLocationFileIcon = 'ilfi',
    kInternetLocationMailIcon = 'ilma',
    kInternetLocationNewsIcon = 'ilnw',
    kInternetLocationNSLNeighborhoodIcon = 'ilns',
    kInternetLocationGenericIcon = 'ilge'
};
```

Discussion
Icon Services defines constants for a number of standard Internet icons. You can pass one of these constants in the `iconType` parameter of the function `GetIconRef` (page 17), for example.

Toolbar Icons

Identify standard toolbar icons.

```
enum {
    kToolbarCustomizeIcon = 'tcus',
    kToolbarDeleteIcon = 'tdel',
    kToolbarFavoritesIcon = 'tfav',
    kToolbarHomeIcon = 'thom'
};
```

Miscellaneous Icon Constants

Identify miscellaneous icons.

```
enum {
    kAppleLogoIcon = 'capl',
    kAppleMenuIcon = 'sapl',
    kBackwardArrowIcon = 'baro',
    kFavoriteItemsIcon = 'favr',
    kForwardArrowIcon = 'faro',
    kGridIcon = 'grid',
    kHelpIcon = 'help',
    kKeepArrangedIcon = 'arng',
    kLockedIcon = 'lock',
    kNoFilesIcon = 'nfil',
    kNoFolderIcon = 'nfld',
    kNoWriteIcon = 'nwrt',
    kProtectedApplicationFolderIcon = 'papp',
    kProtectedSystemFolderIcon = 'psys',
    kRecentItemsIcon = 'rcnt',
    kShortcutIcon = 'shrt',
    kSortAscendingIcon = 'asnd',
    kSortDescendingIcon = 'dsnd',
    kUnlockedIcon = 'ulck',
    kConnectToIcon = 'cnct',
    kGenericWindowIcon = 'gwin',
    kQuestionMarkIcon = 'ques',
    kDeleteAliasIcon = 'dali',
    kEjectMediaIcon = 'ejec',
    kBurningIcon = 'burn',
    kRightContainerArrowIcon = 'rcar'
};
```

Discussion
Icon Services defines constants for a number of miscellaneous icons. You can pass one of these constants in the `iconType` parameter of the function `GetIconRef` (page 17), for example.

Networking Icon Constants

Identify standard networking icons.

```
enum {
    kAppleTalkIcon = 'atlk',
    kAppleTalkZoneIcon = 'atzn',
    kAFPServerIcon = 'afps',
    kFTPServerIcon = 'ftps',
    kHTTPServerIcon = 'https',
    kGenericNetworkIcon = 'gnet',
    kIPFileServerIcon = 'isrv'
};
```

Discussion
Icon Services defines constants for a number of standard networking icons. You can pass one of these constants in the `iconType` parameter of the function `GetIconRef` (page 17), for example.

Special Folder Icon Constants

Identify special folder icons.

```
enum {
    kAppearanceFolderIcon = 'appr',
    kAppleExtrasFolderIcon = 'aexf',
    kAppleMenuFolderIcon = 'amnu',
    kApplicationsFolderIcon = 'apps',
    kApplicationSupportFolderIcon = 'asup',
    kAssistantsFolderIcon = 'astf',
    kColorSyncFolderIcon = 'prof',
    kContextualMenuItemsFolderIcon = 'cmnu',
    kControlPanelDisabledFolderIcon = 'ctrD',
    kControlPanelFolderIcon = 'ctrl',
    kControlStripModulesFolderIcon = 'sdvf',
    kDocumentsFolderIcon = 'docs',
    kExtensionsDisabledFolderIcon = 'extD',
    kExtensionsFolderIcon = 'extn',
    kFavoritesFolderIcon = 'favs',
    kFontsFolderIcon = 'font',
    kHelpFolderIcon = 'fhlp',
    kInternetFolderIcon = 'intf',
    kInternetPlugInFolderIcon = 'fnet',
    kInternetSearchSitesFolderIcon = 'issf',
    kLocalesFolderIcon = 'floc',
    kMacOSReadMeFolderIcon = 'morf',
    kPublicFolderIcon = 'pubf',
    kPreferencesFolderIcon = 'prff',
    kPrinterDescriptionFolderIcon = 'ppdf',
    kPrinterDriverFolderIcon = 'fprd',
    kPrintMonitorFolderIcon = 'prnt',
    kRecentApplicationsFolderIcon = 'rapp',
    kRecentDocumentsFolderIcon = 'rdoc',
    kRecentServersFolderIcon = 'rsrv',
    kScriptingAdditionsFolderIcon = 'fscr',
    kSharedLibrariesFolderIcon = 'flib',
    kScriptsFolderIcon = 'scrf',
    kShutdownItemsDisabledFolderIcon = 'shdD',
    kShutdownItemsFolderIcon = 'shdf',
    kSpeakableItemsFolder = 'spki',
    kStartupItemsDisabledFolderIcon = 'strD',
    kStartupItemsFolderIcon = 'strt',
    kSystemExtensionDisabledFolderIcon = 'macD',
    kSystemFolderIcon = 'macs',
    kTextEncodingsFolderIcon = 'ftex',
    kUsersFolderIcon = 'usrf',
    kUtilitiesFolderIcon = 'utif',
    kVoicesFolderIcon = 'fvoc'
};
```

Discussion
Icon Services defines constants for a number of special folder icons. You can pass one of these constants in the `iconType` parameter of the function `GetIconRef` (page 17), for example.

Standard Finder Icon Constants

Identify standard Finder icons.

```
enum {
    kClipboardIcon = 'CLIP',
    kClippingUnknownTypeIcon = 'clpu',
    kClippingPictureTypeIcon = 'clpp',
    kClippingTextTypeIcon = 'clpt',
    kClippingSoundTypeIcon = 'clps',
    kDesktopIcon = 'desk',
    kFinderIcon = 'FNDR',
    kFontSuitcaseIcon = 'FFIL',
    kFullTrashIcon = 'ftrh',
    kGenericApplicationIcon = 'APPL',
    kGenericCDROMIcon = 'cddr',
    kGenericControlPanelIcon = 'APPC',
    kGenericControlStripModuleIcon = 'sdev',
    kGenericComponentIcon = 'thng',
    kGenericDeskAccessoryIcon = 'APPD',
    kGenericDocumentIcon = 'docu',
    kGenericEditionFileIcon = 'edtf',
    kGenericExtensionIcon = 'INIT',
    kGenericFileServerIcon = 'srvr',
    kGenericFontIcon = 'ffil',
    kGenericFontScalerIcon = 'sclr',
    kGenericFloppyIcon = 'flpy',
    kGenericHardDiskIcon = 'hdsk',
    kGenericIDiskIcon = 'idsk',
    kGenericRemovableMediaIcon = 'rmov',
    kGenericMoverObjectIcon = 'movr',
    kGenericPCCardIcon = 'pcmc',
    kGenericPreferencesIcon = 'pref',
    kGenericQueryDocumentIcon = 'qery',
    kGenericRAMDiskIcon = 'ramd',
    kGenericSharedLibaryIcon = 'shlb',
    kGenericStationeryIcon = 'sdoc',
    kGenericSuitcaseIcon = 'suit',
    kGenericURLIcon = 'gurl',
    kGenericWORMIcon = 'worm',
    kInternationalResourcesIcon = 'ifil',
    kKeyboardLayoutIcon = 'kfil',
    kSoundFileIcon = 'sfil',
    kSystemSuitcaseIcon = 'zsys',
    kTrashIcon = 'trsh',
    kTrueTypeFontIcon = 'tfil',
    kTrueTypeFlatFontIcon = 'sfnt',
    kTrueTypeMultiFlatFontIcon = 'ttcf',
    kUserIDiskIcon = 'udsk',
    kUnknownFSObjectIcon = 'unfs',
    kInternationResourcesIcon = kInternationalResourcesIcon
};
```

Discussion
Icon Services defines constants for a number of standard Finder icons. You can pass one of these constants in the `iconType` parameter of the function `GetIconRef` (page 17), for example.

Standard Icon Badge Constants

Identify standard badges.

```
enum {
    kAppleScriptBadgeIcon = 'scrp',
    kLockedBadgeIcon = 'lbdg',
    kMountedBadgeIcon = 'mbdg',
    kSharedBadgeIcon = 'sbdg',
    kAliasBadgeIcon = 'abdg',
    kAlertCautionBadgeIcon = 'cbdg'
};
```

Discussion
Icon Services defines constants for a number of standard badges. You can pass one of these constants in the `iconType` parameter of the function `GetIconRef` (page 17), for example.

Users and Groups Icon Constants

Identify icons used in the Users and Groups control panel.

```
enum {
    kUserFolderIcon = 'ufld',
    kWorkgroupFolderIcon = 'wfld',
    kGuestUserIcon = 'gusr',
    kUserIcon = 'user',
    kOwnerIcon = 'susr',
    kGroupIcon = 'grup'
};
```

Discussion
Icon Services defines constants for a number of icons used in the Users and Groups control panel. You can pass one of these constants in the `iconType` parameter of the function `GetIconRef` (page 17), for example.

genericDocumentIconResource

Use the constants listed in "Standard Icon Resources" instead.

```
enum {
    genericDocumentIconResource = kGenericDocumentIconResource,
    genericStationeryIconResource = kGenericStationeryIconResource,
    genericEditionFileIconResource = kGenericEditionFileIconResource,
    genericApplicationIconResource = kGenericApplicationIconResource,
    genericDeskAccessoryIconResource = kGenericDeskAccessoryIconResource,
    genericFolderIconResource = kGenericFolderIconResource,
    privateFolderIconResource = kPrivateFolderIconResource,
    floppyIconResource = kFloppyIconResource,
    trashIconResource = kTrashIconResource,
    genericRAMDiskIconResource = kGenericRAMDiskIconResource,
    genericCDROMIconResource = kGenericCDROMIconResource,
    desktopIconResource = kDesktopIconResource,
    openFolderIconResource = kOpenFolderIconResource,
    genericHardDiskIconResource = kGenericHardDiskIconResource,
    genericFileServerIconResource = kGenericFileServerIconResource,
    genericSuitcaseIconResource = kGenericSuitcaseIconResource,
    genericMoverObjectIconResource = kGenericMoverObjectIconResource,
    genericPreferencesIconResource = kGenericPreferencesIconResource,
    genericQueryDocumentIconResource = kGenericQueryDocumentIconResource,
    genericExtensionIconResource = kGenericExtensionIconResource,
    systemFolderIconResource = kSystemFolderIconResource,
    appleMenuFolderIconResource = kAppleMenuFolderIconResource
};
```

Standard Icon Resources

Identify standard icon resources.

