

---

# NSImage Class Reference

Graphics & Animation: 2D Drawing



2010-08-03



Apple Inc.  
© 2010 Apple 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, Bonjour, Carbon, Cocoa, Finder, iChat, Mac, Mac OS, Quartz, and Safari are trademarks of Apple Inc., registered in the United States and other countries.

MobileMe is a trademark of Apple Inc.

Adobe, Acrobat, and PostScript are trademarks or registered trademarks of Adobe Systems Incorporated in the U.S. and/or other countries.

OpenGL is a registered trademark of Silicon Graphics, 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

## **NSImage Class Reference 9**

---

Overview	9
Tasks	9
Initializing a New NSImage Object	9
Setting the Image Attributes	10
Referring to Images by Name	10
Determining the Supported Image Types	10
Working With Image Representations	11
Hit Testing an Image	11
Setting the Image Representation Selection Criteria	11
Managing the Focus	12
Drawing the Image	12
Working With Alignment Metadata	13
Setting the Image Storage Options	13
Setting the Image Drawing Options	14
Assigning a Delegate	14
Producing TIFF Data for the Image	14
Producing a CGImage from an Image	15
Managing Incremental Loads	15
Image Accessibility	15
Class Methods	15
canInitWithPasteboard:	15
imageFileTypes	16
imageNamed:	16
imagePasteboardTypes	18
imageTypes	18
imageUnfilteredFileTypes	19
imageUnfilteredPasteboardTypes	19
imageUnfilteredTypes	20
Instance Methods	20
accessibilityDescription	20
addRepresentation:	21
addRepresentations:	21
alignmentRect	22
backgroundColor	22
bestRepresentationForRect:context:hints:	23
cacheMode	23
cancelIncrementalLoad	24
CGImageForProposedRect:context:hints:	24
delegate	25
drawAtPoint:fromRect:operation:fraction:	25

drawInRect:fromRect:operation:fraction:	26
drawInRect:fromRect:operation:fraction:respectFlipped:hints:	27
drawRepresentation:inRect:	28
hitTestRect:withImageDestinationRect:context:hints:flipped:	28
initByReferencingFile:	29
initByReferencingURL:	30
initWithCGImage:size:	31
initWithContentsOfFile:	31
initWithContentsOfURL:	32
initWithData:	32
initWithDataIgnoringOrientation:	33
initWithIconRef:	33
initWithPasteboard:	33
initWithSize:	34
isTemplate	35
isValid	35
lockFocus	36
lockFocusFlipped:	37
matchesOnMultipleResolution	37
name	37
prefersColorMatch	38
recache	38
removeRepresentation:	39
representations	39
setAccessibilityDescription:	39
setAlignmentRect:	40
setBackgroundColor:	40
setCacheMode:	41
setDelegate:	41
setMatchesOnMultipleResolution:	42
setName:	42
setPrefersColorMatch:	43
setSize:	43
setTemplate:	44
setUsesEPSOnResolutionMismatch:	45
size	45
TIFFRepresentation	46
TIFFRepresentationUsingCompression:factor:	46
unlockFocus	47
usesEPSOnResolutionMismatch	48
Constants	48
Image Hint Dictionary Keys	48
NSCompositingOperation	49
NSImageLoadStatus	51
NSImageCacheMode	53
Image Template Constants	54

Multiple Documents Drag Image 58  
Sharing Permissions Named Images 58  
System Entity Images 59  
Toolbar Named Images 60  
View Type Template Images 64

**Appendix A      Deprecated NSImage Methods 67**

---

Deprecated in Mac OS X v10.6 67  
bestRepresentationForDevice: 67  
cacheDepthMatchesImageDepth 68  
compositeToPoint:fromRect:operation: 68  
compositeToPoint:fromRect:operation:fraction: 69  
compositeToPoint:operation: 70  
compositeToPoint:operation:fraction: 71  
dissolveToPoint:fraction: 72  
dissolveToPoint:fromRect:fraction: 73  
isCachedSeparately 74  
isDataRetained 74  
isFlipped 75  
lockFocusOnRepresentation: 75  
scalesWhenResized 76  
setCacheDepthMatchesImageDepth: 76  
setCachedSeparately: 77  
setDataRetained: 77  
setFlipped: 78  
setScalesWhenResized: 79

**Document Revision History 81**

---



# Tables

## [NSImage Class Reference](#) 9

---

<a href="#">Table 1</a>	<a href="#">Default pasteboard types for image representations</a>	<a href="#">34</a>
<a href="#">Table 2</a>	<a href="#">Placeholder values for compositing equations</a>	<a href="#">51</a>





# NSImage Class Reference

---

<b>Inherits from</b>	NSObject
<b>Conforms to</b>	NSCoding NSCopying NSPasteboardWriting NSPasteboardReading NSObject (NSObject)
<b>Framework</b>	/System/Library/Frameworks/AppKit.framework
<b>Availability</b>	Available in Mac OS X v10.0 and later.
<b>Companion guide</b>	Cocoa Drawing Guide
<b>Declared in</b>	NSGraphics.h NSImage.h
<b>Related sample code</b>	AnimatedTableView ImageBackground MyPhoto RGB Image Sketch-112

## Overview

An `NSImage` object is a high-level class for manipulating image data. You use this class to load existing images or create new ones and composite them into a view or other image. This class works in conjunction with one or more image representation objects (subclasses of `NSImageRep`), which manage the actual image data.

## Tasks

### Initializing a New `NSImage` Object

- [initByReferencingFile:](#) (page 29)  
Initializes and returns an `NSImage` instance and associates it with the specified file.
- [initByReferencingURL:](#) (page 30)  
Initializes and returns an `NSImage` instance and associates it with the specified URL.

- [initWithCGImage:size:](#) (page 31)  
Initializes and returns an `NSImage` instance with the contents of the `CGImage`.
- [initWithContentsOfFile:](#) (page 31)  
Initializes and returns an `NSImage` instance with the contents of the specified file.
- [initWithContentsOfURL:](#) (page 32)  
Initializes and returns an `NSImage` instance with the contents of the specified URL.
- [initWithData:](#) (page 32)  
Initializes and returns an `NSImage` instance with the contents of the specified `NSData` object.
- [initWithDataIgnoringOrientation:](#) (page 33)  
Initializes and returns an `NSImage` instance with the contents of the specified `NSData` object, ignoring the EXIF orientation tags..
- [initWithPasteboard:](#) (page 33)  
Initializes and returns an `NSImage` instance with data from the specified pasteboard.
- [initWithSize:](#) (page 34)  
Initializes and returns an `NSImage` instance whose size is set to the specified value.
- [initWithIconRef:](#) (page 33)  
Initializes the image object with a Carbon-style icon resource.

## Setting the Image Attributes

- [setSize:](#) (page 43)  
Sets the width and height of the image.
- [size](#) (page 45)  
Returns the size of the receiver.
- [isTemplate](#) (page 35)  
Returns a Boolean value indicating whether the image is a template image.
- [setTemplate:](#) (page 44)  
Sets whether the image represents a template image.

## Referring to Images by Name

- + [imageName:](#) (page 16)  
Returns the `NSImage` instance associated with the specified name.
- [setName:](#) (page 42)  
Registers the receiver under the specified name.
- [name](#) (page 37)  
Returns the name associated with the receiver, if any.

## Determining the Supported Image Types

- + [canInitWithPasteboard:](#) (page 15)  
Tests whether the receiver can create an instance of itself using pasteboard data.

- + [imageTypes](#) (page 18)  
Returns an array of UTI strings identifying the image types supported by the registered `NSImageRep` objects, either directly or through a user-installed filter service.
- + [imageUnfilteredTypes](#) (page 20)  
Returns an array of UTI strings identifying the image types supported directly by the registered `NSImageRep` objects.
- + [imageFileTypes](#) (page 16)  
Returns an array of strings identifying the image types supported by the registered `NSImageRep` objects.
- + [imageUnfilteredFileTypes](#) (page 19)  
Returns an array of strings identifying the file types supported directly by the registered `NSImageRep` objects.
- + [imagePasteboardTypes](#) (page 18)  
Returns an array of strings identifying the pasteboard types supported directly by the registered `NSImageRep` objects.
- + [imageUnfilteredPasteboardTypes](#) (page 19)  
Returns an array of strings identifying the pasteboard types supported directly by the registered `NSImageRep` objects.

## Working With Image Representations

- [addRepresentation:](#) (page 21)  
Adds the specified image representation object to to the receiver.
- [addRepresentations:](#) (page 21)  
Adds an array of image representation objects to the receiver.
- [representations](#) (page 39)  
Returns an array containing all of the receiver's image representations.
- [removeRepresentation:](#) (page 39)  
Removes the specified image representation from the receiver and releases it.
- [bestRepresentationForRect:context:hints:](#) (page 23)  
Returns the best representation of the image for the specified rect using the provided hints.
- [bestRepresentationForDevice:](#) (page 67) **Deprecated in Mac OS X v10.6**  
Returns the best representation for the device with the specified characteristics.

## Hit Testing an Image

- [hitTestRect:withImageDestinationRect:context:hints:flipped:](#) (page 28)  
Returns whether the destination rectangle would intersect a non-transparent portion of the image.

## Setting the Image Representation Selection Criteria

- [setPrefersColorMatch:](#) (page 43)  
Sets whether choosing an image representation favors color matching over resolution matching.

- [prefersColorMatch](#) (page 38)  
Returns a Boolean value indicating whether the image prefers to choose image representations using color matching or resolution matching.
- [setUsesEPSOnResolutionMismatch:](#) (page 45)  
Sets whether EPS image representations are preferred when no other representations match the resolution of the device.
- [usesEPSOnResolutionMismatch](#) (page 48)  
Returns a Boolean value indicating whether EPS representations are preferred when no other representations match the resolution of the device.
- [setMatchesOnMultipleResolution:](#) (page 42)  
Sets whether image representations whose resolutions are integral multiples of the device resolution are considered a match.
- [matchesOnMultipleResolution](#) (page 37)  
Returns a Boolean value indicating whether image representations whose resolution is an integral multiple of the device resolution are considered a match.

## Managing the Focus

- [lockFocus](#) (page 36)  
Prepares the image to receive drawing commands.
- [lockFocusFlipped:](#) (page 37)  
Prepares the image to receive drawing commands using the specified flipped state.
- [unlockFocus](#) (page 47)  
Removes the focus from the receiver.
- [lockFocusOnRepresentation:](#) (page 75) **Deprecated in Mac OS X v10.6**  
Prepares the specified image representation to receive drawing commands. (**Deprecated.** Use the code fragment shown in the special considerations below.)

## Drawing the Image

- [drawAtPoint:fromRect:operation:fraction:](#) (page 25)  
Draws all or part of the image at the specified point in the current coordinate system.
- [drawInRect:fromRect:operation:fraction:](#) (page 26)  
Draws all or part of the image in the specified rectangle in the current coordinate system.
- [drawRepresentation:inRect:](#) (page 28)  
Draws the image using the specified image representation object.
- [drawInRect:fromRect:operation:fraction:respectFlipped:hints:](#) (page 27)  
Draws all or part of the image in the specified rectangle respecting the flippedness and hints.
- [compositeToPoint:fromRect:operation:](#) (page 68) **Deprecated in Mac OS X v10.6**  
Composites a portion of the image to the specified point in the current coordinate system.
- [compositeToPoint:fromRect:operation:fraction:](#) (page 69) **Deprecated in Mac OS X v10.6**  
Composites a portion of the image at the specified opacity to the current coordinate system.
- [compositeToPoint:operation:](#) (page 70) **Deprecated in Mac OS X v10.6**  
Composites the entire image to the specified point in the current coordinate system.

- `compositeToPoint:operation:fraction:` (page 71) **Deprecated in Mac OS X v10.6**  
Composites the entire image at the specified opacity in the current coordinate system.
- `dissolveToPoint:fraction:` (page 72) **Deprecated in Mac OS X v10.6**  
Composites the entire image to the specified location using the `NSCompositeSourceOver` operator.
- `dissolveToPoint:fromRect:fraction:` (page 73) **Deprecated in Mac OS X v10.6**  
Composites a portion of the image to the specified location using the `NSCompositeSourceOver` operator.

