
Obtaining and Using Icons With Icon Services

[Carbon](#) > [User Experience](#)



2003-02-01



Apple Inc.
© 2003 Apple Computer, Inc.
All rights reserved.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, mechanical, electronic, photocopying, recording, or otherwise, without prior written permission of Apple Inc., with the following exceptions: Any person is hereby authorized to store documentation on a single computer for personal use only and to print copies of documentation for personal use provided that the documentation contains Apple's copyright notice.

The Apple logo is a trademark of Apple Inc.

Use of the "keyboard" Apple logo (Option-Shift-K) for commercial purposes without the prior written consent of Apple may constitute trademark infringement and unfair competition in violation of federal and state laws.

No licenses, express or implied, are granted with respect to any of the technology described in this document. Apple retains all intellectual property rights associated with the technology described in this document. This document is intended to assist application developers to develop applications only for Apple-labeled computers.

Every effort has been made to ensure that the information in this document is accurate. Apple is not responsible for typographical errors.

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

Apple, the Apple logo, Carbon, Mac, Mac OS, Macintosh, and QuickDraw are trademarks of Apple Inc., registered in the United States and other countries.

Finder is a trademark of Apple Inc.

Simultaneously published in the United States and Canada.

Even though Apple has reviewed this document, APPLE MAKES NO WARRANTY OR REPRESENTATION, EITHER EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT, ITS QUALITY, ACCURACY, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. AS A RESULT, THIS DOCUMENT IS PROVIDED "AS IS," AND YOU, THE READER, ARE ASSUMING THE ENTIRE RISK AS TO ITS QUALITY AND ACCURACY.

IN NO EVENT WILL APPLE BE LIABLE FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES RESULTING FROM ANY DEFECT OR INACCURACY IN THIS DOCUMENT, even if advised of the possibility of such damages.

THE WARRANTY AND REMEDIES SET FORTH ABOVE ARE EXCLUSIVE AND IN LIEU OF ALL OTHERS, ORAL OR WRITTEN, EXPRESS OR IMPLIED. No Apple dealer, agent, or employee is authorized to make any modification, extension, or addition to this warranty.

Some states do not allow the exclusion or limitation of implied warranties or liability for incidental or consequential damages, so the above limitation or exclusion may not apply to you. This warranty gives you specific legal rights, and you may also have other rights which vary from state to state.

Contents

Introduction	Introduction to Obtaining and Using Icons With Icon Services 7
---------------------	---

Chapter 1	Icon Services Concepts 9
------------------	---------------------------------

The IconRef	9
Reference Counting	9
The 'icns' Resource	9
32-bit Icon Data	9
Deep Masks	10
Huge Icons	10

Chapter 2	Icon Services Tasks 11
------------------	-------------------------------

Basic Tasks With Icon Services	11
Obtaining and Releasing IconRefs	11
Using IconRefs	11
Advanced Tasks With Icon Services	12
Registering Icon Data	12
Updating IconRefs	12
Overriding and Restoring Icon Data	12
Modifying Reference Counts	13
Flushing IconRefs	13
Using Badges	13
Guidelines for Designing Icons	14

Document Revision History 15

C O N T E N T S

Figures and Listings

Chapter 2 **Icon Services Tasks 11**

Figure 2-1	Folder icons displayed in standard form and with a badge	13
Listing 2-1	Obtaining an IconRef for the standard help icon	11

Introduction to Obtaining and Using Icons With Icon Services

This document starts with an overview of Icon Services and follows with a detailed description of how to use Icon Services. You should read this document if you are interested in using Icon Services to obtain and display icons for your application or extension. This document assumes that you are familiar with the basic concepts of icon creation and usage, as described in Macintosh Human Interface Guidelines and Inside Macintosh.

I N T R O D U C T I O N

Introduction to Obtaining and Using Icons With Icon Services

Icon Services Concepts

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.

The IconRef

The basic data type used by Icon Services is the `IconRef`, a 32-bit opaque value. You obtain an `IconRef` by calling one of the `GetIconRef` functions described in [“Obtaining and Releasing IconRefs”](#) (page 11). Two or more files that have the same file type and creator and do not provide custom icons will use the same `IconRef`. Files with custom icons have their own `IconRef`.