```
/*Icons for which both icon suites and 'SICN' resources exist*/
enum {
    kGenericDocumentIconResource = -4000,
    kGenericStationeryIconResource = -3985,
    kGenericEditionFileIconResource = -3989,
    kGenericApplicationIconResource = -3996,
    kGenericDeskAccessoryIconResource = -3991,
    kGenericFolderIconResource = -3999,
    kPrivateFolderIconResource = -3994,
    kFloppyIconResource = -3998,
    kTrashIconResource = -3993,
    kGenericRAMDiskIconResource = -3988,
    kGenericCDROMIconResource = -3987
};
/* Icons for which only 'SICN' resources exist*/
enum {
    kDesktopIconResource = -3992,
    kOpenFolderIconResource = -3997,
    kGenericHardDiskIconResource = -3995,
    kGenericFileServerIconResource = -3972,
    kGenericSuitcaseIconResource = -3970,
    kGenericMoverObjectIconResource = -3969
};
/*Icons for which only icon suites exist*/
enum {
    kGenericPreferencesIconResource = -3971,
    kGenericQueryDocumentIconResource = -16506,
    kGenericExtensionIconResource = -16415,
    kSystemFolderIconResource = -3983,
    kHelpIconResource = -20271,
    kAppleMenuFolderIconResource = -3982
};
enum {
    kStartupFolderIconResource = -3981,
    kOwnedFolderIconResource = -3980,
    kDropFolderIconResource = -3979,
    kSharedFolderIconResource = -3978,
    kMountedFolderIconResource = -3977,
    kControlPanelFolderIconResource = -3976,
    kPrintMonitorFolderIconResource = -3975,
    kPreferencesFolderIconResource = -3974,
    kExtensionsFolderIconResource = -3973,
    kFontsFolderIconResource = -3968,
    kFullTrashIconResource = -3984
};
```

startupFolderIconResource

Use the constants described in "Standard Icon Resources" instead.

```
enum {
    startupFolderIconResource = kStartupFolderIconResource,
    ownedFolderIconResource = kOwnedFolderIconResource,
    dropFolderIconResource = kDropFolderIconResource,
    sharedFolderIconResource = kSharedFolderIconResource,
    mountedFolderIconResource = kMountedFolderIconResource,
    controlPanelFolderIconResource = kControlPanelFolderIconResource,
    printMonitorFolderIconResource = kPrintMonitorFolderIconResource,
    preferencesFolderIconResource = kPreferencesFolderIconResource,
    extensionsFolderIconResource = kExtensionsFolderIconResource,
    fontsFolderIconResource = kFontsFolderIconResource,
    fullTrashIconResource = kFullTrashIconResource
};
```

atNone

Use the constants described in "Icon Alignment Constants" instead.

```
enum {
    atNone = kAlignNone,
    atVerticalCenter = kAlignVerticalCenter,
    atTop = kAlignTop,
    atBottom = kAlignBottom,
    atHorizontalCenter = kAlignHorizontalCenter,
    atAbsoluteCenter = kAlignAbsoluteCenter,
    atCenterTop = kAlignCenterTop,
    atCenterBottom = kAlignCenterBottom,
    atLeft = kAlignLeft,
    atCenterLeft = kAlignCenterLeft,
    atTopLeft = kAlignTopLeft,
    atBottomLeft = kAlignBottomLeft,
    atRight = kAlignRight,
    atCenterRight = kAlignCenterRight,
    atTopRight = kAlignTopRight,
    atBottomRight = kAlignBottomRight
};
```

svLarge1Bit

Use the constants described in "Icon Selector Constants" instead.

```
enum {
    svLarge1Bit = kSelectorLarge1Bit,
    svLarge4Bit = kSelectorLarge4Bit,
    svLarge8Bit = kSelectorLarge8Bit,
    svSmall1Bit = kSelectorSmall1Bit,
    svSmall4Bit = kSelectorSmall4Bit,
    svSmall8Bit = kSelectorSmall8Bit,
    svMini1Bit = kSelectorMini1Bit,
    svMini4Bit = kSelectorMini4Bit,
    svMini8Bit = kSelectorMini8Bit,
    svAllLargeData = kSelectorAllLargeData,
    svAllSmallData = kSelectorAllSmallData,
    svAllMiniData = kSelectorAllMiniData,
    svAll1BitData = kSelectorAll1BitData,
    svAll4BitData = kSelectorAll4BitData,
    svAll8BitData = kSelectorAll8BitData,
    svAllAvailableData = kSelectorAllAvailableData
};
```

ttNone

Use the constants described in "Icon Transformation Constants" instead.

```
enum {
    ttNone = kTransformNone,
    ttDisabled = kTransformDisabled,
    ttOffline = kTransformOffline,
    ttOpen = kTransformOpen,
    ttLabel1 = kTransformLabel1,
    ttLabel2 = kTransformLabel2,
    ttLabel3 = kTransformLabel3,
    ttLabel4 = kTransformLabel4,
    ttLabel5 = kTransformLabel5,
    ttLabel6 = kTransformLabel6,
    ttLabel7 = kTransformLabel7,
    ttSelected = kTransformSelected,
    ttSelectedDisabled = kTransformSelectedDisabled,
    ttSelectedOffline = kTransformSelectedOffline,
    ttSelectedOpen = kTransformSelectedOpen
};
```

Result Codes

The table below shows the most common result codes returned by Icon Services and Utilities.

Result Code	Value	Description
`noMaskFoundErr`	-1000	Available in Mac OS X v10.0 and later.
`invalidIconRefErr`	-2580	The `IconRef` is not valid. Available in Mac OS X v10.0 and later.

Result Code	Value	Description
`noSuchIconErr`	-2581	The requested icon could not be found. Available in Mac OS X v10.0 and later.
`noIconDataAvailableErr`	-2582	The necessary icon data is not available. Available in Mac OS X v10.0 and later.

Gestalt Constants

You can check for version and feature availability information by using the Icon Services selectors defined in the Gestalt Manager. For more information, see *Gestalt Manager Reference*.

Deprecated Icon Services and Utilities Functions

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

Deprecated in Mac OS X v10.3

FlushIconRefs

Reclaims memory used by the specified icon if the memory is purgeable. (Deprecated in Mac OS X v10.3. There is no replacement; this function was included to facilitate porting classic applications to Carbon, but it serves no useful purpose in Mac OS X.)

```
OSErr FlushIconRefs (
   OSType creator,
   OSType iconType
);
```

Parameters

creator

The creator code of the file whose icon data is to be flushed.

iconType

The type code of the file whose icon data is to be flushed.

Return Value

A result code. See "Icon Services and Utilities Result Codes" (page 57).

Special Considerations

This function does nothing in Mac OS X.

Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.3.

Not available to 64-bit applications.

Declared In

`IconsCore.h`

FlushIconRefsByVolume

On a given volume, reclaims memory used by purgeable icons. (Deprecated in Mac OS X v10.3. There is no replacement; this function was included to facilitate porting classic applications to Carbon, but it serves no useful purpose in Mac OS X.)

```
OSErr FlushIconRefsByVolume (
   SInt16 vRefNum
);
```

Parameters

vRefNum

The volume whose icon cache is to be flushed.

Return Value

A result code. See "Icon Services and Utilities Result Codes" (page 57).

Discussion

Calling this function locks the bitmap data of all `IconRefs` with non-zero reference counts (that is, all `IconRefs` that are in use) on the volume. The Finder normally maintains a number of `IconRefs` with non-zero reference counts, so you should use the function `FlushIconRefs` (page 59) instead of the `FlushIconRefsByVolume` function whenever feasible.

Special Considerations

This function does nothing in Mac OS X.

Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.3.

Not available to 64-bit applications.

Declared In

`IconsCore.h`

GetIconSizesFromIconRef

Provides an `IconSelectorValue` indicating the sizes and depths of icon data available for an `IconRef`. (Deprecated in Mac OS X v10.3. Use `IsDataAvailableInIconRef` (page 27) instead.)

```
OSErr GetIconSizesFromIconRef (
   IconSelectorValue iconSelectorInput,
   IconSelectorValue *iconSelectorOutputPtr,
   IconServicesUsageFlags iconServicesUsageFlags,
   IconRef theIconRef
);
```

Parameters

iconSelectorInput

The icon sizes and depths you are requesting from the `IconRef`. For a description of the possible values, see "Icon Selector Constants" (page 43).

iconSelectorOutputPtr

On return, this points to a value describing the icon sizes and depths available for the specified `IconRef`. For a description of the possible values, see "Icon Selector Constants" (page 43).

iconServicesUsageFlags

Reserved for future use. Pass the `kIconServicesNormalUsageFlag` constant in this parameter.

theIconRef

The icon family to query.

Return Value
A result code. See "Icon Services and Utilities Result Codes" (page 57).

Discussion
Note that this function may be very time-consuming, as Icon Services may have to search disks or even the network to obtain the requested data.

Special Considerations

Because this function is so time-consuming, it is more efficient to simply query the icon for particular data using the function `IsDataAvailableInIconRef` (page 27).

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.3.
Not available to 64-bit applications.

Declared In
`Icons.h`

Deprecated in Mac OS X v10.5

AddIconToSuite

Adds an icon to an icon suite. (Deprecated in Mac OS X v10.5. Use Icon Services instead.)