## Working With Alignment Metadata

- `alignmentRect` (page 22)  
Returns alignment metadata that your code can use to position the image during layout.
- `setAlignmentRect:` (page 40)  
Sets the alignment metadata that your code can use to position the image during layout.

## Setting the Image Storage Options

- `cacheMode` (page 23)  
Returns the receiver's caching mode.
- `setCacheMode:` (page 41)  
Set the receiver's caching mode.
- `cacheDepthMatchesImageDepth` (page 68) **Deprecated in Mac OS X v10.6**  
Returns a Boolean value indicating whether an image's offscreen window caches use the same bit depth as the image data itself. (**Deprecated.** `NSImage` no longer caches to windows. A cache is now generated appropriate for the destination where an image is drawn. There is no replacement method.)
- `isCachedSeparately` (page 74) **Deprecated in Mac OS X v10.6**  
Returns a Boolean value indicating whether each image representation caches its contents in a separate offscreen window. (**Deprecated.** `NSImage` no longer caches to windows. There is no replacement method.)
- `isDataRetained` (page 74) **Deprecated in Mac OS X v10.6**  
Returns a Boolean value indicating whether the receiver retains its source image data. (**Deprecated.** In Mac OS v10.6, `NSImage` no longer discards data in such a way that the original can no longer be reconstructed. There is no replacement method.)
- `setCacheDepthMatchesImageDepth:` (page 76) **Deprecated in Mac OS X v10.6**  
Sets whether the receiver's offscreen window caches use the same bit depth as the image data itself. (**Deprecated.** `NSImage` no longer caches to windows. A cache is now generated appropriate for the destination where an image is drawn. There is no replacement method.)
- `setCachedSeparately:` (page 77) **Deprecated in Mac OS X v10.6**  
Sets whether each image representation uses a separate offscreen window to cache its contents. (**Deprecated.** `NSImage` no longer caches to windows. There is no replacement method.)
- `setDataRetained:` (page 77) **Deprecated in Mac OS X v10.6**  
Sets whether the receiver retains its source image data. (**Deprecated.** In Mac OS v10.6, `NSImage` no longer discards data in such a way that the original can no longer be reconstructed. There is no replacement method.)

## Setting the Image Drawing Options

- [isValid](#) (page 35)  
Returns a Boolean value indicating whether an image representation from the receiver can be drawn.
- [setBackground-color:](#) (page 40)  
Sets the background color of the image.
- [background-color](#) (page 22)  
Returns the background color of image.
- [recache](#) (page 38)  
Invalidates and frees the offscreen caches of all image representations.
- [isFlipped](#) (page 75) **Deprecated in Mac OS X v10.6**  
Returns a Boolean value indicating whether the image uses a flipped coordinate system. (**Deprecated.** The flipped property of an image was widely misunderstood and has been deprecated. Use [drawInRect:fromRect:operation:fraction:respectFlipped:hints:](#) (page 27) to draw respecting a context's flipped status and [lockFocusFlipped:](#) (page 37) to draw into a flipped image.)
- [scalesWhenResized](#) (page 76) **Deprecated in Mac OS X v10.6**  
Returns a Boolean value indicating whether image representations are scaled to fit the receiver's size. (**Deprecated.** This method was related to caching behavior. In Mac OS X v10.6 and later image caching is no longer necessary and as a result there is no replacement necessary.)
- [setFlipped:](#) (page 78) **Deprecated in Mac OS X v10.6**  
Sets whether the polarity of the y axis is inverted when drawing an image. (**Deprecated.** The flipped property of an image was widely misunderstood and has been deprecated. Use [drawInRect:fromRect:operation:fraction:respectFlipped:hints:](#) (page 27) to draw respecting a context's flipped status and [lockFocusFlipped:](#) (page 37) to draw into a flipped image.)
- [setScaleWhenResized:](#) (page 79) **Deprecated in Mac OS X v10.6**  
Sets whether different-sized image representations are scaled to fit the receiver's size. (**Deprecated.** This method was related to caching behavior. In Mac OS X v10.6 and later image caching is no longer necessary and as a result there is no replacement necessary.)

## Assigning a Delegate

- [setDelegate:](#) (page 41)  
Sets the delegate object of the receiver.
- [delegate](#) (page 25)  
Returns the delegate object of the receiver

## Producing TIFF Data for the Image

- [TIFFRepresentation](#) (page 46)  
Returns a data object containing TIFF data for all of the image representations in the receiver.
- [TIFFRepresentationUsingCompression:factor:](#) (page 46)  
Returns a data object containing TIFF data with the specified compression settings for all of the image representations in the receiver.

## Producing a CGImage from an Image

- [CGImageForProposedRect:context:hints:](#) (page 24)  
Returns a CGImage capturing the drawing of the receiver.

## Managing Incremental Loads

- [cancelIncrementalLoad](#) (page 24)  
Cancels the current download operation immediately, if the image is being incrementally loaded.

## Image Accessibility

- [accessibilityDescription](#) (page 20)  
Returns the image's accessibility description.
- [setAccessibilityDescription:](#) (page 39)  
Sets the image's accessibility description.

## Class Methods

### canInitWithPasteboard:

Tests whether the receiver can create an instance of itself using pasteboard data.

```
+ (BOOL)canInitWithPasteboard:(NSPasteboard *)pasteboard
```

#### Parameters

*pasteboard*

The pasteboard containing the image data.

#### Return Value

YES if the receiver knows how to handle the data on the pasteboard; otherwise, NO.

#### Discussion

This method uses the NSImageRep class method `imageUnfilteredPasteboardTypes` to find a class that can handle the data in the specified pasteboard. If you create your own NSImageRep subclasses, override the `imageUnfilteredPasteboardTypes` method to notify NSImage of the pasteboard types your class supports.

#### Availability

Available in Mac OS X v10.0 and later.

#### See Also

+ [imagePasteboardTypes](#) (page 18)

#### Related Sample Code

CocoaDragAndDrop

People

**Declared In**

NSImage.h

**imageFileTypes**

Returns an array of strings identifying the image types supported by the registered `NSImageRep` objects.

```
+ (NSArray *)imageFileTypes
```

**Return Value**

An array of `NSString` objects, each of which identifies a single supported file type. The array can include encoded HFS file types as well as filename extensions.

**Discussion**

This list includes all file types supported by registered subclasses of `NSImageRep` plus those that can be converted to a supported type by a user-installed filter service. You can pass the array returned by this method directly to the `runModalForTypes:` method of `NSOpenPanel`.

When creating a subclass of `NSImageRep`, do not override this method. Instead, override the `imageUnfilteredFileTypes` method to notify `NSImage` of the file types your class supports directly.

**Availability**

Available in Mac OS X v10.0 and later.

**See Also**

+ [imageUnfilteredFileTypes](#) (page 19)

**Related Sample Code**

CocoaSlides

DeskPictAppDockMenu

Quartz Composer SlideShow

TrackBall

**Declared In**

NSImage.h

**imageNamed:**

Returns the `NSImage` instance associated with the specified name.

```
+ (id)imageNamed:(NSString *)name
```

**Parameters**

*name*

The name associated with the desired image.

**Return Value**

The `NSImage` object associated with the specified name, or `nil` if no such image was found.

**Discussion**

This method searches for named images in several places, returning the first image it finds matching the given name. The order of the search is as follows:



1. Search for an object whose name was set explicitly using the `setName:` method and currently resides in the image cache.
2. Search the application's main bundle for a file whose name matches the specified string. (For information on how the bundle is searched, see *“Accessing a Bundle's Contents”* in *Bundle Programming Guide*.)
3. Search the Application Kit framework for a shared image with the specified name.

When looking for files in the application bundle, it is better (but not required) to include the filename extension in the `name` parameter. When naming an image with the `setName:` method, it is also convention not to include filename extensions in the names you specify. That way, you can easily distinguish between images you have named explicitly and those you want to load from the application's bundle.

One particularly useful image you can retrieve is your application's icon. This image is set by Cocoa automatically and referenced by the string `@NSApplicationIcon`. Icons for other applications can be obtained through the use of methods declared in the `NSWorkspace` class. You can also retrieve many of the standard system images using Cocoa defined constants; for more information, see the *“Image Template Constants”* (page 54), *“Sharing Permissions Named Images”* (page 58), *“System Entity Images”* (page 59), *“Toolbar Named Images”* (page 60), and *“View Type Template Images”* (page 64) sections for applicable constants.

If an application is linked in Mac OS X v10.5 or later, images requested using this method and whose name ends in the word *“Template”* are automatically marked as template images.

The `NSImage` class may cache a reference to the returned image object for performance in some cases. However, the class holds onto cached objects only while the object exists. If the image object is subsequently released, either because its retain count was 0 or it was not referenced anywhere in a garbage-collected application, the object may be quietly removed from the cache. Thus, if you plan to hold onto a returned image object, you must retain it like you would any Cocoa object. You can clear an image object from the cache explicitly by calling the object's `setName:` method and passing `nil` for the image name.

#### Availability

Available in Mac OS X v10.0 and later.

#### See Also

- [setName:](#) (page 42)

- [name](#) (page 37)

[iconForFile:](#) (`NSWorkspace`)

+ [imageFileTypes](#) (page 16)

#### Related Sample Code

[EnhancedDataBurn](#)

[FunHouse](#)

[IconCollection](#)

[MenuMadness](#)

[People](#)

#### Declared In

`NSImage.h`

## imagePasteboardTypes

Returns an array of strings identifying the pasteboard types supported directly by the registered `NSImageRep` objects.

```
+ (NSArray *)imagePasteboardTypes
```

### Return Value

An array of `NSString` objects, each of which identifies a single supported pasteboard type. By default, this list contains the `NSPDFPboardType`, `NSPICTPboardType`, `NSPostScriptPboardType`, and `NSTIFFPboardType` types.

### Discussion

This list includes all pasteboard types supported by registered subclasses of `NSImageRep` plus those that can be converted to a supported type by a user-installed filter service.

When creating a subclass of `NSImageRep`, do not override this method. Instead, override the `imageUnfilteredPasteboardTypes` method to notify `NSImage` of the pasteboard types your class supports.

### Availability

Available in Mac OS X v10.0 and later.

### See Also

+ [imageUnfilteredPasteboardTypes](#) (page 19)

### Related Sample Code

CocoaDragAndDrop  
GeekGameBoard  
GLUT  
Sketch+Accessibility  
Sketch-112

### Declared In

`NSImage.h`

## imageTypes

Returns an array of UTI strings identifying the image types supported by the registered `NSImageRep` objects, either directly or through a user-installed filter service.

```
+ (NSArray *)imageTypes
```

### Return Value

An array of `NSString` objects, each of which contains a UTI identifying a supported image type. Some sample image-related UTI strings include `"public.image"`, `"public.jpeg"`, and `"public.tiff"`. For a list of supported types, see `UTCoreTypes.h`.

### Discussion

The returned list includes UTIs all file types supported by registered subclasses of `NSImageRep` plus those that can be converted to a supported type by a user-installed filter service. You can use the returned UTI strings with any method that supports UTIs.

You should not override this method directly. Instead, you should override the `imageTypes` method of `NSImageRep`.

**Availability**

Available in Mac OS X v10.5 and later.

**See Also**

+ [imageUnfilteredTypes](#) (page 20)

**Related Sample Code**

AnimatedTableView

LightTable

**Declared In**

NSImage.h

**imageUnfilteredFileTypes**

Returns an array of strings identifying the file types supported directly by the registered NSImageRep objects.

```
+ (NSArray *)imageUnfilteredFileTypes
```

**Return Value**

An array of NSString objects, each of which identifies a single supported file type. File types are identified by file extension and HFS file types.

**Discussion**

The returned list does not contain pasteboard types that are available only through a user-installed filter service.

**Availability**

Available in Mac OS X v10.0 and later.

**See Also**

+ [imageFileTypes](#) (page 16)

**Declared In**

NSImage.h

**imageUnfilteredPasteboardTypes**

Returns an array of strings identifying the pasteboard types supported directly by the registered NSImageRep objects.

```
+ (NSArray *)imageUnfilteredPasteboardTypes
```

**Return Value**

An array of NSString objects, each of which identifies a single supported pasteboard type.

**Discussion**

The returned list does not contain pasteboard types that are supported only through a user-installed filter service.

**Availability**

Available in Mac OS X v10.0 and later.