Reference Counting

`IconRef` values are reference counted, so that the icon data represented by a particular `IconRef` can be shared by several clients simultaneously. Each client that uses a particular `IconRef` increments that `IconRef`'s reference count. When there are no more clients using a particular `IconRef`, the icon data associated with it is disposed of.

The 'icns' Resource

The `'icns'` resource is a means of providing a single source for icon data, as opposed to the variety of icon resources represented by `'ICN#'`, `'icl8'` and other familiar resource types. Combining all icon data into a single resource type speeds up icon fetching and simplifies resource management.

32-bit Icon Data

The `'icns'` resource provides for 32-bit-deep icon data.

Deep Masks

Icons provided through the 'icons' resource feature deep masks, meaning an icon mask can have 256 different levels of transparency.

Huge Icons

The 'icons' resource adds “huge” icons, which are 48 pixels by 48 pixels, as well as providing the sizes contained in other icon resources. For more information, see 'icons'.

Icon Services Tasks

This chapter outlines basic and advanced tasks that you can perform using Icon Services.

Basic Tasks With Icon Services

This section describes basic tasks you can perform with Icon Services.

Obtaining and Releasing IconRefs

In order to call Icon Services functions, your application must obtain an `IconRef` for the icon data you want to use. There are three functions you can use to accomplish this task; the one you choose depends on how much information you have about the icon you wish to use. If you need an icon from the desktop database or a registered icon (an icon that has been previously identified to Icon Services), the simplest and fastest way to obtain an `IconRef` is to use the function `GetIconRef`. Icon Services defines a number of constants for non-fileFinder icons, which makes it simple to use the `GetIconRef` function to obtain an `IconRef` for one of these icons. Listing 3-1 shows an example of how to obtain an `IconRef` for the standard Help icon:

Listing 2-1 Obtaining an `IconRef` for the standard help icon

```
err = GetIconRef(kOnSystemDisk, kSystemIconsCreator, kHelpIcon,
               &iconRef);
```

If you need an `IconRef` for a folder that you know has no custom icons associated with it, use the function `GetIconRefFromFolder`. If you need an `IconRef` for a file object for which you don't have any information, use the function `GetIconRefFromFile`.

Using IconRefs

Once you obtain a valid `IconRef`, you can use Icon Services functions to accomplish two major types of tasks. The first type of task is to draw an icon of the appropriate size and type in a specified area. The second task is checking to see whether the user has clicked on or selected an icon (also known as hit-testing). These functions are designed to be similar to familiar Icon Utilities functions.

Drawing Icons

Icon Services provides two basic drawing functions. For the task of drawing an icon directly to the screen, use the function `PlotIconRef`. If you need to convert icon data into a QuickDraw region, use the function `IconRefToRgn`.

Note: The introduction of deep masks means that you cannot simply draw over an icon and assume the previous icon will be erased. Call the function `IconRefToRgn` to determine the area occupied by the current icon, erase that area, then draw the new icon.

Hit-Testing

Icon Services provides several ways to determine whether a user has interacted with an icon. To determine whether a user has clicked on an icon, use the function `PtInIconRef`. If you need to determine whether an icon falls within a given rectangle (like a selection rectangle, for example), use the function `RectInIconRef`. You can also use the function `IconRefToRgn` to do hit-testing.

Advanced Tasks With Icon Services

Icon Services gives you several ways to modify the icon data used by Icon Services. You can add icon data to the Icon Services cache by registering icon data with the functions `RegisterIconRefFromIconFamily` or `RegisterIconRefFromIconFile`. You can also release registered data by using the function `UnregisterIconRef`. You can override existing data in an `IconRef` and replace it with custom data by using the functions `OverrideIconRef` or `OverrideIconRefFromResource`. You can also restore the original data in an `IconRef` by using the function `RemoveIconOverride`.

Registering Icon Data

You can speed up access to frequently-used icon data by registering icons. The preferred way of registering icons is to use the function `RegisterIconRefFromIconFile`. This will make the icon data purgeable if you need to free up memory later. If you can't use the `RegisterIconRefFromResource` function, you can use the function `RegisterIconRefFromIconFamily`. To unregister icon data, use the function `UnregisterIconRef`.

Updating IconRefs

If you need to refresh icon data without releasing an `IconRef`, you can use the function `UpdateIconRef` to obtain current icon data. This might be useful after you have changed a file's custom icon, for example.