```
OSErr AddIconToSuite (
   Handle theIconData,
   IconSuiteRef theSuite,
   ResType theType
);
```

Parameters

theIconData

A handle to the data for the new icon to be added to the icon suite. You can obtain a handle to icon data using various functions, such as `GetIcon` (page 65) or `GetResource`.

The handle to the icon data is added at the location reserved for icon data of the type specified by `theType`. If the icon suite already includes a handle to icon data for that type, this function replaces the handle to the old data without disposing of it. In this case you may want to call the `GetIconFromSuite` (page 67) function first to obtain the old handle so that you can dispose of it.

The handles that you add to the suite do not have to be associated with a resource fork. For example, your application might get icon data from the desktop database rather than reading it from a resource, or your application might read icon data from a resource and then detach it.

theSuite

A handle to the icon suite to which to add the icon.

theType

The resource type of the new icon. The resource type should be that of an icon family member.

Return Value
A result code. See "Icon Services and Utilities Result Codes" (page 57).

Discussion
This function is most often used to read icons into an empty icon suite created with the `NewIconSuite` (page 79) function.

Special Considerations

This function may move or purge memory blocks in the application heap. For that reason, your application should not call it at interrupt time.

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
`Icons.h`

DisposeCIcon

Releases the memory occupied by a color icon structure. (Deprecated in Mac OS X v10.5. Use Icon Services instead.)

```
void DisposeCIcon (
   CIconHandle theIcon
);
```

Parameters

theIcon

A handle to the color icon structure to dispose of, previously obtained from the `GetCIcon` (page 64) function.

Discussion
To dispose of a handle obtained from `GetIcon` or `GetResource`, use the `ReleaseResource` function to release the memory occupied by the icon resource data.

Special Considerations

This function may move or purge memory blocks in the application heap. For that reason, your application should not call it at interrupt time.

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
`Icons.h`

DisposeIconSuite

Releases the memory occupied by an icon suite. (Deprecated in Mac OS X v10.5. Use Icon Services instead.)

```
OSErr DisposeIconSuite (
   IconSuiteRef theIconSuite,
   Boolean disposeData
);
```

Parameters

theIconSuite

A handle to the icon suite to be disposed of.

disposeData

A Boolean value indicating whether or not to dispose of handles in the icon suite that are not associated with a resource fork.

Set this value to `TRUE` to automatically release icon data that is associated with the specified icon suite but not explicitly associated with a resource fork. If you set this value to `FALSE`, the function does not dispose of any icon data that is associated with the specified icon suite.

Return Value

A result code. See "Icon Services and Utilities Result Codes" (page 57).

Discussion

This function does not release the memory of any icons explicitly associated with an open resource fork, that is, any handles to icon resource data that your application added to the suite using the functions `GetIconSuite` (page 69) or `AddIconToSuite` (page 61). For handles to icon data that your application added to the icon suite using `AddIconToSuite` (for example, if your application read in an icon resource, detached it, then added the handle to the suite), you can request that `AddIconToSuite` release the memory associated with the handles.

Special Considerations

This function may move or purge memory blocks in the application heap. For that reason, your application should not call it at interrupt time.

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

`Icons.h`

ForEachIconDo

Performs an action on one or more icons in an icon suite. (Deprecated in Mac OS X v10.5. Use Icon Services instead.)

```
OSErr ForEachIconDo (
   IconSuiteRef theSuite,
   IconSelectorValue selector,
   IconActionUPP action,
   void *yourDataPtr
);
```

Parameters

theSuite

A handle to an icon suite.

selector

Indicates which icons in the suite to perform the operation on. See "Icon Selector Constants" (page 43) for a description of the values you can use in this parameter.

action

A universal procedure pointer to your icon action callback function. The `ForEachIconDo` function uses this icon action function to perform an action on the specified icons in the icon suite.

`ForEachIconDo` calls your icon action function once for each type of icon specified in the `selector` parameter. `ForEachIconDo` passes to your icon action function a handle to the icon to perform the action on. Your icon action function should perform any action as indicated by the `yourDataPtr` parameter and return a result code.

See the `IconActionProcPtr` (page 36) callback for more information about icon action functions.

yourDataPtr

A pointer to data or other information required by your icon action function that is passed to your icon action function. Typically, you use this parameter to specify which action your icon action function should perform.

Return Value
A result code. See "Icon Services and Utilities Result Codes" (page 57). The result code returned by your icon action function. If your icon action function returns a nonzero function result, `ForEachIconDo` immediately returns to the application.

Special Considerations

This function may move or purge memory blocks in the application heap. For that reason, your application should not call it at interrupt time.

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
`Icons.h`

GetCIcon

Gets a handle to a color icon of resource type `'cicn'`. (Deprecated in Mac OS X v10.5. Use Icon Services instead.)

```
CIconHandle GetCIcon (
   SInt16 iconID
);
```

Parameters

iconID

The resource ID for an icon of resource type `'cicn'`. In general, you should specify your icon resources as purgeable.

Return Value
A handle to the `CIcon` (page 38) structure for the icon, or `NULL` if the function could not find the resource.

Discussion
The function searches the current resource chain for the resource. If it finds the resource, it reads the resource, creates a color icon structure for the icon, and initializes the fields of the structure according to the information contained in the `'cicn'` resource.

To draw an icon obtained from this function in a specified rectangle, you can use either the `PlotCIcon` (page 80) function, or the `PlotCIconHandle` (page 81) function. The latter function allows you to specify transforms and alignments.

When you are finished with a handle obtained from this function, use the `DisposeCIcon` (page 62) function to release the memory occupied by the color icon structure.

Special Considerations

This function may move or purge memory blocks in the application heap. For that reason, your application should not call it at interrupt time.

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
`Icons.h`

GetIcon

Gets a handle to an icon resource of type `'ICON'`. (Deprecated in Mac OS X v10.5. Use Icon Services instead.)

```
Handle GetIcon (
   SInt16 iconID
);
```

Parameters

iconID

The resource ID for an icon of resource type `'ICON'`. The function searches the current resource chain for the resource. In general, you should specify your icon resources as purgeable.

Return Value
A handle to the icon with the specified ID or `NULL` if the function could not find the resource.

Discussion
To draw an icon obtained from this function in a specified rectangle, you can use either the `PlotIcon` (page 82) function, or the `PlotIconHandle` (page 82) function. The latter function allows you to specify transforms and alignments.

When you are finished using a handle obtained from this function, use the `ReleaseResource` function to release the memory occupied by the icon resource data.

Special Considerations

This function may move or purge memory blocks in the application heap. For that reason, your application should not call it at interrupt time.

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
`Icons.h`

GetIconCacheData

Gets the data associated with an icon cache. (Deprecated in Mac OS X v10.5. Use Icon Services instead.)

```
OSErr GetIconCacheData (
   IconCacheRef theCache,
   void **theData
);
```

Parameters

theCache

A handle to the icon cache whose data is desired.

theData

On return, a pointer to a pointer to the data associated with the icon cache.

You associate data with an icon cache when you first create the cache using the `MakeIconCache` (page 78) function. You can also set this data using the `SetIconCacheData` (page 99) function.

Return Value
A result code. See "Icon Services and Utilities Result Codes" (page 57).

Discussion
All the Icon Utilities functions that accept a handle to an icon suite also accept a handle to an icon cache.

Special Considerations

This function may move or purge memory blocks in the application heap. For that reason, your application should not call it at interrupt time.

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
`Icons.h`

GetIconCacheProc

Gets the icon getter function associated with an icon cache. (Deprecated in Mac OS X v10.5. Use Icon Services instead.)

```
OSErr GetIconCacheProc (
   IconCacheRef theCache,
   IconGetterUPP *theProc
);
```

Parameters

theCache

A handle to the icon cache whose associated icon getter function is desired.

theProc

On return, a pointer to the universal procedure pointer to the icon getter callback function associated with the specified cache. See the `IconGetterProcPtr` (page 37) callback for more information on icon getter functions.

Return Value

A result code. See "Icon Services and Utilities Result Codes" (page 57).

Discussion

All the Icon Utilities functions that accept a handle to an icon suite also accept a handle to an icon cache. An icon cache is like an icon suite except that it also contains a pointer to an icon getter callback function and a pointer to data that can be used as a reference constant. An icon cache typically does not contain handles to the icon resources for all icon family members. Instead, if the icon cache does not contain an entry for a specific type of icon in an icon family, the Icon Utilities functions call your application's icon getter function to retrieve the data for that icon type.

Special Considerations

This function may move or purge memory blocks in the application heap. For that reason, your application should not call it at interrupt time.

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

`Icons.h`

GetIconFromSuite

Gets an icon from an icon suite. (Deprecated in Mac OS X v10.5. Use Icon Services instead.)

```
OSErr GetIconFromSuite (
   Handle *theIconData,
   IconSuiteRef theSuite,
   ResType theType
);
```

Parameters

theIconData

On return, a pointer to a handle to the data for the requested icon. If an icon of the specified type does not exist in the given icon suite, this parameter is `NULL`.

If you intend to dispose of the handle, pass a `NULL` handle to the `AddIconToSuite` (page 61) function to delete the corresponding entry in the suite.

You can use the handle returned by this function to manipulate the icon data, for example, to alter its color or add three-dimensional shading. However, you should not use the returned handle to draw the icon with other Icon Utilities functions.

To plot an icon from an icon suite, you should normally use the `PlotIconSuite` (page 86) function. The `PlotIconHandle` (page 82) function may not draw the icon correctly if you pass it the handle returned in this parameter.

theSuite

A handle to the icon suite from which to get the icon.

theType

The resource type of the desired icon.

Return Value

A result code. See "Icon Services and Utilities Result Codes" (page 57).

Special Considerations

This function may move or purge memory blocks in the application heap. For that reason, your application should not call it at interrupt time.

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

`Icons.h`

GetIconRefFromFile

Provides an `IconRef` object for a file, folder or volume. (Deprecated in Mac OS X v10.5. Use `GetIconRefFromFileInfo` (page 18) instead.)

```
OSErr GetIconRefFromFile (
   const FSSpec *theFile,
   IconRef *theIconRef,
   SInt16 *theLabel
);
```

Parameters

theFile

A pointer to the `FSSpec` structure specifying the file, folder or volume for the `IconRef`.

theIconRef

On return, a pointer to an `IconRef` object. You are responsible for releasing the object by calling `ReleaseIconRef` (page 33).

theLabel

On return, a pointer to the file or folder's label.

Return Value
A result code. See "Icon Services and Utilities Result Codes" (page 57).

Discussion
Use this function if you have no information about the file object passed in the `theFile` parameter. If you have already called the File System Manager function `PBGetCatInfo`, you can use the function `GetIconRefFromFolder` (page 19) if the object is a folder without custom icons or the function `GetIconRef` (page 17) if the object is a file without custom icons.

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
`IconsCore.h`

GetIconSuite

Creates an icon suite in memory that contains handles to a specified icon family's resources. (Deprecated in Mac OS X v10.5. Use Icon Services instead.)