**See Also**

+ [imagePasteboardTypes](#) (page 18)

**Related Sample Code**

GeekGameBoard

**Declared In**

NSImage.h

**imageUnfilteredTypes**

Returns an array of UTI strings identifying the image types supported directly by the registered NSImageRep objects.

```
+ (NSArray *)imageUnfilteredTypes
```

**Return Value**

An array of NSString objects, each of which contains a UTI identifying a supported image type. Some sample image-related UTI strings include "public.image", "public.jpeg", and "public.tiff". For a list of supported types, see UTCoreTypes.h.

**Discussion**

The returned list includes UTI strings only for those file types that are supported directly by registered subclasses of NSImageRep. It does not include types that are supported through user-installed filter services. You can use the returned UTI strings with any method that supports UTIs.

You should not override this method directly. Instead, you should override the `imageUnfilteredTypes` method of NSImageRep.

**Availability**

Available in Mac OS X v10.5 and later.

**See Also**

+ [imageTypes](#) (page 18)

**Declared In**

NSImage.h

## Instance Methods

**accessibilityDescription**

Returns the image's accessibility description.

```
- (NSString *)accessibilityDescription
```

**Return Value**

A short localized string that does not include the name of the interface element.

**Availability**

Available in Mac OS X v10.6 and later.

**See Also**

- [setAccessibilityDescription:](#) (page 39)

**Declared In**

NSImage.h

**addRepresentation:**

Adds the specified image representation object to to the receiver.

```
- (void)addRepresentation:(NSImageRep *) imageRep
```

**Parameters**

*imageRep*

The image representation to add.

**Discussion**

After invoking this method, you may need to explicitly set features of the new image representation, such as the size, number of colors, and so on. This is true particularly when the `NSImage` object has multiple image representations to choose from. See `NSImageRep` and its subclasses for the methods you use to complete initialization.

Any representation added by this method is retained by the receiver. Image representations cannot be shared among multiple `NSImage` objects.

**Availability**

Available in Mac OS X v10.0 and later.

**See Also**

- [representations](#) (page 39)  
- [removeRepresentation:](#) (page 39)

**Related Sample Code**

ColorMatching

CompositeLab

FunHouse

Reducer

Son of Grab

**Declared In**

NSImage.h

**addRepresentations:**

Adds an array of image representation objects to the receiver.

```
- (void)addRepresentations:(NSArray *) imageReps
```

**Parameters**

*imageReps*

An array of `NSImageRep` objects.

**Discussion**

After invoking this method, you may need to explicitly set features of the new image representations, such as their size, number of colors, and so on. This is true particularly when the `NSImage` object has multiple image representations to choose from. See `NSImageRep` and its subclasses for the methods you use to complete initialization.

Representations added by this method are retained by the receiver. Image representations cannot be shared among multiple `NSImage` objects.

**Availability**

Available in Mac OS X v10.0 and later.

**See Also**

- [representations](#) (page 39)
- [removeRepresentation:](#) (page 39)

**Declared In**

`NSImage.h`

**alignmentRect**

Returns alignment metadata that your code can use to position the image during layout.

- `(NSRect)alignmentRect`

**Return Value**

A rectangle containing the layout information for the image. If not set, the returned rectangle has an origin of (0, 0) and a size that matches the size of the image.

**Discussion**

The returned rectangle is merely a hint that your own code can use to determine positioning. The `NSImage` class does not use this rectangle during drawing. However, instances of `NSCell` typically use this information when laying out images within their own boundaries.

**Availability**

Available in Mac OS X v10.5 and later.

**See Also**

- [setAlignmentRect:](#) (page 40)

**Declared In**

`NSImage.h`

**backgroundColor**

Returns the background color of image.

- `(NSColor *)backgroundColor`

**Return Value**

The background color of the image. The default color is transparent, as returned by the `clearColor` method of `NSColor`.

**Discussion**

The background color is visible only if the drawn image representation does not completely cover all of the pixels available for the image's current size.

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

NSImage.h

**bestRepresentationForRect:context:hints:**

Returns the best representation of the image for the specified rect using the provided hints.

```
- (NSImageRep *)bestRepresentationForRect:(NSRect)rect context:(NSGraphicsContext *)referenceContext hints:(NSDictionary *)hints
```

**Parameters**

*rect*

The area of the image to return.

*referenceContext*

A graphics context. This value can be `nil`.

*hints*

An optional dictionary of hints that provide more context for selecting or generating a `CGImage`, and may override properties of the *referenceContext*. See [“Image Hint Dictionary Keys”](#) (page 48) for a summary of the possible key-value pairs.

**Return Value**

The image representation that most closely matches the specified criteria.

**Availability**

Available in Mac OS X v10.6 and later.

**Declared In**

NSImage.h

**cacheMode**

Returns the receiver's caching mode.

```
- (NSImageCacheMode)cacheMode
```

**Return Value**

A value indicating the caching mode. For a list of possible values, see [NSImageCacheMode](#) (page 53). This value is set to `NSImageCacheDefault` by default.

**Availability**

Available in Mac OS X v10.2 and later.

**See Also**

- [setCacheMode:](#) (page 41)

**Declared In**

NSImage.h

**cancelIncrementalLoad**

Cancels the current download operation immediately, if the image is being incrementally loaded.

```
- (void)cancelIncrementalLoad
```

**Discussion**

This call has no effect if the image is not loading.

**Availability**

Available in Mac OS X v10.2 and later.

**Declared In**

NSImage.h

**CGImageForProposedRect:context:hints:**

Returns a `CGImage` capturing the drawing of the receiver.

```
- (CGImageRef)CGImageForProposedRect:(NSRect *)proposedDestRect
    context:(NSGraphicsContext *)referenceContext hints:(NSDictionary *)hints
```

**Parameters**

*proposedDestRect*

On input, the proposed destination rectangle for drawing the image. If `NULL`, it defaults to the smallest pixel-integral rectangle containing `{{0,0}, [self size]}`. The *proposedDestRect* is in user space in the reference context.

*referenceContext*

A graphics context.

*hints*

A dictionary of hints that provide more context for selecting or generating a `CGImage`, and may override properties of the *referenceContext*.

**Return Value**

A `CGImageRef`. This may be an existing `CGImage` if one is available. If not, a new `CGImage` is created.

**Discussion**

An `NSImage` is potentially resolution independent, and may have representations that allow it to draw well in many contexts. A `CGImage` is more like a single pixel-based representation. This method produces a snapshot of how the `NSImage` would draw if it was asked to draw in the proposed rectangle in the graphics context.

All input parameters are optional. They provide hints for how to choose among existing `CGImage`s, or how to create one if there isn't already a `CGImage` available. The parameters are only hints.

This method is typically called, not overridden.

**Availability**

Available in Mac OS X v10.6 and later.



**Declared In**

NSImage.h

**delegate**

Returns the delegate object of the receiver

- (id &lt; NSImageDelegate &gt;)delegate

**Return Value**

The current delegate object, or nil if no delegate has been set.

**Availability**

Available in Mac OS X v10.0 and later.

**See Also**- [setDelegate:](#) (page 41)**Declared In**

NSImage.h

**drawAtPoint:fromRect:operation:fraction:**

Draws all or part of the image at the specified point in the current coordinate system.

- (void)drawAtPoint:(NSPoint)*point* fromRect:(NSRect)*srcRect*  
operation:(NSCompositingOperation)*op* fraction:(CGFloat)*delta***Parameters***point*

The location in the current coordinate system at which to draw the image.

*srcRect*The source rectangle specifying the portion of the image you want to draw. The coordinates of this rectangle are specified in the image's own coordinate system. If you pass in `NSZeroRect`, the entire image is drawn.*op*The compositing operation to use when drawing the image. See the [NSCompositingOperation](#) (page 49) constants.*delta*

The opacity of the image, specified as a value from 0.0 to 1.0. Specifying a value of 0.0 draws the image as fully transparent while a value of 1.0 draws the image as fully opaque. Values greater than 1.0 are interpreted as 1.0.

**Discussion**

The image content is drawn at its current resolution and is not scaled unless the CTM of the current coordinate system itself contains a scaling factor. The image is otherwise positioned and oriented using the current coordinate system.

Unlike the [compositeToPoint:fromRect:operation:](#) (page 68) and [compositeToPoint:fromRect:operation:fraction:](#) (page 69) methods, this method checks the rectangle you pass to the *srcRect* parameter and makes sure it does not lie outside the image bounds.

**Availability**

Available in Mac OS X v10.0 and later.

**See Also**

- [dissolveToPoint:fraction:](#) (page 72)
- [drawInRect:fromRect:operation:fraction:](#) (page 26)

**Related Sample Code**

ComplexBrowser

Reducer

**Declared In**

NSImage.h

**drawInRect:fromRect:operation:fraction:**

Draws all or part of the image in the specified rectangle in the current coordinate system.

```
- (void)drawInRect:(NSRect)dstRect fromRect:(NSRect)srcRect
  operation:(NSCompositingOperation)op fraction:(CGFloat)delta
```

**Parameters**

*dstRect*

The rectangle in which to draw the image, specified in the current coordinate system.

*srcRect*

The source rectangle specifying the portion of the image you want to draw. The coordinates of this rectangle must be specified using the image's own coordinate system. If you pass in `NSZeroRect`, the entire image is drawn.

*op*

The compositing operation to use when drawing the image. See the [NSCompositingOperation](#) (page 49) constants.

*delta*

The opacity of the image, specified as a value from 0.0 to 1.0. Specifying a value of 0.0 draws the image as fully transparent while a value of 1.0 draws the image as fully opaque. Values greater than 1.0 are interpreted as 1.0.

**Discussion**

If the `srcRect` and `dstRect` rectangles have different sizes, the source portion of the image is scaled to fit the specified destination rectangle. The image is otherwise positioned and oriented using the current coordinate system.

Unlike the [compositeToPoint:fromRect:operation:](#) (page 68) and [compositeToPoint:fromRect:operation:fraction:](#) (page 69) methods, this method checks the rectangle you pass to the `srcRect` parameter and makes sure it does not lie outside the image bounds.

**Availability**

Available in Mac OS X v10.0 and later.

**See Also**

- [dissolveToPoint:fraction:](#) (page 72)
- [drawAtPoint:fromRect:operation:fraction:](#) (page 25)

**Related Sample Code**

AnimatedTableView  
 CocoaVideoFrameToNSImage  
 PhotoSearch  
 Transformed Image  
 WebKitDOMElementPlugIn

**Declared In**

NSImage.h

**drawInRect:fromRect:operation:fraction:respectFlipped:hints:**

Draws all or part of the image in the specified rectangle respecting the flippedness and hints.

```
- (void)drawInRect:(NSRect)dstSpacePortionRect fromRect:(NSRect)srcSpacePortionRect
  operation:(NSCompositingOperation)op fraction:(CGFloat)requestedAlpha
  respectFlipped:(BOOL)respectContextIsFlipped hints:(NSDictionary *)hints
```

**Parameters**

*dstSpacePortionRect*

The rectangle in which to draw the image, specified in the current coordinate system.

*srcSpacePortionRect*

The source rectangle specifying the portion of the image you want to draw. The coordinates of this rectangle must be specified using the image's own coordinate system. If you pass in `NSZeroRect`, the entire image is drawn.

*op*

The compositing operation to use when drawing the image. See the [NSCompositingOperation](#) (page 49) constants.

*requestedAlpha*

The alpha of the image, specified as a value from 0.0 to 1.0. Specifying a value of 0.0 draws the image as fully transparent while a value of 1.0 draws the image as fully opaque. Values greater than 1.0 are interpreted as 1.0.

*respectContextIsFlipped*

YES if the drawing should respect the context flipped state, otherwise NO.

*hints*

An optional dictionary of hints that provide more context for selecting or generating the image. See [“Image Hint Dictionary Keys”](#) (page 48) for a summary of the possible key-value pairs.

**Discussion**

If the *srcSpacePortionRect* and *dstSpacePortionRect* rectangles have different sizes, the source portion of the image is scaled to fit the specified destination rectangle.

**Availability**

Available in Mac OS X v10.6 and later.

**Declared In**

NSImage.h

## drawRepresentation:inRect:

Draws the image using the specified image representation object.

```
- (BOOL)drawRepresentation:(NSImageRep *)imageRep inRect:(NSRect)dstRect
```

### Parameters

*imageRep*

The image representation object to be drawn.