Overriding and Restoring Icon Data

You may wish to redraw icons on a temporary basis without going to the trouble of obtaining a new `IconRef`. One example of this is when a user has started to download a file and you want to use partial file icons to represent the various stages of the download process. Icon Services provides two functions to temporarily override the data in an `IconRef`; the one you choose depends on the source of the data you will use for the override. If you obtain the source data from another `IconRef`, use the function `OverrideIconRef`. If the source data is contained in a resource, use the function `OverrideIconRefFromResource`. You can restore the original icon data by using the function `RemoveIconOverride`.

Modifying Reference Counts

You may find it useful to modify the reference count of an `IconRef` in preference to obtaining multiple `IconRefs` to the same data. This might be useful when your application maintains multiple instances of the same icon data, such as multiple file-copying operations. Use the function `AcquireIconRef` to increment the reference count for an `IconRef`. This allows you to keep the icon data in the cache without going through the trouble of obtaining additional `IconRefs`. If you modify the reference count directly, be sure to decrement the reference count as needed by using the function `ReleaseIconRef`.

Flushing IconRefs

When you want to free up memory by releasing purgeable icon data from the cache, the preferred method is to use the function `FlushIconRefs`. You may also use the function `FlushIconRefsByVolume`, but this has two potential problems:

1. The additional scope of the `FlushIconRefsByVolume` function means it will take longer to complete.
2. Using the `FlushIconRefsByVolume` function makes the icon data of all currently used `IconRefs` non-purgeable. This means any subsequent efforts to flush icon data will be much less effective.

Using Badges

A badge is an overlay or replacement for an icon. You can use a badge to signify that a folder contains special files, for example. Badges are described by a `'badg'` resource you store in a file's resource fork or a folder's invisible icon file. Figure 3-1 shows a folder alias icon displayed in standard form and with a badge.

Figure 2-1 Folder icons displayed in standard form and with a badge



There are two steps required to use a custom badge with a file object.

- Clear the `kExtendedFlagsAreInvalid` bit and set the `kExtendedFlagHasCustomBadge` bit in the `extendedFinderFlags` field of the `ExtendedFileInfo` structure (if the object is a file) or the `ExtendedFolderInfo` structure (if the object is a folder).
- Add a resource of type `kCustomBadgeResourceType ('badg')` and ID `kCustomBadgeResourceID` to a file's resource fork or a folder's icon file.

There are three ways to use the data from a `'badg'` resource.

1. The `customBadgeType` and `customBadgeCreator` fields let you designate a custom badge to display on top of another icon, as shown in [Figure 3-1](#) (page 13).

2. The `windowBadgeType` and `windowBadgeCreator` fields let you designate which icon to display in Finder window header of the badged file or folder.
3. The `overrideType` and `overrideCreator` fields let you designate the badge as a replacement for the standard icon for this file or folder.

The type and creator codes specified in a 'badg' resource must be registered with Icon Services before you can use the badge. For more information, see [“Registering Icon Data”](#) (page 12).

If you supply a custom icon resource for a badge, Icon Services will use it in preference to other available data. For a complete description of the badge resource, see 'badg'.

Guidelines for Designing Icons

Here are some guidelines to follow if you choose to design custom icons for use with Icon Services.

- You must provide at least one of the following icon types: 'ICN#', 'ics#', or 'icl#'.
- If you provide a deep mask, all of the non-transparent pixels in the deep mask should correspond to a black pixel in the black-and-white mask. This is important for hit-testing and proper erasing and drawing of the icon.
- If you provide 32-bit icon data, you should also provide an 8-bit version of the icon. This ensures that the icon can be displayed on an 8-bit display without unwanted dithering.
- 8-bit icon data is no longer limited to the 34 colors of the classic icon color palette. All 256 colors from the System palette are available.
- 4-bit icons are still supported. However, the 4-bit display configuration is rarely used and often unsupported, so we recommend that you do not provide 4-bit icon data. If you don't provide 4-bit icon data, Icon Services will use black-and-white icon data instead.

Document Revision History

This table describes the changes to *Obtaining and Using Icons With Icon Services*.

Date	Notes
2003-02-01	Updated structure.
1999-12-01	First version.

REVISION HISTORY

Document Revision History