```
OSErr GetIconSuite (
   IconSuiteRef *theIconSuite,
   SInt16 theResID,
   IconSelectorValue selector
);
```

Parameters

theIconSuite

On return, a pointer to a handle to an icon suite for the requested icon family, for which this function allocates the memory. To release the memory occupied by an icon suite, you must use the `DisposeIconSuite` function.

theResID

The resource ID of the icons in the icon family to be read into memory. In general, you should specify your icon resources as purgeable.

selector

Indicates which icons from the icon family to include in the icon suite. See "Icon Selector Constants" (page 43) for a description of the values you can use in this parameter.

Return Value
A result code. See "Icon Services and Utilities Result Codes" (page 57).

Discussion
When you create an icon suite from icon family resources, the associated resource file should remain open while you use Icon Utilities functions. If you call the `SetResLoad` function with the `load` parameter set to `FALSE` before you call this function, the suite is filled with unloaded resource handles.

When you create an icon suite using this function, it sets the default label for the suite to none. To set a new default label for an icon suite, use the `SetSuiteLabel` (page 100) function. To perform operations on one or more icons in an icon suite, use the `ForEachIconDo` (page 63) function. To draw the icon described by the icon suite using the icon family member that is most suitable for the current bit depth of the display device, use the `PlotIconSuite` (page 86) function.

As an alternative to this function, you can also create an empty icon suite using the `NewIconSuite` (page 79) function and then add icons to it one at a time using the `AddIconToSuite` (page 61) function.

Special Considerations

This function may move or purge memory blocks in the application heap. For that reason, your application should not call it at interrupt time.

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

`Icons.h`

GetLabel

Gets the color and string used for a given label in the Label menu of the Finder and in the Labels control panel. (Deprecated in Mac OS X v10.5. Use Icon Services instead.)

```
OSErr GetLabel (
   SInt16 labelNumber,
   RGBColor *labelColor,
   Str255 labelString
);
```

Parameters

labelNumber

An integer from 1 to 7 indicating which label's information is requested.

labelColor

On return, a pointer to the color of the specified label.

labelString

On return, the string associated with the specified label.

Return Value

A result code. See "Icon Services and Utilities Result Codes" (page 57).

Special Considerations

This function may move or purge memory blocks in the application heap. For that reason, your application should not call it at interrupt time.

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
`Icons.h`

GetSuiteLabel

Gets the default label setting associated with an icon suite. (Deprecated in Mac OS X v10.5. Use Icon Services instead.)

```
SInt16 GetSuiteLabel (
   IconSuiteRef theSuite
);
```

Parameters

`theSuite`

A handle to an icon suite.

Return Value
The default label setting associated with the specified icon suite. The default label setting is an integer from 1 to 7 that specifies which of the label colors shown in the Finder's Label menu is applied to icons of that suite when your application displays them. The function returns 0 if the suite doesn't have a label. You can override the default label setting for a suite by specifying a label in the `transform` parameter of the `PlotIconSuite` (page 86) function. To get information about the color and string for a specific label, you can use the `GetLabel` (page 70) function.

Special Considerations

This function may move or purge memory blocks in the application heap. For that reason, your application should not call it at interrupt time.

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
`Icons.h`

IconFamilyToIconSuite

Provides icon suite data for a given icon family. (Deprecated in Mac OS X v10.5. Use Icon Services instead.)

```
OSErr IconFamilyToIconSuite (
   IconFamilyHandle iconFamily,
   IconSelectorValue whichIcons,
   IconSuiteRef *iconSuite
);
```

Parameters

`iconFamily`

A handle to an `iconFamily` data structure to use as a source for icon data. For more information on the `IconFamily` data structure, see 'icns'.

whichIcons

The depths and sizes of icons to extract from the IconFamily data structure. For a description of the possible values, see "Icon Selector Constants" (page 43).

iconSuite

On return, a pointer to the structure which contains icon data as specified in the `iconFamily` and `whichIcons` parameters. Icon Services returns `NULL` if no appropriate icon data is found.

Return Value
A result code. See "Icon Services and Utilities Result Codes" (page 57).

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
`Icons.h`

IconIDToRgn

Converts the icon mask in an icon family to a region. (Deprecated in Mac OS X v10.5. Use Icon Services instead.)

```
OSErr IconIDToRgn (
   RgnHandle theRgn,
   const Rect *iconRect,
   IconAlignmentType align,
   SInt16 iconID
);
```

Parameters

theRgn

On return, a handle to the requested region. You must allocate memory for the region handle before calling this function.

The returned region corresponds to the icon's mask (the mask defined by either an `'ICN#'` or `'ics#'` resource in an icon family, according to the rectangle and alignment specified in the `iconRect` and `align` parameters).

Once you have a region that describes the icon mask for a given icon, you can use it to perform accurate hit-testing and outline dragging of the icon in your application.

iconRect

A pointer to the rectangle in which to draw the icon, specified in local coordinates of the current graphics port. The function uses this rectangle as the bounding box of the region. The function determines, from the size of the rectangle specified here, which icon mask to use from the specified icon family.

align

Specifies how the function should align the mask within the rectangle. See "Icon Alignment Constants" (page 41) for a description of the values you can use in this parameter.

iconID

The resource ID of the icon for which to create a region. In general, you should specify your icon resources as purgeable.

Return Value
A result code. See "Icon Services and Utilities Result Codes" (page 57).

Special Considerations

This function may move or purge memory blocks in the application heap. For that reason, your application should not call it at interrupt time.

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

`Icons.h`

IconMethodToRgn

Converts, to a region, the mask for an icon that `IconMethodToRgn` obtains with the aid of your icon getter callback function. (Deprecated in Mac OS X v10.5. Use Icon Services instead.)

```
OSErr IconMethodToRgn (
   RgnHandle theRgn,
   const Rect *iconRect,
   IconAlignmentType align,
   IconGetterUPP theMethod,
   void *yourDataPtr
);
```

Parameters

theRgn

On return, a handle to the requested region. You must allocate memory for the region handle before calling this function. Once you have a region that describes the icon mask for a given icon, you can use it to perform accurate hit-testing and outline dragging of the icon in your application.

iconRect

A pointer to the rectangle in which to draw the icon, specified in local coordinates of the current graphics port. The function obtains the data for the icon mask from your icon getter function and then converts the icon mask to a region. The function uses the rectangle specified in this parameter as the bounding box of the region.

align

Specifies how the function should align the mask within the rectangle. See "Icon Alignment Constants" (page 41) for a description of the values you can use in this parameter.

theMethod

A universal procedure pointer to your icon getter callback function. `IconMethodToRgn` passes to your icon getter function the type of the icon to get and the value specified in the `yourDataPtr` parameter. The `IconMethodToRgn` function examines the size of the rectangle and requests the appropriate icon from your icon getter function—an icon of icon type `'ICN#'` or `'ics#'`. Your icon getter function should return a handle to the data of the requested icon type. The `IconMethodToRgn` function extracts the mask from the icon data that your icon getter function returns. If your icon getter function returns data that does not correspond to an icon of type `'ICN#'` or type `'ics#'`, `IconMethodToRgn` attempts to generate a mask from the returned data.

Your icon getter function can get the data for the icon and its mask using whatever method is appropriate to your application. For example, your application might maintain its own cache of icons (and pass a pointer to it in the `yourDataPtr` parameter) or use its icon getter function to get an icon from the desktop database.

See the `IconGetterProcPtr` (page 37) callback for more information on creating an icon getter function.

Return Value
A result code. See "Icon Services and Utilities Result Codes" (page 57).

Special Considerations

This function may move or purge memory blocks in the application heap. For that reason, your application should not call it at interrupt time.

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
`Icons.h`

IconRefToRgn

Converts an Icon Services icon into a QuickDraw region. (Deprecated in Mac OS X v10.5. Use `IconRefToHIShape` (page 25) instead.)

```
OSErr IconRefToRgn (
   RgnHandle theRgn,
   const Rect *iconRect,
   IconAlignmentType align,
   IconServicesUsageFlags iconServicesUsageFlags,
   IconRef theIconRef
);
```

Parameters

theRgn

A handle to the requested region. You must call the QuickDraw function `NewRegion` to allocate memory for the region handle before calling the `IconRefToRgn` function.

iconRect

A pointer to the rectangle defining the area that Icon Services uses as the bounding box of the region.

align
The value which determines how Icon Services aligns the region within the rectangle. For a description of possible return values, see "Icon Alignment Constants" (page 41).

iconServicesUsageFlags
Reserved for future use. Pass the `kIconServicesNormalUsageFlag` constant in this parameter.

theIconRef
The `IconRef` for the icon family to use for drawing the requested region.

Return Value
A result code. See "Icon Services and Utilities Result Codes" (page 57).

Discussion
Icon Services uses the rectangle and alignment values to automatically select the icon used to generate the region data.

This function is similar to the Icon Utilities function `IconSuiteToRegion`.

Icon Services uses the icon's black-and-white mask to determine the region data, even if you provide a deep mask.

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
`Icons.h`

IconSuiteToIconFamily

Provides `IconFamily` data for a specified `IconSuite`. (Deprecated in Mac OS X v10.5. Use Icon Services instead.)

```
OSErr IconSuiteToIconFamily (
   IconSuiteRef iconSuite,
   IconSelectorValue whichIcons,
   IconFamilyHandle *iconFamily
);
```

Parameters
iconSuite
The `IconSuiteRef` to use as a source for icon data.

whichIcons
The depths and sizes of icons to extract from the `iconFamily` data structure. For a description of the possible values, see "Icon Selector Constants" (page 43).

iconFamily
On return, a pointer to a handle to the structure which contains icon data as specified in the `iconSuite` and `whichIcons` parameters. Icon Services returns `NULL` if no appropriate icon data is found. For more information on the `IconFamily` data structure, see 'icns'.

Return Value
A result code. See "Icon Services and Utilities Result Codes" (page 57).

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
`Icons.h`

IconSuiteToRgn

Converts the icon mask in an icon suite to a region. (Deprecated in Mac OS X v10.5. Use Icon Services instead.)