*dstRect*

The rectangle in which to draw the image representation, specified in the current coordinate system.

### Return Value

YES if the image was successfully drawn; otherwise, returns NO.

### Discussion

This method fills the specified rectangle with the image's current background color and then sends a message to the specified image representation asking if to draw itself. If the image supports the ability to scale itself when it is resized, this method sends a `drawInRect:` message; otherwise, it sends a `drawAtPoint:` message.

You should not call this method directly; an `NSImage` object uses it to cache and print its image representations. You can override this method to change the way images are rendered into their caches and onto the printed page. For example, you could scale or rotate the coordinate system before sending this message to `super` to continue rendering the image representation.

If the background color is fully transparent and the image data is not being cached, the specified rectangle is not to be filled before the representation draws.

### Availability

Available in Mac OS X v10.0 and later.

### See Also

- [backgroundColor](#) (page 22)

### Declared In

`NSImage.h`

## hitTestRect:withImageDestinationRect:context:hints:flipped:

Returns whether the destination rectangle would intersect a non-transparent portion of the image.

```
- (BOOL)hitTestRect:(NSRect)testRectDestSpace withImageDestinationRect:(NSRect)imageRectDestSpace context:(NSGraphicsContext *)referenceContext hints:(NSDictionary *)hints flipped:(BOOL)flipped
```

### Parameters

*testRectDestSpace*

The rectangle to hit test.

*imageRectDestSpace*

A rectangle representing the drawn size of the image.

*referenceContext*

A graphics context. This value can be `nil`.

*hints*

An optional dictionary of hints that provide more context for selecting or generating a `CGImage`, and may override properties of the `referenceContext`. See [“Image Hint Dictionary Keys”](#) (page 48) for a summary of the possible key-value pairs.

*flipped*

YES if the image is flipped, otherwise NO.

**Return Value**

YES if the `testRectDestSpace` intersects with non-transparent content within the `imageRectDestSpace`, otherwise NO.

**Discussion**

This method simulates the results of hit-testing the test rectangle as if the image was drawn in the graphics context using the provided hints and respecting the specified flippedness..

**Availability**

Available in Mac OS X v10.6 and later.

**Declared In**

NSImage.h

**initWithReferencingFile:**

Initializes and returns an `NSImage` instance and associates it with the specified file.

```
- (id)initWithReferencingFile:(NSString *)filename
```

**Parameters***filename*

A full or relative path name specifying the file with the desired image data. Relative paths must be relative to the current working directory.

**Return Value**

An initialized `NSImage` instance, or `nil` if the new instance cannot be initialized.

**Discussion**

This method initializes the image object lazily. It does not actually open the specified file or create any image representations from its data until an application attempts to draw the image or request information about it.

The *filename* parameter should include the file extension that identifies the type of the image data. The mechanism that actually creates the image representation for *filename* looks for an `NSImageRep` subclass that handles that data type from among those registered with `NSImage`.

Because this method doesn't actually create image representations for the image data, your application should do error checking before attempting to use the image; one way to do so is by invoking the [`isValid`](#) (page 35) method to check whether the image can be drawn.

This method invokes [`setDataRetained:`](#) (page 77) with an argument of YES, thus enabling it to hold onto its filename. When archiving an image created with this method, only the image's filename is written to the archive.

If the cached version of the image uses less memory than the original image data, the original data is flushed and the cached image is used. (This can occur for images whose resolution is greater than 72 dpi.) If you resize the image by less than 50%, the data is loaded in again from the file. If you expect the file to change or be deleted, you should use [initWithContentsOfFile:](#) (page 31) instead.

#### Availability

Available in Mac OS X v10.0 and later.

#### Related Sample Code

CompositeLab

FilterDemo

ImagePicker

PictureTaker

Rulers

#### Declared In

NSImage.h

## initWithReferencingURL:

Initializes and returns an `NSImage` instance and associates it with the specified URL.

```
- (id)initWithReferencingURL:(NSURL *)url
```

#### Parameters

*url*

The URL identifying the image.

#### Return Value

An initialized `NSImage` instance, or `nil` if the new instance cannot be initialized.

#### Discussion

This method initializes the image object lazily. It does not attempt to retrieve the data from the specified URL or create any image representations from that data until an application attempts to draw the image or request information about it.

This *url* parameter should include a file extension that identifies the type of the image data. The mechanism that actually creates the image representation looks for an `NSImageRep` subclass that handles that data type from among those registered with `NSImage`.

Because this method doesn't actually create image representations for the image data, your application should do error checking before attempting to use the image; one way to do so is by invoking the [isValid](#) (page 35) method to check whether the image can be drawn.

This method invokes [setDataRetained:](#) (page 77) with an argument of `YES`, thus enabling it to hold onto its URL. When archiving an image created with this method, only the image's URL is written to the archive.

#### Availability

Available in Mac OS X v10.2 and later.

#### Related Sample Code

AnimatedTableView

TrackBall

**Declared In**

NSImage.h

**initWithCGImage:size:**

Initializes and returns an `NSImage` instance with the contents of the `CGImage`.

```
- (id)initWithCGImage:(CGImageRef)cgImage size:(NSSize)size
```

**Parameters***cgImage*

The source `CGImage`.

*size*

The size of the new image. If `size` is `NSZeroSize`, the pixel dimensions of *cgImage* are assumed as the image's size.

**Return Value**

An initialized `NSImage` instance, or `nil` if the new instance cannot be initialized.

**Discussion**

You should not assume anything about the image, other than that drawing it is equivalent to drawing the `CGImage`.

This is not a designated initializer.

**Availability**

Available in Mac OS X v10.6 and later.

**Related Sample Code**

CameraBrowser

QuickLookDownloader

SimpleCameraBrowser

**Declared In**

NSImage.h

**initWithContentsOfFile:**

Initializes and returns an `NSImage` instance with the contents of the specified file.

```
- (id)initWithContentsOfFile:(NSString *)filename
```

**Parameters***filename*

A full or relative path name specifying the file with the desired image data. Relative paths must be relative to the current working directory.

**Return Value**

An initialized `NSImage` instance, or `nil` if the method cannot create an image representation from the contents of the specified file.

**Discussion**

Unlike [initWithReferencingFile:](#) (page 29), which initializes an `NSImage` object lazily, this method immediately opens the specified file and creates one or more image representations from its data.

The *filename* parameter should include the file extension that identifies the type of the image data. This method looks for an `NSImageRep` subclass that handles that data type from among those registered with `NSImage`.

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

`NSImage.h`

**initWithContentsOfURL:**

Initializes and returns an `NSImage` instance with the contents of the specified URL.

```
- (id)initWithContentsOfURL:(NSURL *)aURL
```

**Parameters**

*aUrl*

The URL identifying the image.

**Return Value**

An initialized `NSImage` instance, or `nil` if the method cannot create an image representation from the contents of the specified URL.

**Availability**

Available in Mac OS X v10.0 and later.

**Related Sample Code**

[CocoaCreateMovie](#)

**Declared In**

`NSImage.h`

**initWithData:**

Initializes and returns an `NSImage` instance with the contents of the specified `NSData` object.

```
- (id)initWithData:(NSData *)data
```

**Parameters**

*data*

The data object containing the image data.

**Return Value**

An initialized `NSImage` instance, or `nil` if the method cannot create an image representation from the contents of the specified data object.

**Availability**

Available in Mac OS X v10.0 and later.



**Declared In**

NSImage.h

**initWithDataIgnoringOrientation:**

Initializes and returns an `NSImage` instance with the contents of the specified `NSData` object, ignoring the EXIF orientation tags..

```
- (id)initWithDataIgnoringOrientation:(NSData *)data
```

**Parameters***data*

The data object containing the image data.

**Return Value**

An initialized `NSImage` instance, or `nil` if the method cannot create an image representation from the contents of the specified data object.

**Availability**

Available in Mac OS X v10.6 and later.

**Declared In**

NSImage.h

**initWithIconRef:**

Initializes the image object with a Carbon-style icon resource.

```
- (id)initWithIconRef:(IconRef)iconRef
```

**Parameters***iconRef*

A reference to a Carbon icon resource.

**Return Value**

An initialized `NSImage` instance.

**Discussion**

Creates one or more bitmap image representations, one for each size icon contained in the `IconRef` data structure. This initialization method automatically retains the data in the `iconRef` parameter and loads the bitmaps from that data file lazily.

**Availability**

Available in Mac OS X v10.5 and later.

**Declared In**

NSImage.h

**initWithPasteboard:**

Initializes and returns an `NSImage` instance with data from the specified pasteboard.

```
- (id)initWithPasteboard:(NSPasteboard *)pasteboard
```

**Parameters***pasteboard*

The pasteboard containing the image data.

**Return Value**An initialized `NSImage` instance, or `nil` if the method cannot create an image representation from the contents of the pasteboard.**Discussion**The specified pasteboard should contain a type supported by one of the registered `NSImageRep` subclasses. Table 1 lists the default pasteboard types and file extensions for several `NSImageRep` subclasses.**Table 1** Default pasteboard types for image representations

Image representation class	Default pasteboard type	Default file extensions
<code>NSBitmapImageRep</code>	<code>NSTIFFPboardType</code>	tiff, gif, jpg, and others
<code>NSPDFImageRep</code>	<code>NSPDFPboardType</code>	pdf
<code>NSEPSImageRep</code>	<code>NSPostscriptPboardType</code>	eps
<code>NSPICIImageRep</code>	<code>NSPICTPboardType</code>	pict

If the specified pasteboard contains the value `NSFileNamesPboardType`, each filename on the pasteboard should have an extension supported by one of the registered `NSImageRep` subclasses. You can use the `imageUnfilteredFileTypes` method of a given subclass to obtain the list of supported types for that class.

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**`NSImage.h`**initWithSize:**Initializes and returns an `NSImage` instance whose size is set to the specified value.

- (id)initWithSize:(NSSize)aSize

**Parameters***aSize*

The size of the image, measured in points.

**Return Value**An initialized `NSImage` instance.**Discussion**

This method does not add any image representations to the image object.. It is permissible to initialize the receiver by passing a size of (0.0, 0.0); however, the receiver's size must be set to a non-zero value before the `NSImage` object is used or an exception will be raised.

### Availability

Available in Mac OS X v10.0 and later.

### See Also

- [setSize:](#) (page 43)

### Related Sample Code

CompositeLab

FunHouse

RGB Image

RGB ValueTransformers

Sketch-112

### Declared In

NSImage.h

## isTemplate

Returns a Boolean value indicating whether the image is a template image.

- (BOOL)isTemplate

### Return Value

YES if the image is a template image; otherwise, NO.

### Discussion

Template images consist of black and clear colors (and an alpha channel). Template images are not intended to be used as standalone images and are usually mixed with other content to create the desired final appearance.

### Availability

Available in Mac OS X v10.5 and later.

### See Also

- [setTemplate:](#) (page 44)

### Declared In

NSImage.h

## isValid

Returns a Boolean value indicating whether an image representation from the receiver can be drawn.

- (BOOL)isValid

### Return Value

YES if the receiver can be drawn; otherwise, NO.

**Discussion**

If the receiver is initialized with an existing image file, but the corresponding image data is not yet loaded into memory, this method loads the data and expands it as needed. If the receiver contains no image representations and no associated image file, this method creates a valid cached image representation and initializes it to the default bit depth. This method returns `NO` in cases where the file or URL from which it was initialized is nonexistent or when the data in an existing file is invalid.

**Availability**

Available in Mac OS X v10.0 and later.

**See Also**

- [initWithReferencingFile:](#) (page 29)
- [initWithReferencingURL:](#) (page 30)

**Declared In**

NSImage.h

**lockFocus**

Prepares the image to receive drawing commands.

- (void)lockFocus

**Discussion**

This method sets the current drawing context to the area of the offscreen window used to cache the receiver's contents. Subsequent drawing commands are composited to this offscreen window. If the offscreen drawing area already has some content, any new drawing commands are composited with that content. This method does not modify the original image data directly.

When locking focus, this method chooses the best image representation object available and locks focus on that object. If the receiver has no image representations, this method creates one with the default depth and locks focus on it. For information on how the "best" representation is chosen, see the "Images" chapter of *Cocoa Drawing Guide*.

A successful `lockFocus` message must be balanced with a matching `unlockFocus` (page 47) message to the same `NSImage` object. These messages bracket the code that draws the image.