```
OSErr IconSuiteToRgn (
   RgnHandle theRgn,
   const Rect *iconRect,
   IconAlignmentType align,
   IconSuiteRef theIconSuite
);
```

Parameters

theRgn

On return, a handle to the requested region. You must allocate memory for the region handle before calling this function.

The returned region corresponds to the icon's mask (the mask defined by either an `'ICN#'` or `'ics#'` entry in an icon suite, according to the rectangle and alignment specified in the `iconRect` and `align` parameters).

Once you have a region that describes the icon mask for a given icon, you can use it to perform accurate hit-testing and outline dragging of the icon in your application.

iconRect

A pointer to the rectangle in which the icon is to be drawn, specified in local coordinates of the current graphics port. The function uses this rectangle as the bounding box of the region. The function determines, from the size of the rectangle specified here, which icon mask to use from the icon suite.

align

Specifies how the function should align the region within the rectangle. See "Icon Alignment Constants" (page 41) for a description of the values you can use in this parameter.

theIconSuite

A handle to an icon suite.

Return Value
A result code. See "Icon Services and Utilities Result Codes" (page 57).

Special Considerations

This function may move or purge memory blocks in the application heap. For that reason, your application should not call it at interrupt time.

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
`Icons.h`

LoadIconCache

Loads into an icon cache a handle to the appropriate icon data for a specified destination rectangle and the current bit depth, for drawing later with a specified alignment and transform. (Deprecated in Mac OS X v10.5. Use Icon Services instead.)

```
OSErr LoadIconCache (
   const Rect *theRect,
   IconAlignmentType align,
   IconTransformType transform,
   IconCacheRef theIconCache
);
```

Parameters

theRect
A pointer to the rectangle in which to draw the icon, specified in local coordinates of the current graphics port. The function uses the rectangle specified in this parameter and the bit depth of the display device to determine which icon type to load into the cache.

align
Specifies how to align the icon within the rectangle. See "Icon Alignment Constants" (page 41) for a description of the values you can use in this parameter.

transform
Specifies how to modify the appearance of the icon. See "Icon Transformation Constants" (page 43) for a description of the values you can use in this parameter.

theIconCache
A reference to the icon cache into which to load the icon data.

Return Value
A result code. See "Icon Services and Utilities Result Codes" (page 57).

Discussion
This function can be useful, for example, if you suspect that the icon may be drawn at a time not convenient for loading resource data (for instance, when the resource fork isn't in the current resource chain). The function uses the same criteria as the `PlotIconSuite` (page 86) function to select the icon to load.

This function uses the icon getter callback function associated with the icon cache to get the appropriate icon. The icon getter function returns a handle to the requested icon data, and `LoadIconCache` adds the returned handle to the entry for that icon in the icon cache.

After calling this function, you can pass the same parameters to `PlotIconSuite` to plot the icon data. Note that if you specify an alignment when you call `LoadIconCache`, then call `PlotIconSuite` and specify no alignment, `PlotIconSuite` draws the icon using the alignment that you originally specified to `LoadIconCache`.

All the Icon Utilities functions that accept a handle to an icon suite also accept a handle to an icon cache.

Special Considerations

This function may move or purge memory blocks in the application heap. For that reason, your application should not call it at interrupt time.

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
`Icons.h`

MakeIconCache

Gets a handle to an empty icon cache. (Deprecated in Mac OS X v10.5. Use Icon Services instead.)

```
OSErr MakeIconCache (
   IconCacheRef *theCache,
   IconGetterUPP makeIcon,
   void *yourDataPtr
);
```

Parameters

theCache
On return, a pointer to a handle to the new, empty icon cache. The function allocates the necessary memory. You can add icon data to the new cache using the `LoadIconCache` (page 77) function.

makeIcon
A universal procedure pointer to the icon getter callback function to associate with the icon cache. See the `IconGetterProcPtr` (page 37) callback for more information on icon getter callback functions.

yourDataPtr
A pointer to the data to associate with the icon cache.

Return Value
A result code. See "Icon Services and Utilities Result Codes" (page 57).

Discussion
All the Icon Utilities functions that accept a handle to an icon suite also accept a handle to an icon cache. An icon cache is like an icon suite except that it also contains a pointer to an icon getter callback function and a pointer to data that can be used as a reference constant. An icon cache typically does not contain handles to the icon resources for all icon family members. Instead, if the icon cache does not contain an entry for a specific type of icon in an icon family, the Icon Utilities functions call your application's icon getter function to retrieve the data for that icon type.

Special Considerations

This function may move or purge memory blocks in the application heap. For that reason, your application should not call it at interrupt time.

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
`Icons.h`

NewIconSuite

Gets a handle to an empty icon suite. (Deprecated in Mac OS X v10.5. Use Icon Services instead.)

```
OSErr NewIconSuite (
   IconSuiteRef *theIconSuite
);
```

Parameters

theIconSuite

On return, a pointer to a handle to a new, empty icon suite. Use the `AddIconToSuite` (page 61) function to add handles to icon data.

Return Value

A result code. See "Icon Services and Utilities Result Codes" (page 57).

Discussion

When you create an icon suite using this function, it sets the default label for the suite to none. To set a new default label for an icon suite, use the `SetSuiteLabel` (page 100) function. `NewIconSuite` allocates the memory for the icon suite handle. To release the memory occupied by an icon suite, you must use the `DisposeIconSuite` (page 62) function.

Special Considerations

This function may move or purge memory blocks in the application heap. For that reason, your application should not call it at interrupt time.

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

`Icons.h`

OverrideIconRefFromResource

Replaces the bitmaps in an `IconRef` with bitmaps from a specified resource file. (Deprecated in Mac OS X v10.5. Use `OverrideIconRef` (page 30) instead.)

```
OSErr OverrideIconRefFromResource (
   IconRef theIconRef,
   const FSSpec *resourceFile,
   SInt16 resourceID
);
```

Parameters

theIconRef

An `IconRef` to be updated.

resourceFile

A pointer to the file system specification structure for the resource file containing the replacement bitmaps.

resourceID

The resource ID containing the replacement bitmaps. This value must be non-zero. You should provide a resource of type `'icns'` if possible. If an `'icns'` resource is not available, Icon Services uses standard icon suite resources, such as `'ICN#'`, instead.

Return Value
A result code. See "Icon Services and Utilities Result Codes" (page 57).

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
`IconsCore.h`

PlotCIcon

Draws a color icon of resource type `'cicn'` to which you have a handle. (Deprecated in Mac OS X v10.5. Use Icon Services instead.)

```
void PlotCIcon (
   const Rect *theRect,
   CIconHandle theIcon
);
```

Parameters

theRect

A pointer to the rectangle in which to draw the icon, specified in local coordinates of the current graphics port.

theIcon

A handle to the color icon structure of the color icon to draw. You can obtain a handle to the icon using the `GetCIcon` (page 64) function, or `GetResource` or other Resource Manager functions.

Discussion
The `iconMask` field of the `CIcon` (page 38) structure determines which pixels in the `iconPMap` field are drawn and which are not. Only pixels with 1s in corresponding positions in the `iconMask` field are drawn. If the screen depth is 1 or 2 bits per pixel, this function uses the `iconBMap` field instead of the `iconPMap` field (unless the `rowBytes` field of `IconBMap` contains 0, indicating that there is no bitmap for the icon).

When this function draws the icon, it uses the `bounds` field of `iconPMap` as the source rectangle of the image. If the destination rectangle is not the same size as the icon or its mask, the function stretches or shrinks the icon to fit. The icon's pixels are remapped to the current depth and color table, if necessary. The `bounds` fields of `iconPMap`, `iconBMap`, and `iconMask` are expected to be equal in size.

Unlike `PlotCIconHandle` (page 81), this function does not allow you to specify any transforms or alignment. This function uses the QuickDraw function `CopyMask` and doesn't send any of its drawing commands through QuickDraw bottleneck functions. Therefore, calls to this function are not recorded as pictures.

Special Considerations

This function may move or purge memory blocks in the application heap. For that reason, your application should not call it at interrupt time.

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
`Icons.h`

PlotCIconHandle

Draws an icon of resource type `'cicn'` to which you have a handle. (Deprecated in Mac OS X v10.5. Use Icon Services instead.)

```
OSErr PlotCIconHandle (
   const Rect *theRect,
   IconAlignmentType align,
   IconTransformType transform,
   CIconHandle theCIcon
);
```

Parameters

theRect
A pointer to the rectangle in which to draw the icon, specified in local coordinates of the current graphics port.

align
Specifies how the function should align the icon within the rectangle. See "Icon Alignment Constants" (page 41) for a description of the values you can use in this parameter.

transform
Specifies how the function should modify the appearance of the icon. See "Icon Transformation Constants" (page 43) for a description of the values you can use in this parameter.

theCIcon
A handle to the color icon structure of the icon to draw. You can obtain a handle to the icon using the `GetCIcon` (page 64) function or `GetResource` or other Resource Manager functions.

Return Value
A result code. See "Icon Services and Utilities Result Codes" (page 57).

Discussion
Unlike `PlotCIcon` (page 80), this function doesn't honor the current foreground and background colors.

Special Considerations

This function may move or purge memory blocks in the application heap. For that reason, your application should not call it at interrupt time.

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
`Icons.h`

PlotIcon

Draws an icon of resource type `'ICON'` to which you have a handle. (Deprecated in Mac OS X v10.5. Use Icon Services instead.)

```
void PlotIcon (
   const Rect *theRect,
   Handle theIcon
);
```

Parameters

theRect

A pointer to the rectangle in which to draw the icon, specified in local coordinates of the current graphics port.

theIcon

A handle to the icon to draw. You must have previously obtained this handle using the `GetIcon` (page 65) function, or `GetResource` or other Resource Manager functions.

Discussion

This function does not allow you to specify any transforms or alignment. The `PlotIcon` function uses the QuickDraw function `CopyBits` with the `srcCopy` transfer mode. To plot an icon of resource type `'ICON'` with a specified transform and alignment, use the `PlotIconHandle` (page 82) function.

If the destination rectangle is not 32 by 32 pixels, the function stretches or shrinks the icon to fit.

Special Considerations

This function may move or purge memory blocks in the application heap. For that reason, your application should not call it at interrupt time.

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

`Icons.h`

PlotIconHandle

Draws an icon of resource type `'ICON'` or `'ICN#'` to which you have a handle. (Deprecated in Mac OS X v10.5. Use Icon Services instead.)