If `lockFocus` is unable to focus on the image, it raises an `NSImageCacheException`.

**Availability**

Available in Mac OS X v10.0 and later.

**See Also**

- [bestRepresentationForDevice:](#) (page 67)
- [isValid](#) (page 35)
- [prefersColorMatch](#) (page 38)
- [representations](#) (page 39)

**Related Sample Code**

FunHouse

Image Difference

RGB Image

RGB ValueTransformers

Sketch-112

**Declared In**

NSImage.h

**lockFocusFlipped:**

Prepares the image to receive drawing commands using the specified flipped state.

```
- (void)lockFocusFlipped:(BOOL)flipped
```

**Parameters**

*flipped*

YES if the drawing context should be flipped, otherwise NO.

**Availability**

Available in Mac OS X v10.6 and later.

**Declared In**

NSImage.h

**matchesOnMultipleResolution**

Returns a Boolean value indicating whether image representations whose resolution is an integral multiple of the device resolution are considered a match.

```
- (BOOL)matchesOnMultipleResolution
```

**Return Value**

YES if image representations whose resolution is an integral multiple of the device resolution are considered a match; otherwise, NO.

**Discussion**

When this method returns NO, only image representations whose resolution is exactly the same as the device resolution are considered matches. If this method returns YES and multiple image representations fit this criteria, the one whose resolution is closest to the device resolution is chosen.

The default value is YES.

**Availability**

Available in Mac OS X v10.0 and later.

**See Also**

- [setMatchesOnMultipleResolution:](#) (page 42)

**Declared In**

NSImage.h

**name**

Returns the name associated with the receiver, if any.

- (NSString \*)name

**Return Value**

The name associated with the receiver, or `nil` if no name is assigned.

**Availability**

Available in Mac OS X v10.0 and later.

**See Also**

- [setName:](#) (page 42)

**Declared In**

NSImage.h

## prefersColorMatch

Returns a Boolean value indicating whether the image prefers to choose image representations using color matching or resolution matching.

- (BOOL)prefersColorMatch

**Return Value**

YES if color matching is preferred over resolution matching; otherwise NO if resolution matching is preferred.

**Discussion**

Both color matching and resolution matching may influence the choice of an image representation. This method simply indicates which technique is used first during the selection process. The default value is YES.

**Availability**

Available in Mac OS X v10.0 and later.

**See Also**

- [setPrefersColorMatch:](#) (page 43)

**Declared In**

NSImage.h

## recache

Invalidates and frees the offscreen caches of all image representations.

- (void)recache

**Discussion**

If you modify an image representation, you must send a [recache](#) (page 38) message to the corresponding image object to force the changes to be recached. The next time any image representation is drawn, it is asked to recreate its cached image. If you do not send this message, the image representation may use the old cache data. This method simply clears the cached image data; it does not delete the `NSCachedImageRep` objects associated with any image representations.

If you do not plan to use an image again right away, you can free its caches to reduce the amount of memory consumed by your program.

### Availability

Available in Mac OS X v10.0 and later.

### Related Sample Code

TextInputView

### Declared In

NSImage.h

## removeRepresentation:

Removes the specified image representation from the receiver and releases it.

```
- (void)removeRepresentation:(NSImageRep *)imageRep
```

### Parameters

*imageRep*

The image representation object you want to remove.

### Availability

Available in Mac OS X v10.0 and later.

### See Also

- [representations](#) (page 39)

### Declared In

NSImage.h

## representations

Returns an array containing all of the receiver's image representations.

```
- (NSArray *)representations
```

### Return Value

An array containing zero or more NSImageRep objects.

### Availability

Available in Mac OS X v10.0 and later.

### Related Sample Code

CocoaCreateMovie

Reducer

### Declared In

NSImage.h

## setAccessibilityDescription:

Sets the image's accessibility description.

```
- (void)setAccessibilityDescription:(NSString *)description
```

**Parameters***description*

A short localized string that does not include the name of the interface element.

**Discussion**

This description will be used automatically by interface elements that display images. Like all accessibility descriptions, the string should be a short localized string that does not include the name of the interface element. For instance, "delete" rather than "delete button".

**Availability**

Available in Mac OS X v10.6 and later.

**Declared In**

NSImage.h

**setAlignmentRect:**

Sets the alignment metadata that your code can use to position the image during layout.

```
- (void)setAlignmentRect:(NSRect)rect
```

**Parameters***rect*

The alignment rectangle for the image.

**Discussion**

Alignment rectangles specify baselines that you can use to position the content of an image more accurately. These baselines are merely hints that your own code can use to determine positioning and are not used internally by `NSImage` itself during drawing. For example, if you have a 20 x 20 pixel icon that includes a glow effect, you might set the alignment rectangle to `{{2, 2}, {16, 16}}` to indicate the position of the underlying icon without the glow effect.

**Availability**

Available in Mac OS X v10.5 and later.

**See Also**

- [alignmentRect](#) (page 22)

**Declared In**

NSImage.h

**setBackground-color:**

Sets the background color of the image.

```
- (void)setBackgroundColor:(NSColor *)aColor
```

**Parameters***aColor*

The new background color for the image.



**Discussion**

The background color is visible only if the drawn image representation does not completely cover all of the pixels available for the image's current size. The background color is ignored for cached image representations; such caches are always created with a white background. This method does not cause the receiver to recache itself.

**Availability**

Available in Mac OS X v10.0 and later.

**See Also**

- [recache](#) (page 38)
- [backgroundColor](#) (page 22)

**Declared In**

NSImage.h

**setCacheMode:**

Set the receiver's caching mode.

```
- (void)setCacheMode:(NSImageCacheMode)mode
```

**Parameters**

*mode*

The caching mode to use with this image. For a list of possible values, see [NSImageCacheMode](#) (page 53).

**Discussion**

The caching mode determines when the receiver's image representations use offscreen caches. Offscreen caches speed up rendering time but do so by using extra memory. In the default caching mode ([NSImageCacheDefault](#)), each image representation chooses the caching technique that produces the fastest drawing times. For example, in the default mode, the [NSPDFImageRep](#) and [NSEPSImageRep](#) classes use the [NSImageCacheAlways](#) mode but the [NSBitmapImageRep](#) class uses the [NSImageCacheBySize](#) mode.

For more information on image caching behavior, see the "Images" chapter of *Cocoa Drawing Guide*.

**Availability**

Available in Mac OS X v10.2 and later.

**See Also**

- [cacheMode](#) (page 23)

**Declared In**

NSImage.h

**setDelegate:**

Sets the delegate object of the receiver.

```
- (void)setDelegate:(id < NSImageDelegate >)anObject
```

**Parameters***anObject*

The new delegate object.

**Availability**

Available in Mac OS X v10.0 and later.

**See Also**- [delegate](#) (page 25)**Declared In**

NSImage.h

**setMatchesOnMultipleResolution:**

Sets whether image representations whose resolutions are integral multiples of the device resolution are considered a match.

- (void)setMatchesOnMultipleResolution:(BOOL)*flag***Parameters***flag*

YES if image representations whose resolution is an integral multiple of the device resolution should be considered a match; otherwise, NO.

**Availability**

Available in Mac OS X v10.0 and later.

**See Also**- [matchesOnMultipleResolution](#) (page 37)**Declared In**

NSImage.h

**setName:**

Registers the receiver under the specified name.

- (BOOL)setName:(NSString \*)*aString***Parameters***aString*

The name to associate with the receiver.

**Return Value**

YES if the receiver was successfully registered with the given name; otherwise, NO.

**Discussion**If the receiver is already registered under a different name, this method unregisters the other name. If a different image is registered under the name specified in *aString*, this method does nothing and returns NO.

When naming an image using this method, it is convention not to include filename extensions in the names you specify. That way, you can easily distinguish between images you have named explicitly and those you want to load from the application's bundle. For information about the rules used to search for images, and for information about the ownership policy of named images, see the `imageNamed:` method.

#### Availability

Available in Mac OS X v10.0 and later.

#### See Also

- [name](#) (page 37)
- + [imageNamed:](#) (page 16)

#### Related Sample Code

ClockControl  
 QTKitMovieShuffler

#### Declared In

NSImage.h

### setPrefersColorMatch:

Sets whether choosing an image representation favors color matching over resolution matching.

```
- (void)setPrefersColorMatch:(BOOL)flag
```

#### Parameters

*flag*

YES if the receiver should match the color capabilities of the rendering device first; otherwise, NO to indicate that resolution matching is preferred.

#### Discussion

Both color matching and resolution matching may influence the choice of an image representation. You use this method to choose which technique should be used first during the selection process.

#### Availability

Available in Mac OS X v10.0 and later.

#### See Also

- [prefersColorMatch](#) (page 38)

#### Declared In

NSImage.h

### setSize:

Sets the width and height of the image.

```
- (void)setSize:(NSSize)aSize
```

#### Parameters

*aSize*

The new size of the image, measured in points.

**Discussion**

The size of an `NSImage` object must be set before it can be used. If the size of the image hasn't already been set when an image representation is added, the size is taken from the image representation's data. For EPS images, the size is taken from the image's bounding box. For TIFF images, the size is taken from the `ImageLength` and `ImageWidth` attributes.

Changing the size of an `NSImage` after it has been used effectively resizes the image. Changing the size invalidates all its caches and frees them. When the image is next composited, the selected representation will draw itself in an offscreen window to recreate the cache.

**Availability**

Available in Mac OS X v10.0 and later.

**See Also**

- [size](#) (page 45)
- [initWithSize:](#) (page 34)

**Related Sample Code**

ButtonMadness  
 DesktopImage  
 FunkyOverlayWindow  
 SourceView  
 STUCAuthoringDeviceCocoaSample

**Declared In**

`NSImage.h`

**setTemplate:**

Sets whether the image represents a template image.

```
- (void)setTemplate:(BOOL)isTemplate
```

**Parameters**

*isTemplate*

Specify YES if the image is a template image; otherwise, NO.

**Discussion**

Images you mark as template images should consist of only black and clear colors. You can use the alpha channel in the image to adjust the opacity of black content, however.

Template images are not intended to be used as standalone images. They are always mixed with other content and processed to create the desired appearance. You can mark an image as a “template image” to notify clients who care that the image contains only black and clear content. The most common use for template images is in image cells. For example, you might use a template image to provide the content for a button or segmented control. Cocoa cells take advantage of the nature of template images—that is, their simplified color scheme and use of transparency—to improve the appearance of the corresponding control in each of its supported states.

**Availability**

Available in Mac OS X v10.5 and later.

**See Also**

- [isTemplate](#) (page 35)

**Declared In**

NSImage.h

**setUsesEPSOnResolutionMismatch:**

Sets whether EPS image representations are preferred when no other representations match the resolution of the device.

- (void)setUsesEPSOnResolutionMismatch:(BOOL)flag

**Parameters**

*flag*

YES if EPS image representations are preferred; otherwise NO.

**Availability**

Available in Mac OS X v10.0 and later.

**See Also**

- [usesEPSOnResolutionMismatch](#) (page 48)

- [setMatchesOnMultipleResolution:](#) (page 42)

**Declared In**

NSImage.h

**size**

Returns the size of the receiver.

- (NSSize)size

**Return Value**

The size of the receiver or (0.0, 0.0) if no size has been set and the size cannot be determined from any of the receiver's image representations.

**Availability**

Available in Mac OS X v10.0 and later.

**See Also**

- [setSize:](#) (page 43)

**Related Sample Code**

DragNDropOutlineView

FunHouse

NineSlice

RGB Image

Sketch-112

**Declared In**

NSImage.h

## TIFFRepresentation

Returns a data object containing TIFF data for all of the image representations in the receiver.

- (NSData \*)TIFFRepresentation

### Return Value

A data object containing the TIFF data, or `nil` if the TIFF data could not be created.

### Discussion

You can use the returned data object to write the TIFF data to a file. For each image representation, this method uses the TIFF compression option associated with that representation or `NSTIFFCompressionNone`, if no option is set.

If one of the receiver's image representations does not support the creation of TIFF data natively (PDF and EPS images, for example), this method creates the TIFF data from that representation's cached content.

Additional image formats can be saved by using the `NSBitmapImageRep` method `representationUsingType:properties:`.

### Availability

Available in Mac OS X v10.0 and later.