```
OSErr PlotIconHandle (
   const Rect *theRect,
   IconAlignmentType align,
   IconTransformType transform,
   Handle theIcon
);
```

Parameters

theRect

A pointer to the rectangle in which to draw the icon, specified in local coordinates of the current graphics port.

align

Specifies how the function should align the icon within the rectangle. See "Icon Alignment Constants" (page 41) for a description of the values you can use in this parameter.

transform

Specifies how the function should modify the appearance of the icon. See "Icon Transformation Constants" (page 43) for a description of the values you can use in this parameter.

theIcon

A handle to the icon to draw. You must have previously obtained a handle to the icon using the `GetIcon` (page 65) function, or `GetResource` or other Resource Manager functions.

Return Value
A result code. See "Icon Services and Utilities Result Codes" (page 57).

Discussion
To plot an icon from an icon suite, you should normally use `PlotIconSuite` (page 86). This function may not draw the icon correctly if you pass it the handle returned in the `theIconData` parameter of `GetIconFromSuite` (page 67).

Special Considerations

This function may move or purge memory blocks in the application heap. For that reason, your application should not call it at interrupt time.

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
`Icons.h`

PlotIconID

Draws the icon described by an icon family. (Deprecated in Mac OS X v10.5. Use Icon Services instead.)

```
OSErr PlotIconID (
   const Rect *theRect,
   IconAlignmentType align,
   IconTransformType transform,
   SInt16 theResID
);
```

Parameters

theRect

A pointer to the rectangle, specified in local coordinates of the current graphics port, in which to draw the icon.

You cannot determine which icon from the family specified by `theResID` the function will draw. The function determines, from the size of the specified destination rectangle and the current bit depth of the display device, which icon of a given size to draw from an icon family. For example, if the destination rectangle has the coordinates (100,100,116,116) and the display device is set to 4-bit color, the function draws the icon of type `'ics4'` if that icon is available in the icon family.

If the width or height of a destination rectangle is greater than or equal to 32, the function uses the 32-by-32 pixel icon with the appropriate bit depth for the display device. If the destination rectangle is less than 32 by 32 pixels and greater than 16 pixels wide or 12 pixels high, `PlotIconID` uses the 16-by-16 pixel icon with the appropriate bit depth. If the destination rectangle's height is less than or equal to 12 pixels or its width is less than or equal to 16 pixels, `PlotIconID` uses the 12-by-16 pixel icon with the appropriate bit depth. (Typically only the Finder and Standard File Package use 12-by-16 pixel icons.)

The destination rectangle must be exactly 32 by 32 pixels, 16 by 16 pixels, or 12 by 16 pixels for the function to draw the icon without stretching it. If the destination rectangle is not one of these standard sizes, the function expands or shrinks the icon to fit.

align

Specifies how the function should align the icon within the rectangle. For example, you can specify that it center the icon within the rectangle or align it at one side or the other. The function moves the icon so that the edges of its mask align with the specified side or direction. See "Icon Alignment Constants" (page 41) for a description of the values you can use here.

transform

Specifies how the function should modify the appearance of the icon. See "Icon Transformation Constants" (page 43) for a description of the values you can use here.

theResID

The resource ID of the icon to draw. The icon resource must be of resource type `'ICN#'`, `'ics#'`, `'icl4'`, `'icl8'`, `'ics4'`, or `'ics8'`. In general, you should specify your icon resources as purgeable.

Return Value

A result code. See "Icon Services and Utilities Result Codes" (page 57).

Special Considerations

This function may move or purge memory blocks in the application heap. Your application should not call this function at interrupt time.

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

`Icons.h`

PlotIconMethod

Draws an icon obtained with the aid of your icon getter callback function. (Deprecated in Mac OS X v10.5. Use Icon Services instead.)

```
OSErr PlotIconMethod (
   const Rect *theRect,
   IconAlignmentType align,
   IconTransformType transform,
   IconGetterUPP theMethod,
   void *yourDataPtr
);
```

Parameters

theRect

A pointer to the rectangle in which to draw the icon, specified in local coordinates of the current graphics port.

align

Specifies how to align the icon within the specified rectangle. See "Icon Alignment Constants" (page 41) for a description of the values you can use here.

transform

Specifies how the function should modify the appearance of the icon. See "Icon Transformation Constants" (page 43) for a description of the values you can use here.

theMethod

A universal procedure pointer to your icon getter callback function. `PlotIconMethod` uses your icon getter function to obtain the icon to draw.

`PlotIconMethod` passes to your icon getter function the type of the icon to draw and the value specified in the `yourDataPtr` parameter. The `PlotIconMethod` function examines the current bit depth of the display devices and calls your icon getter function once for each display device that intersects the rectangle specified in the parameter `theRect`. Your icon getter function should return a handle to the requested icon's data. Your icon getter function can get the icon data using whatever method is appropriate to your application. For example, your application might maintain its own cache of icons or use its icon getter function to get an icon from the desktop database.

For more information see the `IconGetterProcPtr` (page 37) callback.

yourDataPtr

A pointer to data that is passed to your icon getter callback function.

Return Value

A result code. See "Icon Services and Utilities Result Codes" (page 57).

Special Considerations

This function may move or purge memory blocks in the application heap. For that reason, your application should not call it at interrupt time.

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

`Icons.h`

PlotIconRef

Draws an icon using appropriate size and depth data from an `IconRef`. (Deprecated in Mac OS X v10.5. Use `PlotIconRefInContext` (page 30) instead.)

```
OSErr PlotIconRef (
   const Rect *theRect,
   IconAlignmentType align,
   IconTransformType transform,
   IconServicesUsageFlags theIconServicesUsageFlags,
   IconRef theIconRef
);
```

Parameters

theRect
A pointer to the rectangle where the icon is to be drawn.

align
A value specifying how Icon Services should align the icon within the rectangle.

transform
A value specifying how Icon Services should modify the appearance of the icon.

theIconServicesUsageFlags
Reserved for future use. Pass the `kIconServicesNormalUsageFlag` constant in this parameter.

theIconRef
The `IconRef` for the icon to draw.

Return Value

A result code. See "Icon Services and Utilities Result Codes" (page 57).

Discussion

This function is similar to the Icon Utilities function `PlotIconSuite`.

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

`Icons.h`

PlotIconSuite

Draws the icon described by an icon suite using the most appropriate icon in the suite for the current bit depth of the display device and the rectangle in which the icon is to be drawn. (Deprecated in Mac OS X v10.5. Use Icon Services instead.)

```
OSErr PlotIconSuite (
   const Rect *theRect,
   IconAlignmentType align,
   IconTransformType transform,
   IconSuiteRef theIconSuite
);
```

Parameters

theRect

A pointer to the rectangle in which to draw the icon.

The function plots a single icon from the icon suite in the current graphics port. You cannot determine which icon from a given suite it will draw; the function bases this decision on the size of the specified destination rectangle and the current bit depth of the display device. For example, if the destination rectangle has the coordinates (100,100,116,116) and the display device is set to 4-bit color, the function draws the icon of type `'ics4'` if that icon is available in the icon suite.

If the width or height of a destination rectangle is greater than or equal to 32 pixels, the function uses the 32-by-32 pixel icon with the appropriate bit depth for the display device. If the destination rectangle is less than 32 by 32 pixels and greater than 16 pixels wide or 12 pixels high, the function uses the 16-by-16 pixel icon with the appropriate bit depth. If the destination rectangle's height is less than or equal to 12 pixels or its width is less than or equal to 16 pixels, the function uses the 12-by-16 pixel icon with the appropriate bit depth. (Typically, only the Finder and Standard File Package use 12-by-16 pixel icons.)

The destination rectangle passed in the `theRect` parameter must be exactly 32 by 32 pixels, 16 by 16 pixels, or 12 by 16 pixels for the function to draw the icon without stretching it. If the destination rectangle is not one of these standard sizes, the function expands or shrinks the icon to fit.

align

Specifies how the function should align the icon within the rectangle. For example, you can specify that the function center the icon within the rectangle or align it at one side or the other. See "Icon Alignment Constants" (page 41) for a description of the values you can use here.

transform

Specifies how the function should modify the appearance of the icon. See "Icon Transformation Constants" (page 43) for a description of the values you can use here.

If you don't specify a label constant in this parameter, the function displays the icon using the default label for that icon suite. When you create an icon suite using the `GetIconSuite` (page 69) function or the `NewIconSuite` (page 79) function, these functions set the default label for the suite to none. To set a new default label for an icon suite, use the `SetSuiteLabel` (page 100) function.

theIconSuite

A handle to the icon suite from which the function gets the icon to draw. You can get a handle to an icon suite using the `GetIconSuite` or `NewIconSuite` functions.

Return Value

A result code. See "Icon Services and Utilities Result Codes" (page 57).

Special Considerations

This function may move or purge memory blocks in the application heap. For that reason, your application should not call it at interrupt time.

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
`Icons.h`

PlotSICNHandle

Draws a small icon of resource type `'SICN'` to which you have a handle. (Deprecated in Mac OS X v10.5. Use Icon Services instead.)

```
OSErr PlotSICNHandle (
   const Rect *theRect,
   IconAlignmentType align,
   IconTransformType transform,
   Handle theSICN
);
```

Parameters

theRect

A pointer to the rectangle in which to draw the icon, specified in local coordinates of the current graphics port.

align

Specifies how the function should align the icon within the rectangle. See "Icon Alignment Constants" (page 41) for a description of the values you can use in this parameter.

transform

Specifies how the function should modify the appearance of the icon. See "Icon Transformation Constants" (page 43) for a description of the values you can use in this parameter.

theSICN

A handle to the icon to draw. You can obtain a handle to the icon using `GetResource` or other Resource Manager functions.

Return Value
A result code. See "Icon Services and Utilities Result Codes" (page 57).

Discussion
Only `'SICN'` resources with a single member—or with two members, the second of which is a mask for the first—plot correctly.

Special Considerations

This function may move or purge memory blocks in the application heap. For that reason, your application should not call it at interrupt time.

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
`Icons.h`

PtInIconID

Determines whether a specified point is within an icon. (Deprecated in Mac OS X v10.5. Use Icon Services instead.)