### See Also

- [TIFFRepresentationUsingCompression:factor:](#) (page 46)

`representationUsingType:properties:` (`NSBitmapImageRep`)

`TIFFRepresentation` (`NSBitmapImageRep`)

`TIFFRepresentationUsingCompression:factor:` (`NSBitmapImageRep`)

### Related Sample Code

[GLSLShowpiece](#)

[GLUT](#)

[ImageKitDemo](#)

[People](#)

[Sketch-112](#)

### Declared In

`NSImage.h`

## TIFFRepresentationUsingCompression:factor:

Returns a data object containing TIFF data with the specified compression settings for all of the image representations in the receiver.

- (NSData \*)TIFFRepresentationUsingCompression:(NSTIFFCompression)comp  
factor:(float)aFloat

### Parameters

*comp*

The type of compression to use. For a list of values, see the constants in `NSBitmapImageRep`.

*aFloat*

Provides a hint for compression types that implement variable compression ratios. Currently, only JPEG compression uses a compression factor.

**Return Value**

A data object containing the TIFF data, or `nil` if the TIFF data could not be created.

**Discussion**

You can use the returned data object to write the TIFF data to a file. If the specified compression isn't applicable, no compression is used. If a problem is encountered during generation of the TIFF data, this method may raise an exception.

If one of the receiver's image representations does not support the creation of TIFF data natively (PDF and EPS images, for example), this method creates the TIFF data from that representation's cached content.

Additional image formats can be saved by using the `NSBitmapImageRep` method `representationUsingType:properties:`.

**Availability**

Available in Mac OS X v10.0 and later.

**See Also**

- [TIFFRepresentation](#) (page 46)

`representationUsingType:properties:` (`NSBitmapImageRep`)

`TIFFRepresentation` (`NSBitmapImageRep`)

`TIFFRepresentationUsingCompression:factor:` (`NSBitmapImageRep`)

**Related Sample Code**

PDFKitLinker2

**Declared In**

NSImage.h

## unlockFocus

Removes the focus from the receiver.

- (void)unlockFocus

**Discussion**

This message must be sent after a successful `lockFocus` or `lockFocusOnRepresentation:` message and the completion of any intermediate drawing commands. This method restores the focus to the previous owner, if any.

Do not send this message if the preceding call to `lock focus` raised an `NSImageCacheException`.

**Availability**

Available in Mac OS X v10.0 and later.

**Related Sample Code**

FunHouse

Image Difference

RGB Image

RGB ValueTransformers

Sketch-112

**Declared In**

NSImage.h

**usesEPSOnResolutionMismatch**

Returns a Boolean value indicating whether EPS representations are preferred when no other representations match the resolution of the device.

- (BOOL)usesEPSOnResolutionMismatch

**Return Value**

YES if EPS image representations are preferred; otherwise NO.

**Discussion**

The default value is NO.

**Availability**

Available in Mac OS X v10.0 and later.

**See Also**

- [setUsesEPSOnResolutionMismatch:](#) (page 45)
- [matchesOnMultipleResolution](#) (page 37)

**Declared In**

NSImage.h

## Constants

**Image Hint Dictionary Keys**

These constants are a subset of the dictionary keys used in the hints dictionary for the methods [CGImageForProposedRect:context:hints:](#) (page 24), [bestRepresentationForRect:context:hints:](#) (page 23), [drawInRect:fromRect:operation:fraction:respectFlipped:hints:](#) (page 27), and [hitTestRect:withImageDestinationRect:context:hints:flipped:](#) (page 28). Additional hint keys are also valid including: Context Options in CIColorContext, and the entries in an NSScreen device description dictionary as described in deviceDescription.

```
NSString *const NSImageHintCTM;
NSString *const NSImageHintInterpolation;
```

**Constants**

NSImageHintCTM

Provides a context transform hint. The value for this key is an NSAffineTransform.

Available in Mac OS X v10.6 and later.

Declared in NSImage.h.



NSImageHintInterpolation

Provides an interpolation hint. The value for this key is an `NSNumber` with an `NSImageInterpolation` value.

Available in Mac OS X v10.6 and later.

Declared in `NSImage.h`.

## NSCompositingOperation

These constants specify compositing operators described in terms of having source and destination images, each having an opaque and transparent region. The destination image after the operation is defined in terms of the source and destination before images.

```
enum {
    NSCompositeClear           = 0,
    NSCompositeCopy           = 1,
    NSCompositeSourceOver     = 2,
    NSCompositeSourceIn      = 3,
    NSCompositeSourceOut     = 4,
    NSCompositeSourceAtop    = 5,
    NSCompositeDestinationOver = 6,
    NSCompositeDestinationIn = 7,
    NSCompositeDestinationOut = 8,
    NSCompositeDestinationAtop = 9,
    NSCompositeXOR           = 10,
    NSCompositePlusDarker    = 11,
    NSCompositeHighlight     = 12,
    NSCompositePlusLighter   = 13
}
typedef NSUInteger NSCompositingOperation;
```

### Constants

NSCompositeClear

Transparent. ( $R = 0$ )

Available in Mac OS X v10.0 and later.

Declared in `NSGraphics.h`.

NSCompositeCopy

Source image. ( $R = S$ )

Available in Mac OS X v10.0 and later.

Declared in `NSGraphics.h`.

NSCompositeSourceOver

Source image wherever source image is opaque, and destination image elsewhere. ( $R = S + D*(1 - Sa)$ )

Available in Mac OS X v10.0 and later.

Declared in `NSGraphics.h`.

NSCompositeSourceIn

Source image wherever both images are opaque, and transparent elsewhere. ( $R = S*Da$ )

Available in Mac OS X v10.0 and later.

Declared in `NSGraphics.h`.

`NSCompositeSourceOut`

Source image wherever source image is opaque but destination image is transparent, and transparent elsewhere. ( $R = S*(1 - D_a)$ )

Available in Mac OS X v10.0 and later.

Declared in `NSGraphics.h`.

`NSCompositeSourceAtop`

Source image wherever both images are opaque, destination image wherever destination image is opaque but source image is transparent, and transparent elsewhere. ( $R = S*D_a + D*(1 - S_a)$ )

Available in Mac OS X v10.0 and later.

Declared in `NSGraphics.h`.

`NSCompositeDestinationOver`

Destination image wherever destination image is opaque, and source image elsewhere. ( $R = S*(1 - D_a) + D$ )

Available in Mac OS X v10.0 and later.

Declared in `NSGraphics.h`.

`NSCompositeDestinationIn`

Destination image wherever both images are opaque, and transparent elsewhere. ( $R = D*S_a$ )

Available in Mac OS X v10.0 and later.

Declared in `NSGraphics.h`.

`NSCompositeDestinationOut`

Destination image wherever destination image is opaque but source image is transparent, and transparent elsewhere. ( $R = D*(1 - S_a)$ )

Available in Mac OS X v10.0 and later.

Declared in `NSGraphics.h`.

`NSCompositeDestinationAtop`

Destination image wherever both images are opaque, source image wherever source image is opaque but destination image is transparent, and transparent elsewhere. ( $R = S*(1 - D_a) + D*S_a$ )

Available in Mac OS X v10.0 and later.

Declared in `NSGraphics.h`.

`NSCompositeXOR`

Exclusive OR of source and destination images. ( $R = S*(1 - D_a) + D*(1 - S_a)$ )

Works only with black and white images and is not recommended for color contexts.

Available in Mac OS X v10.0 and later.

Declared in `NSGraphics.h`.

`NSCompositePlusDarker`

Sum of source and destination images, with color values approaching 0 as a limit. ( $R = \text{MAX}(0, (1 - D) + (1 - S))$ )

Available in Mac OS X v10.0 and later.

Declared in `NSGraphics.h`.

NSCompositeHighlight

Source image wherever source image is opaque, and destination image elsewhere. (Deprecated. Mapped to NSCompositeSourceOver.)

Available in Mac OS X v10.0 and later.

Declared in NSGraphics.h.

NSCompositePlusLighter

Sum of source and destination images, with color values approaching 1 as a limit. ( $R = \text{MIN}(1, S + D)$ )

Available in Mac OS X v10.0 and later.

Declared in NSGraphics.h.

**Discussion**

These compositing operators are defined in and used by [compositeToPoint:fromRect:operation:](#) (page 68), [compositeToPoint:operation:](#) (page 70), [compositeToPoint:fromRect:operation:fraction:](#) (page 69), [compositeToPoint:operation:fraction:](#) (page 71), [drawAtPoint:fromRect:operation:fraction:](#) (page 25), and [drawInRect:fromRect:operation:fraction:](#) (page 26). They are also used by drawing methods in other classes that take a compositing operator.

The equations after each constant represent the mathematical formulas used to calculate the color value of the resulting pixel. Table 2 lists the meaning of each placeholder value in the equations.

**Table 2** Placeholder values for compositing equations

Para	Para
R	The premultiplied result color.
S	The source color
D	The destination color
Sa	The alpha value of the source color
Da	The alpha value of the destination color

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

NSGraphics.h

**NSImageLoadStatus**

These constants are status values passed to the incremental loading delegate method `image:didLoadRepresentation:withStatus:.`

```
enum {
    NSImageLoadStatusCompleted,
    NSImageLoadStatusCancelled,
    NSImageLoadStatusInvalidData,
    NSImageLoadStatusUnexpectedEOF,
    NSImageLoadStatusReadError
}
typedef NSUInteger NSImageLoadStatus;
```

**Constants**

`NSImageLoadStatusCompleted`

Enough data has been provided to completely decompress the image.

Available in Mac OS X v10.2 and later.

Declared in `NSImage.h`.

`NSImageLoadStatusCancelled`

Image loading was canceled.

The image contains the portions of the data that have already been successfully decompressed, if any.

Available in Mac OS X v10.2 and later.

Declared in `NSImage.h`.

`NSImageLoadStatusInvalidData`

An error occurred during image decompression.

The image data is probably corrupt. The image contains the portions of the data that have already been successfully decompressed, if any.

Available in Mac OS X v10.2 and later.

Declared in `NSImage.h`.

`NSImageLoadStatusUnexpectedEOF`

Not enough data was available for full decompression of the image.

The image contains the portions of the data that have already been successfully decompressed, if any.

Available in Mac OS X v10.2 and later.

Declared in `NSImage.h`.

`NSImageLoadStatusReadError`

Not enough data was available for full decompression of the image.

The image contains the portions of the data that have already been successfully decompressed, if any.

Available in Mac OS X v10.2 and later.

Declared in `NSImage.h`.

**Availability**

Available in Mac OS X v10.2 and later.

**Declared In**

`NSImage.h`

## NSImageCacheMode

These constants specify the caching policy on a per `NSImage` basis. The caching policy is set using `cacheMode` (page 23) and `setCacheMode:` (page 41).

```
enum {
    NSImageCacheDefault,
    NSImageCacheAlways,
    NSImageCacheBySize,
    NSImageCacheNever
}
typedef NSUInteger NSImageCacheMode;
```

### Constants

`NSImageCacheDefault`

Caching is unspecified.

Use the image rep's default.

Available in Mac OS X v10.2 and later.

Declared in `NSImage.h`.

`NSImageCacheAlways`

Always generate a cache when drawing.

Available in Mac OS X v10.2 and later.

Declared in `NSImage.h`.

`NSImageCacheBySize`

Cache if cache size is smaller than the original data.

Available in Mac OS X v10.2 and later.

Declared in `NSImage.h`.

`NSImageCacheNever`

Never cache; always draw direct.

Available in Mac OS X v10.2 and later.

Declared in `NSImage.h`.

### Discussion

The following table specifies the default caching policy for the various types of image representation.

Image Rep Class	Default caching policy
<code>NSBitmapImageRep</code>	<code>NSImageCacheBySize</code> . Cache if bitmap is 32-bits in 16-bit world or greater than 72 dpi.
<code>NSPICTImageRep</code>	<code>NSImageCacheBySize</code> . Same reasoning as <code>NSBitmapImageRep</code> in the event the PICT contains a bitmap.
<code>NSPDFImageRep</code>	<code>NSImageCacheAlways</code>
<code>NSCIImageRep</code>	<code>NSImageCacheBySize</code> . Cache if the bitmap depth does not match the screen depth or the resolution is greater than 72 dpi.
<code>NSEPSImageRep</code>	<code>NSImageCacheAlways</code>
<code>NSCustomImageRep</code>	<code>NSImageCacheAlways</code>

**Availability**

Available in Mac OS X v10.2 and later.

**Declared In**

NSImage.h

**Image Template Constants**

Images representing standard artwork and icons that you can use in your applications

```

NSString *const NSImageNameQuickLookTemplate;
NSString *const NSImageNameBluetoothTemplate;
NSString *const NSImageNameIChatTheaterTemplate;
NSString *const NSImageNameSlideshowTemplate;
NSString *const NSImageNameActionTemplate;
NSString *const NSImageNameSmartBadgeTemplate;
NSString *const NSImageNamePathTemplate;
NSString *const NSImageNameInvalidDataFreestandingTemplate;
NSString *const NSImageNameLockLockedTemplate;
NSString *const NSImageNameLockUnlockedTemplate;
NSString *const NSImageNameGoRightTemplate;
NSString *const NSImageNameGoLeftTemplate;
NSString *const NSImageNameRightFacingTriangleTemplate;
NSString *const NSImageNameLeftFacingTriangleTemplate;
NSString *const NSImageNameAddTemplate;
NSString *const NSImageNameRemoveTemplate;
NSString *const NSImageNameRevealFreestandingTemplate;
NSString *const NSImageNameFollowLinkFreestandingTemplate;
NSString *const NSImageNameEnterFullScreenTemplate;
NSString *const NSImageNameExitFullScreenTemplate;
NSString *const NSImageNameStopProgressTemplate;
NSString *const NSImageNameStopProgressFreestandingTemplate;
NSString *const NSImageNameRefreshTemplate;
NSString *const NSImageNameRefreshFreestandingTemplate;
NSString *const NSImageNameFolder;
NSString *const NSImageNameTrashEmpty;
NSString *const NSImageNameTrashFull;
NSString *const NSImageNameHomeTemplate;
NSString *const NSImageNameBookmarksTemplate;
NSString *const NSImageNameCaution;
NSString *const NSImageNameStatusAvailable;
NSString *const NSImageNameStatusPartiallyAvailable;
NSString *const NSImageNameStatusUnavailable;
NSString *const NSImageNameStatusNone;
NSString *const NSImageNameApplicationIcon;
NSString *const NSImageNameMenuOnStateTemplate;
NSString *const NSImageNameMenuMixedStateTemplate;
NSString *const NSImageNameUserGuest;
NSString *const NSImageNameMobileMe;

```

**Constants**

NSImageNameQuickLookTemplate

A Quick Look template image. 

Available in Mac OS X v10.5 and later.

Declared in NSImage.h.


NSImageNameBluetoothTemplate

A Bluetooth template image. 

Available in Mac OS X v10.5 and later.

Declared in NSImage.h.

NSImageNameIChatTheaterTemplate

An iChat Theater template image. 

Available in Mac OS X v10.5 and later.

Declared in NSImage.h.


NSImageNameSlideshowTemplate

A slideshow template image. 

Available in Mac OS X v10.5 and later.

Declared in NSImage.h.


NSImageNameActionTemplate

An action menu template image. 

Available in Mac OS X v10.5 and later.

Declared in NSImage.h.


NSImageNameSmartBadgeTemplate

A badge for a “smart” item. 

Available in Mac OS X v10.5 and later.

Declared in NSImage.h.


NSImageNamePathTemplate

A path button template image. 

Available in Mac OS X v10.5 and later.

Declared in NSImage.h.

NSImageNameInvalidDataFreestandingTemplate

An invalid data template image. Place this icon to the right of any fields containing invalid data. You can use this image to implement a borderless button. 

Available in Mac OS X v10.5 and later.

Declared in NSImage.h.


NSImageNameLockLockedTemplate

A locked lock template image. Use to indicate locked content. 

Available in Mac OS X v10.5 and later.

Declared in NSImage.h.

NSImageNameLockUnlockedTemplate

An unlocked lock template image. Use to indicate modifiable content that can be locked. 

Available in Mac OS X v10.5 and later.

Declared in NSImage.h.

`NSImageNameGoRightTemplate`

A “go forward” template image. ▶

Available in Mac OS X v10.5 and later.

Declared in `NSImage.h`.

`NSImageNameGoLeftTemplate`

A “go back” template image. ◀

Available in Mac OS X v10.5 and later.

Declared in `NSImage.h`.

`NSImageNameRightFacingTriangleTemplate`

A generic right-facing triangle template image. ▶

Available in Mac OS X v10.5 and later.

Declared in `NSImage.h`.

`NSImageNameLeftFacingTriangleTemplate`

A generic left-facing triangle template image. ◀

Available in Mac OS X v10.5 and later.

Declared in `NSImage.h`.

`NSImageNameAddTemplate`

An add item template image. +

Available in Mac OS X v10.5 and later.

Declared in `NSImage.h`.

`NSImageNameRemoveTemplate`

A remove item template image. -

Available in Mac OS X v10.5 and later.

Declared in `NSImage.h`.

`NSImageNameRevealFreestandingTemplate`

A reveal contents template image. You can use this image to implement a borderless button. 🔍

Available in Mac OS X v10.5 and later.

Declared in `NSImage.h`.

`NSImageNameFollowLinkFreestandingTemplate`

A link template image. You can use this image to implement a borderless button. ➡

Available in Mac OS X v10.5 and later.

Declared in `NSImage.h`.

`NSImageNameEnterFullScreenTemplate`

An enter full-screen mode template image. 🖥️

Available in Mac OS X v10.5 and later.

Declared in `NSImage.h`.

`NSImageNameExitFullScreenTemplate`


An exit full-screen mode template image. 🖥️

Available in Mac OS X v10.5 and later.

Declared in `NSImage.h`.




`NSImageNameStopProgressTemplate`

A stop progress button template image. 

Available in Mac OS X v10.5 and later.

Declared in `NSImage.h`.

`NSImageNameStopProgressFreestandingTemplate`

A stop progress template image. You can use this image to implement a borderless button. 

Available in Mac OS X v10.5 and later.

Declared in `NSImage.h`.


`NSImageNameRefreshTemplate`

A refresh template image. 

Available in Mac OS X v10.5 and later.

Declared in `NSImage.h`.

`NSImageNameRefreshFreestandingTemplate`

A refresh template image. You can use this image to implement a borderless button. 

Available in Mac OS X v10.5 and later.

Declared in `NSImage.h`.

### Discussion

To access these images, pass the specified constant to the `imageNamed:` (page 16) method.

Images with the word “Template” in their title identify shapes that are not intended as standalone images. You would typically use these icons as the custom image for a button, or you might apply them to a cell in a control. For example, you might use the `NSImageNameLockLockedTemplate` image to indicate an item is not modifiable. Template images should use black and clear colors only and it is fine to include varying levels of alpha.

Images with the word “Freestanding” in their title can be used to implement borderless buttons. You do not need to include any extra bezel artwork behind such images.

You should always use named images according to their intended purpose, and not according to how the image appears when loaded. The appearance of images can change between releases. If you use an image for its intended purpose (and not because of it looks), your code should look correct from release to release.

The size and aspect ratio of system images may change from release to release. In some situations, you should explicitly resize images as appropriate for your use. If you use these images in conjunction with an `NSButtonCell` object, however, you can use the `setImageScaling:` method of the cell to control scaling instead. Similarly, for an `NSSegmentedCell` object, you can use the `setImageScaling:forSegment:` method to control scaling.

The string value for each constant is equal to the constant name without the “ImageName” portion. You might need this information to locate images by name in Interface Builder. For example, the constant `NSImageNameRefreshFreestandingTemplate` would correspond to an image named “NSRefreshFreestandingTemplate” in Interface Builder.

### Declared In

`NSImage.h`


## Multiple Documents Drag Image

Drag images you can use in your applications. To access this image, pass the specified constant to the `imageNamed:` (page 16) method.

```
NSString *const NSImageNameMultipleDocuments;
```

### Constants

`NSImageNameMultipleDocuments`

A drag image for multiple items. 

Available in Mac OS X v10.5 and later.

Declared in `NSImage.h`.

### Discussion

You can use this icon as the drag image when dragging multiple items. You should not use this image for any other intended purpose, however. The appearance of images can change between releases. If you use an image for its intended purpose (and not because of how it looks), your code should look correct from release to release.

The size and aspect ratio of system images may change from release to release. In some situations, you should explicitly resize images as appropriate for your use. If you use these images in conjunction with an `NSButtonCell` object, however, you can use the `setImageScaling:` method of the cell to control scaling instead. Similarly, for an `NSSegmentedCell` object, you can use the `setImageScaling:forSegment:` method to control scaling.

The string value for each constant is equal to the constant name without the “ImageName” portion. You might need this information to locate images by name in Interface Builder. For example, the constant `NSImageNameMultipleDocuments` would correspond to an image named “NSMultipleDocuments” in Interface Builder.


## Sharing Permissions Named Images

Images representing sharing permission icons that you can use in your applications. To access this image, pass the specified constant to the `imageNamed:` (page 16) method.

```
NSString *const NSImageNameUser;
NSString *const NSImageNameUserGroup;
NSString *const NSImageNameEveryone;
```

### Constants


`NSImageNameUser`

Permissions for a single user. 

Available in Mac OS X v10.5 and later.

Declared in `NSImage.h`.


`NSImageNameUserGroup`

Permissions for a group of users. 

Available in Mac OS X v10.5 and later.

Declared in `NSImage.h`.

NSImageNameEveryone

Permissions for all users. 

Available in Mac OS X v10.5 and later.

Declared in NSImage.h.

**Discussion**

You should use these images to reflect user and group permission or sharing information. The appearance of images can change between releases. If you use an image for its intended purpose (and not because of how it looks), your code should look correct from release to release.

The size and aspect ratio of system images may change from release to release. In some situations, you should explicitly resize images as appropriate for your use. If you use these images in conjunction with an `NSButtonCell` object, however, you can use the `setImageScaling:` method of the cell to control scaling instead. Similarly, for an `NSSegmentedCell` object, you can use the `setImageScaling:forSegment:` method to control scaling.

The string value for each constant is equal to the constant name without the “ImageName” portion. You might need this information to locate images by name in Interface Builder. For example, the constant `NSImageNameEveryone` would correspond to an image named “NSEveryone” in Interface Builder.

**System Entity Images**

Images representing Finder items. To access this image, pass the specified constant to the `imageName:` (page 16) method.

```
NSString *const NSImageNameBonjour;
NSString *const NSImageNameDotMac;
NSString *const NSImageNameComputer;
NSString *const NSImageNameFolderBurnable;
NSString *const NSImageNameFolderSmart;
NSString *const NSImageNameNetwork;
```

**Constants**


NSImageNameBonjour

A Bonjour icon. 

Available in Mac OS X v10.5 and later.

Declared in NSImage.h.


NSImageNameDotMac

A Dot Mac icon. 

Available in Mac OS X v10.5 and later.

Declared in NSImage.h.


NSImageNameComputer

A computer icon. 

Available in Mac OS X v10.5 and later.

Declared in NSImage.h.

`NSImageNameFolderBurnable`

A burnable folder icon. 

Available in Mac OS X v10.5 and later.

Declared in `NSImage.h`.


`NSImageNameFolderSmart`

A smart folder icon. 

Available in Mac OS X v10.5 and later.

Declared in `NSImage.h`.

`NSImageNameNetwork`

A network icon. 

Available in Mac OS X v10.5 and later.

Declared in `NSImage.h`.

### Discussion

You should use these images to reflect specific elements of the Mac OS X environment. For example, you might use the burnable folder icon if your software allows the user to organize content for burning onto an optical disk. The appearance of images can change between releases. If you use an image for its intended purpose (and not because of how it looks), your code should look correct from release to release.

The size and aspect ratio of system images may change from release to release. In some situations, you should explicitly resize images as appropriate for your use. If you use these images in conjunction with an `NSButtonCell` object, however, you can use the `setImageScaling:` method of the cell to control scaling instead. Similarly, for an `NSSegmentedCell` object, you can use the `setImageScaling:forSegment:` method to control scaling.

The string value for each constant is equal to the constant name without the “ImageName” portion. You might need this information to locate images by name in Interface Builder. For example, the constant `NSImageNameNetwork` would correspond to an image named “NSNetwork” in Interface Builder.