```
Boolean PtInIconID (
   Point testPt,
   const Rect *iconRect,
   IconAlignmentType align,
   SInt16 iconID
);
```

Parameters

testPt

The point to be tested, specified in local coordinates of the current graphics port. A point is considered to be within an icon if the point is within the icon's mask.

iconRect

A pointer to the rectangle in which the icon appears, specified in local coordinates of the current graphics port. The function determines, from the size of the rectangle specified in this parameter, which icon mask from the given icon family to test the point against. The rectangle which you specify here should be the same rectangle that you last used to draw the icon. The function then uses the location of this rectangle (and the alignment of the icon in the rectangle) to determine whether the specified point is within the icon.

align

Specifies how the icon against which to hit-test is aligned within the rectangle specified by the `iconRect` parameter. The alignment which you specify here should be the same alignment that you last used to draw the icon. See "Icon Alignment Constants" (page 41) for a description of the values you can use in this parameter.

iconID

A resource ID for an icon family. In general, you should specify your icon resources as purgeable.

Return Value

`TRUE` if the point is in the icon and `FALSE` if it is not.

Special Considerations

This function may move or purge memory blocks in the application heap. For that reason, your application should not call it at interrupt time.

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

`Icons.h`

PtInIconMethod

Determines whether a specified point is within an icon obtained with the aid of your icon getter callback function. (Deprecated in Mac OS X v10.5. Use Icon Services instead.)

```
Boolean PtInIconMethod (
   Point testPt,
   const Rect *iconRect,
   IconAlignmentType align,
   IconGetterUPP theMethod,
   void *yourDataPtr
);
```

Parameters

testPt

The point to be tested, specified in local coordinates of the current graphics port. A point is considered to be within an icon if the point is within the icon's mask.

iconRect

A pointer to the rectangle in which the icon appears, specified in local coordinates of the current graphics port. The rectangle which you specify here should be the same rectangle that you last used to draw the icon.

align

Specifies how the icon against which to hit-test is aligned within the rectangle specified by the `iconRect` parameter. The alignment which you specify here should be the same alignment that you last used to draw the icon. See "Icon Alignment Constants" (page 41) for a description of the values you can use in this parameter.

theMethod

A universal procedure pointer to your icon getter callback function. `PtInIconMethod` passes to your icon getter function the type of icon your function should retrieve (either `'ICN#'` or `'ics#'`) and also passes the value specified in the `yourDataPtr` parameter. The `PtInIconMethod` function examines the size of the specified rectangle and requests the appropriate icon from your icon getter function. Your icon getter function should return a handle to the requested icon's data. The `PtInIconMethod` function extracts the mask from the icon data that your icon getter function returns. If your icon getter function returns data that does not correspond to an icon of type `'ICN#'` or type `'ics#'`, `PtInIconMethod` attempts to generate a mask from the returned data.

Your icon getter function can get the icon's data using whatever method is appropriate to your application. For example, your application might maintain its own cache of icons (and pass a pointer to it in the `yourDataPtr` parameter) or use its icon getter function to get an icon from the desktop database.

See the `IconGetterProcPtr` (page 37) callback for more information on creating an icon getter function.

yourDataPtr

A pointer to data that is passed to your icon getter function.

Return Value

`TRUE` if the point is in the icon and `FALSE` if it is not.

Special Considerations

This function may move or purge memory blocks in the application heap. For that reason, your application should not call it at interrupt time.

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
`Icons.h`

PtInIconRef

Tests whether a specified point falls within an icon's mask. (Deprecated in Mac OS X v10.5. Use `IconRefContainsCGPoint` (page 23) instead.)

```
Boolean PtInIconRef (
   const Point *testPt,
   const Rect *iconRect,
   IconAlignmentType align,
   IconServicesUsageFlags theIconServicesUsageFlags,
   IconRef theIconRef
);
```

Parameters

testPt
A pointer to the location, specified in local coordinates of the current graphics port, that Icon Services tests to see whether it falls within the mask of the indicated icon.

iconRect
A pointer to the rectangle defining the area that Icon Services uses to determine which icon is hit-tested. Use the same `Rect` value as when the icon was last drawn.

align
A value that specifies how the indicated icon is aligned within the rectangle specified in the `iconRect` parameter. Use the same `IconAlignmentType` value as when the icon was last drawn. for a description of possible return values, see "Icon Alignment Constants" (page 41).

theIconServicesUsageFlags
Reserved for future use. Pass the `kIconServicesNormalUsageFlag` constant in this parameter.

theIconRef
The icon to be tested.

Return Value
`true` if the point specified in the `testPt` parameter falls within the appropriate icon mask, `false` otherwise.

Discussion
This function is similar to the Icon Utilities function `PtInIconSuite`. The function is useful when you want to determine whether a user has clicked on a particular icon, for example.

Icon Services uses the icon's black-and-white mask for hit-testing, even if you provide a deep mask.

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
`Icons.h`

PtInIconSuite

Determines whether a specified point is within an icon. (Deprecated in Mac OS X v10.5. Use Icon Services instead.)

```
Boolean PtInIconSuite (
   Point testPt,
   const Rect *iconRect,
   IconAlignmentType align,
   IconSuiteRef theIconSuite
);
```

Parameters

testPt

The point to be tested, specified in local coordinates of the current graphics port. A point is considered to be within an icon if the point is within the icon's mask.

iconRect

A pointer to the rectangle in which the icon appears, specified in local coordinates of the current graphics port. The function determines, from the size of the rectangle specified in this parameter, which icon mask (`'ICN#'` or `'ics#'`) from the specified icon suite to test the point against. The function then uses the location of this rectangle (and the location of the icon in the rectangle) to determine whether the given point is within the icon. The rectangle which you specify here should be the same rectangle that you last used to draw the icon.

align

Specifies how the icon against which to hit-test is aligned within the rectangle specified by the `iconRect` parameter. The alignment which you specify here should be the same alignment that you last used to draw the icon. See "Icon Alignment Constants" (page 41) for a description of the values you can use in this parameter.

theIconSuite

A handle to an icon suite.

Return Value

`TRUE` if the point is in the icon and `FALSE` if it is not.

Special Considerations

This function may move or purge memory blocks in the application heap. For that reason, your application should not call it at interrupt time.

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

`Icons.h`

ReadIconFile

Copies data from a given file into an icon family. (Deprecated in Mac OS X v10.5. Use `ReadIconFromFSRef` (page 31) instead.)

```
OSErr ReadIconFile (
   const FSSpec *iconFile,
   IconFamilyHandle *iconFamily
);
```

Parameters

iconFile

A pointer to the file specification structure for the source file for icon data.

iconFamily

A handle to an `iconFamily` data structure to be used as the target data structure. Icon Services resizes the handle as needed. For more information on the `IconFamily` data structure, see 'icns'.

Return Value

A result code. See "Icon Services and Utilities Result Codes" (page 57).

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

`IconsCore.h`

RectInIconID

Hit-tests a rectangle against the appropriate icon mask from an icon family for a specified destination rectangle and alignment. (Deprecated in Mac OS X v10.5. Use Icon Services instead.)

```
Boolean RectInIconID (
   const Rect *testRect,
   const Rect *iconRect,
   IconAlignmentType align,
   SInt16 iconID
);
```

Parameters

testRect

A pointer to the rectangle to be tested, specified in local coordinates of the current graphics port.

iconRect

A pointer to the rectangle in which the icon appears, specified in local coordinates of the current graphics port. The rectangle which you specify here should be the same rectangle that you last used to draw the icon. Like the `PtInIconID` (page 89) function, this function determines, from the size of the rectangle specified in this parameter, which icon mask from the icon family to test the `testRect` parameter against.

align

Specifies how the icon against which to hit-test is aligned within the rectangle specified by `iconRect`. The alignment which you specify here should be the same alignment that you last used to draw the icon. See "Icon Alignment Constants" (page 41) for a description of the values you can use in this parameter.

iconID

A resource ID for an icon family. In general, you should specify your icon resources as purgeable.

Return Value
`TRUE` if the rectangle intersects the icon and `FALSE` if it doesn't.

Special Considerations

This function may move or purge memory blocks in the application heap. For that reason, your application should not call it at interrupt time.

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
`Icons.h`

RectInIconMethod

Hit-tests a rectangle against an icon obtained by your icon getter callback function for a specified destination rectangle and alignment. (Deprecated in Mac OS X v10.5. Use Icon Services instead.)

```
Boolean RectInIconMethod (
   const Rect *testRect,
   const Rect *iconRect,
   IconAlignmentType align,
   IconGetterUPP theMethod,
   void *yourDataPtr
);
```

Parameters

testRect
A pointer to the rectangle to be tested, specified in local coordinates of the current graphics port.

iconRect
A pointer to the rectangle in which the icon appears, specified in local coordinates of the current graphics port. The rectangle which you specify here should be the same rectangle that you last used to draw the icon.

align
Specifies how the icon against which to hit-test is aligned within the rectangle specified by `iconRect`. The alignment which you specify here should be the same alignment that you last used to draw the icon. See "Icon Alignment Constants" (page 41) for a description of the values you can use in this parameter.

theMethod

A universal procedure pointer to your icon getter callback function. `RectInIconMethod` passes to your icon getter function the type of the icon your function should retrieve and the value specified in the `yourDataPtr` parameter. The `RectInIconMethod` function examines the size of the rectangle and requests the appropriate icon from your icon getter function—an icon of icon type `'ICN#'` or `'ics#'`. Your icon getter function should return a handle to the data of the requested icon type. The `RectInIconMethod` function extracts the mask from the icon data that your icon getter function returns. If your icon getter function returns data that does not correspond to an icon of type `'ICN#'` or type `'ics#'`, `RectInIconMethod` attempts to generate a mask from the returned data.

Your icon getter function can get the data for the icon and its mask using whatever method is appropriate to your application. For example, your application might maintain its own cache of icons (and pass a pointer to it in the `yourDataPtr` parameter) or use its icon getter function to get an icon from the desktop database.

See the `IconGetterProcPtr` (page 37) callback for more information on creating an icon getter function.

yourDataPtr

A pointer to data that is passed to your icon getter function.

Return Value

`TRUE` if the rectangle intersects the icon and `FALSE` if it doesn't.

Special Considerations

This function may move or purge memory blocks in the application heap. For that reason, your application should not call it at interrupt time.

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

`Icons.h`

RectInIconRef

Tests whether a specified rectangle falls within an icon's mask. (Deprecated in Mac OS X v10.5. Use `IconRefIntersectsCGRect` (page 24) instead.)

```
Boolean RectInIconRef (
   const Rect *testRect,
   const Rect *iconRect,
   IconAlignmentType align,
   IconServicesUsageFlags iconServicesUsageFlags,
   IconRef theIconRef
);
```

Parameters

testRect

A pointer to the rectangle, specified in local coordinates of the current graphics port, that Icon Services tests to see whether it falls within the mask of the indicated icon.

iconRect

A pointer to the area that Icon Services uses to determine which icon is hit-tested. Use the same `Rect` value as when the icon was last drawn.

align

A value that specifies how the indicated icon is aligned within the rectangle specified in the `iconRect` parameter. Use the same `IconAlignmentType` value as when the icon was last drawn. for a description of possible return values, see "Icon Alignment Constants" (page 41).

iconServicesUsageFlags

Reserved for future use. Pass the `kIconServicesNormalUsageFlag` constant in this parameter.

theIconRef

A pointer to a value of type `IconRef` specifying the icon family to use for drawing the requested icon.

Return Value

`true` if the rectangle specified in the `testRect` parameter intersects the appropriate icon mask, `false` otherwise.

Discussion

This function is similar to the Icon Utilities function `RectInIconSuite`. The function is useful when you want to determine whether a user selection intersects a particular icon, for example.

Icon Services uses the icon's black-and-white mask for hit-testing, even if you provide a deep mask.

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

`Icons.h`

RectInIconSuite

Hit-tests a rectangle against the appropriate icon mask from an icon suite for a specified destination rectangle and alignment. (Deprecated in Mac OS X v10.5. Use Icon Services instead.)

```
Boolean RectInIconSuite (
   const Rect *testRect,
   const Rect *iconRect,
   IconAlignmentType align,
   IconSuiteRef theIconSuite
);
```

Parameters

testRect

A pointer to the rectangle to be tested, specified in local coordinates of the current graphics port.

iconRect

A pointer to the rectangle in which the icon appears, specified in local coordinates of the current graphics port. The rectangle which you specify here should be the same rectangle that you last used to draw the icon. Like the `PtInIconSuite` (page 92) function, this function determines, from the size of the rectangle specified in this parameter, which icon mask from the icon suite specified by `theIconSuite` to test the test rectangle against. For example, if the coordinates of the `iconRect` parameter are (100,100,116,116) and the icon cache contains entries for each icon family member, `RectInIconSuite` uses the icon mask defined by the `'ics#'` entry.

The function then intersects the rectangle specified by `testRect` with the icon mask in the `iconRect` rectangle.

align

Specifies how the icon against which to hit-test is aligned within the rectangle specified by `iconRect`. The alignment which you specify here should be the same alignment that you last used to draw the icon. See "Icon Alignment Constants" (page 41) for a description of the values you can use in this parameter.

theIconSuite

A handle to an icon suite.

Return Value

`TRUE` if the rectangle intersects the icon and `FALSE` if it doesn't.

Special Considerations

This function may move or purge memory blocks in the application heap. For that reason, your application should not call it at interrupt time.

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

`Icons.h`

RegisterIconRefFromIconFile

Adds a file-derived `IconRef` to the Icon Services registry. (Deprecated in Mac OS X v10.5. Use `RegisterIconRefFromFSRef` (page 32) instead.)

```
OSErr RegisterIconRefFromIconFile (
   OSType creator,
   OSType iconType,
   const FSSpec *iconFile,
   IconRef *theIconRef
);
```

Parameters

creator

The creator code of the icon data you wish to register. You can use your application's creator code, for example. Lower-case creator codes are reserved for the system.

iconType

The type code of the icon data you wish to register.

iconFile
A pointer to the file system specification structure for the file to use as the icon data source.

theIconRef
On return, a pointer to the desired icon data.

Return Value
A result code. See "Icon Services and Utilities Result Codes" (page 57).

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
`IconsCore.h`

RegisterIconRefFromResource

Adds a resource-derived `IconRef` to the Icon Services registry. (Deprecated in Mac OS X v10.5. Use `RegisterIconRefFromFSRef` (page 32) instead.)

```
OSErr RegisterIconRefFromResource (
   OSType creator,
   OSType iconType,
   const FSSpec *resourceFile,
   SInt16 resourceID,
   IconRef *theIconRef
);
```

Parameters

creator
The creator code of the icon data you wish to register. You can use your application's creator code, for example. Lower-case creator codes are reserved for the system.

iconType
The type code of the icon data you wish to register.

resourceFile
A pointer to the file system specification structure for the resource file from which to read the icon data.

resourceID
The resource ID of the icon data to be registered. This value must be non-zero.

You should provide a resource of type `'icns'` if possible. If an `'icns'` resource is not available, Icon Services uses standard icon suite resources, such as `'ICN#'`, instead.

theIconRef
On return, a pointer to the desired icon data.

Return Value
A result code. See "Icon Services and Utilities Result Codes" (page 57).

Discussion
You can use the `RegisterIconRefFromResource` function to register icons from `'icns'` resources or "classic" custom icon resources (`'ics#'` , `'ICN#'` , etc.). Icon Services searches `'icns'` resources before searching other icon resources.

Calling this function increments the reference count of the `IconRef`.

Remember to call the function `ReleaseIconRef` (page 33) when you're done with an `IconRef`.

Special Considerations

Before using the recommended replacement function, you need to move the contents of the icon resource into an icon family `.icns` file.

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
`IconsCore.h`

SetIconCacheData

Sets the data associated with an icon cache. (Deprecated in Mac OS X v10.5. Use Icon Services instead.)

```
OSErr SetIconCacheData (
   IconCacheRef theCache,
   void *theData
);
```

Parameters

theCache
A reference to the icon cache whose data is to be set.

theData
A pointer to the data to set.

Return Value
A result code. See "Icon Services and Utilities Result Codes" (page 57).

Discussion
All the Icon Utilities functions that accept a handle to an icon suite also accept a handle to an icon cache.

Special Considerations

This function may move or purge memory blocks in the application heap. For that reason, your application should not call it at interrupt time.

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
`Icons.h`

SetIconCacheProc

Sets the icon getter callback function associated with an icon cache. (Deprecated in Mac OS X v10.5. Use Icon Services instead.)

```
OSErr SetIconCacheProc (
   IconCacheRef theCache,
   IconGetterUPP theProc
);
```

Parameters

theCache

A reference to the icon cache whose icon getter function is to be set.

theProc

A universal procedure pointer to the icon getter callback function to associate with the specified cache. See the `IconGetterProcPtr` (page 37) callback for more information on icon getter functions.

Return Value

A result code. See "Icon Services and Utilities Result Codes" (page 57).

Discussion

All the Icon Utilities functions that accept a handle to an icon suite also accept a handle to an icon cache. An icon cache is like an icon suite except that it also contains a pointer to an icon getter callback function and a pointer to data that can be used as a reference constant. An icon cache typically does not contain handles to the icon resources for all icon family members. Instead, if the icon cache does not contain an entry for a specific type of icon in an icon family, the Icon Utilities functions call your application's icon getter function to retrieve the data for that icon type.

Special Considerations

This function may move or purge memory blocks in the application heap. For that reason, your application should not call it at interrupt time.

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

`Icons.h`

SetSuiteLabel

Specifies the default label associated with an icon suite. (Deprecated in Mac OS X v10.5. Use Icon Services instead.)

```
OSErr SetSuiteLabel (
   IconSuiteRef theSuite,
   SInt16 theLabel
);
```

Parameters

theSuite

A handle to an icon suite.

theLabel

An integer from 1 to 7 that specifies a label for the icon suite, or 0 to set the icon suite's label to none. The default label setting helps to determine which of the label colors shown in the Finder's Label menu is applied to icons of that suite when your application displays them.

You can override the default label setting for a suite by specifying a label in the `transform` parameter of the `PlotIconSuite` (page 86) function. For example, suppose the color currently set for the third label displayed in the Finder's Label menu is red, and the color for the fourth label is green. If you set the default label for a suite using `SetSuiteLabel(theSuite,3)`, then draw an icon from the same suite using `PlotIconSuite` and specifying `kTransformNone` in the `transform` parameter, the label color red is applied to the icon. However, if you specify `kTransformLabel4` in the `transform` parameter of the `PlotIconSuite` function, the label color green is applied to the icon.

Return Value
A result code. See "Icon Services and Utilities Result Codes" (page 57).

Special Considerations

This function may move or purge memory blocks in the application heap. For that reason, your application should not call it at interrupt time.

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
`Icons.h`

WriteIconFile

Copies data from a given icon family into a file. (Deprecated in Mac OS X v10.5. Use the File Manager instead.)

```
OSErr WriteIconFile (
   IconFamilyHandle iconFamily,
   const FSSpec *iconFile
);
```

Parameters

iconFamily

A handle to an `iconFamily` data structure to be used as a source for icon data. For more information on the `IconFamily` data structure, see 'icns'.

iconFile

A pointer to the file specification structure for the file to use as the target for icon data.

Return Value
A result code. See "Icon Services and Utilities Result Codes" (page 57).

Special Considerations

Icon Services is designed to read icon data from a file and cache the data, but not to write out icon data. You can use File Manager functions to write your icon data to a file.

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
`IconsCore.h`

Document Revision History

This table describes the changes to *Icon Services and Utilities Reference*.

Date	Notes
2007-04-06	Updated for Mac OS X v10.5.
	Added information about three new functions: `IconRefToHIShape` (page 25), `IconRefContainsCGPoint` (page 23), and `IconRefIntersectsCGRect` (page 24).
2006-03-08	Added information about deprecated functions.
2005-07-07	Fixed typographical error. Clarified usage of GetIconFamilyData and SetIconFamilyData.
2003-02-19	Added result codes.
	Updated formatting.
	Fixed typographical errors.

Document Revision History

Index

K

L

M

N

O

P

R

S

T

U

W