### Declared In

`NSImage.h`


## Toolbar Named Images

Images that you can use in application toolbars. To access this image, pass the specified constant to the `imageName:` (page 16) method.

```
NSString *const NSImageNameUserAccounts;
NSString *const NSImageNamePreferencesGeneral;
NSString *const NSImageNameAdvanced;
NSString *const NSImageNameInfo;
NSString *const NSImageNameFontPanel;
NSString *const NSImageNameColorPanel;
```

### Constants


NSImageNameUserAccounts

User account toolbar icon. Use in a preferences window only. 

Available in Mac OS X v10.5 and later.

Declared in NSImage.h.

NSImageNamePreferencesGeneral

General preferences toolbar icon. Use in a preferences window only. 

Available in Mac OS X v10.5 and later.

Declared in NSImage.h.


NSImageNameAdvanced

Advanced preferences toolbar icon. Use in a preferences window only. 

Available in Mac OS X v10.5 and later.

Declared in NSImage.h.

NSImageNameInfo

An information toolbar icon. 

Available in Mac OS X v10.5 and later.

Declared in NSImage.h.

NSImageNameFontPanel

A font panel toolbar icon. 

Available in Mac OS X v10.5 and later.

Declared in NSImage.h.

NSImageNameColorPanel

A color panel toolbar icon. 

Available in Mac OS X v10.5 and later.

Declared in NSImage.h.


NSImageNameFolder

A folder image. 

Available in Mac OS X v10.6 and later.

Declared in NSImage.h.


NSImageNameTrashEmpty

An image of the empty trash can. 

Available in Mac OS X v10.6 and later.

Declared in `NSImage.h`.


NSImageNameTrashFull

An image of the full trash can. 

Available in Mac OS X v10.6 and later.

Declared in `NSImage.h`.


NSImageNameHomeTemplate

Home image suitable for a template. 

Available in Mac OS X v10.6 and later.

Declared in `NSImage.h`.

NSImageNameBookmarksTemplate

Bookmarks image suitable for a template. 

Available in Mac OS X v10.6 and later.

Declared in `NSImage.h`.


NSImageNameCaution

Caution Image. 

Available in Mac OS X v10.6 and later.

Declared in `NSImage.h`.

NSImageNameStatusAvailable

Small green indicator, similar to iChat's available image. 

Available in Mac OS X v10.6 and later.

Declared in `NSImage.h`.


NSImageNameStatusPartiallyAvailable

Small yellow indicator, similar to iChat's idle image. 

Available in Mac OS X v10.6 and later.

Declared in `NSImage.h`.


NSImageNameStatusUnavailable

Small red indicator, similar to iChat's unavailable image. 

Available in Mac OS X v10.6 and later.

Declared in `NSImage.h`.

NSImageNameStatusNone

Small clear indicator. 

Available in Mac OS X v10.6 and later.

Declared in `NSImage.h`.

NSImageNameApplicationIcon

Generic application icon. 

Available in Mac OS X v10.6 and later.

Declared in `NSImage.h`.

NSImageNameMenuOnStateTemplate

A check mark. Drawing these outside of menus is discouraged. ✓

Available in Mac OS X v10.6 and later.

Declared in `NSImage.h`.


NSImageNameMenuMixedStateTemplate

A horizontal dash. Drawing these outside of menus is discouraged. —

Available in Mac OS X v10.6 and later.

Declared in `NSImage.h`.

NSImageNameUserGuest

Shaded user figure. 

Available in Mac OS X v10.6 and later.

Declared in `NSImage.h`.

NSImageNameMobileMe

MobileMe logo. Note that this is preferred to using the [NSImageNameDotMac](#) (page 59) image,

although that image is not expected to be deprecated. 

Available in Mac OS X v10.6 and later.

Declared in `NSImage.h`.

**Discussion**

You should use these images as icons for toolbar items. The appearance of images can change between releases. If you use an image for its intended purpose (and not because of how it looks), your code should look correct from release to release.

The size and aspect ratio of system images may change from release to release. In some situations, you should explicitly resize images as appropriate for your use. If you use these images in conjunction with an `NSButtonCell` object, however, you can use the `setImageScaling:` method of the cell to control scaling instead. Similarly, for an `NSSegmentedCell` object, you can use the `setImageScaling:forSegment:` method to control scaling.

Constants that end in the word "Template" name black and clear images that return YES for `isTemplate` (page 35). These images can be processed into variants appropriate for different situations. For example, these images can invert in a selected table view row. See `setTemplate:` (page 44): for more comments. These images are inappropriate for display without further processing, but `NSCell` and its subclasses will perform the processing.

Some images also contain the word "Freestanding". This indicates that an image is appropriate for use as a borderless button, it doesn't need any extra bezel artwork behind it. For example, Safari uses [NSImageNameStopProgressFreestandingTemplate](#) (page 57) as the stop button in a button on its toolbar, while it uses [NSImageNameStopProgressFreestandingTemplate](#) (page 57) in the downloads window where it appears inline with a progress indicator.

The string value for each constant is equal to the constant name without the “ImageName” portion. You might need this information to locate images by name in Interface Builder. For example, the constant `NSImageNameColorPanel` would correspond to an image named “NSColorPanel” in Interface Builder.

### Declared In

`NSImage.h`


## View Type Template Images

Images used in segmented controls to switch the current view type. To access this image, pass the specified constant to the `imageName:` (page 16) method.

```
NSString *const NSImageNameIconViewTemplate;
NSString *const NSImageNameListViewTemplate;
NSString *const NSImageNameColumnViewTemplate;
NSString *const NSImageNameFlowViewTemplate;
```

### Constants


`NSImageNameIconViewTemplate`

An icon view mode template image. 

Available in Mac OS X v10.5 and later.

Declared in `NSImage.h`.

`NSImageNameListViewTemplate`

A list view mode template image. 

Available in Mac OS X v10.5 and later.

Declared in `NSImage.h`.


`NSImageNameColumnViewTemplate`

A column view mode template image. 

Available in Mac OS X v10.5 and later.

Declared in `NSImage.h`.

`NSImageNameFlowViewTemplate`

A cover flow view mode template image. 

Available in Mac OS X v10.5 and later.

Declared in `NSImage.h`.

### Discussion

Images with the word “Template” in their title identify shapes that are not intended as standalone images. You would typically use these icons as the custom image for a button, or you might apply them to a cell in a control. For example, you might use the `NSImageNameIconViewTemplate` image to indicate an item is not modifiable. Template images should use black and clear colors only and it is fine to include varying levels of alpha.

You should use these images in conjunction with the buttons (usually part of a segmented control) that change the current viewing mode. The appearance of images can change between releases. If you use an image for its intended purpose (and not because of how it looks), your code should look correct from release to release.



The size and aspect ratio of system images may change from release to release. In some situations, you should explicitly resize images as appropriate for your use. If you use these images in conjunction with an `NSButtonCell` object, however, you can use the `setImageScaling:` method of the cell to control scaling instead. Similarly, for an `NSSegmentedCell` object, you can use the `setImageScaling:forSegment:` method to control scaling.

The string value for each constant is equal to the constant name without the “ImageName” portion. You might need this information to locate images by name in Interface Builder. For example, the constant `NSImageNameFlowViewTemplate` would correspond to an image named “NSFlowViewTemplate” in Interface Builder.

### Declared In

`NSImage.h`



# Deprecated NSImage Methods

---

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

## Deprecated in Mac OS X v10.6

### bestRepresentationForDevice:

Returns the best representation for the device with the specified characteristics. (Deprecated in Mac OS X v10.6.)

```
- (NSImageRep *)bestRepresentationForDevice:(NSDictionary *)deviceDescription
```

#### Parameters

*deviceDescription*

A dictionary of attributes for the specified device, or `nil` to specify the current device. For a list of dictionary keys and values appropriate to display and print devices, see the constants in `NSScreen`.

#### Return Value

The image representation that most closely matches the specified criteria.

#### Discussion

If *deviceDescription* is `nil`, this method uses the attributes of the device on which the content is to be drawn.

For information on how the "best" representation is chosen, see the "Images" chapter of *Cocoa Drawing Guide*.

#### Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.6.

#### See Also

- [representations](#) (page 39)
- [prefersColorMatch](#) (page 38)
- [matchesOnMultipleResolution](#) (page 37)
- [usesEPSOnResolutionMismatch](#) (page 48)

#### Related Sample Code

LayerBackedOpenGLView

NSOpenGL Fullscreen

PDF Annotation Editor

Sketch-112

**Declared In**

NSImage.h

**cacheDepthMatchesImageDepth**

Returns a Boolean value indicating whether an image's offscreen window caches use the same bit depth as the image data itself. (Deprecated in Mac OS X v10.6. NSImage no longer caches to windows. A cache is now generated appropriate for the destination where an image is drawn. There is no replacement method.)

- (BOOL)cacheDepthMatchesImageDepth

**Return Value**

YES if the offscreen window caches use the same bit depth as the image data; otherwise, NO. The default value is NO.

**Availability**

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.6.

**See Also**

- [setCacheDepthMatchesImageDepth](#): (page 76)

**Declared In**

NSImage.h

**compositeToPoint:fromRect:operation:**

Composites a portion of the image to the specified point in the current coordinate system. (Deprecated in Mac OS X v10.6.)

```
- (void)compositeToPoint:(NSPoint)aPoint fromRect:(NSRect)srcRect
    operation:(NSCompositingOperation)op
```

**Parameters**

*aPoint*

The point at which to draw the image, specified in the current coordinate system.

*srcRect*

The portion of the image you want to draw, specified in the image's coordinate system.

*op*

The compositing operation to use when drawing the image to the screen. The supported compositing operations are described in “Constants” (page 48).

**Discussion**

This method draws the specified portion of the image without checking the bounds rectangle you pass into the *srcRect* parameter. If you specify a source rectangle that strays outside of the image's bounds rectangle, it is conceivable that you could composite parts of the offscreen cache window that do not belong to the receiver's image. You can avoid this problem using the [drawAtPoint:fromRect:operation:fraction:](#) (page 25) method, which checks the source rectangle before drawing.

## Deprecated NSImage Methods

During drawing, the image is composited from its offscreen window cache. Because the offscreen cache is not created until the image representation is first used, this method may need to render the image before compositing. Bitmap representations in particular are not cached until they are explicitly rendered. You can use the [lockFocus](#) (page 36) and [unlockFocus](#) (page 47) methods to force the cached version to be created.

Compositing part of an image is as efficient as compositing the whole image, but printing just part of an image is not. When printing, it's necessary to draw the whole image and rely on a clipping path to be sure that only the desired portion appears.

During printing, this method ignores the `op` parameter. Even though this parameter is ignored, this method attempts to render the image as close as possible to its appearance when the compositing operation is used on the screen. In either case, the best image representation is chosen for the printing context.

**Important:** If you are writing new code, or updating old code, you should avoid using this method. Instead, you should use the `drawAtPoint:fromRect:operation:fraction:` or `drawInRect:fromRect:operation:fraction:` method to draw the image. Although the method itself is not deprecated, the behavior it provides is not recommended for general use.

**Availability**

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.6.

**See Also**

- [dissolveToPoint:fromRect:fraction:](#) (page 73)
- [drawAtPoint:fromRect:operation:fraction:](#) (page 25)
- [drawInRect:fromRect:operation:fraction:](#) (page 26)

**Declared In**

NSImage.h

**compositeToPoint:fromRect:operation:fraction:**

Composites a portion of the image at the specified opacity to the current coordinate system. (Deprecated in Mac OS X v10.6.)

```
- (void)compositeToPoint:(NSPoint)aPoint fromRect:(NSRect)srcRect
    operation:(NSCompositingOperation)op fraction:(CGFloat)delta
```

**Parameters**

*aPoint*

The point at which to draw the image, specified in the current coordinate system.

*srcRect*

The portion of the image you want to draw, specified in the image's coordinate system.

*op*

The compositing operation to use when drawing the image to the screen. The supported compositing operations are described in [“Constants”](#) (page 48).

*delta*

The desired opacity of the image, specified as a value between 0.0 and 1.0, with 1.0 representing total opacity. Values larger than 1.0 are interpreted as 1.0. This method always expects to render something, so for values that are equal to or less than 0, this method renders at full opacity.

#### Discussion

Behaves the same as `compositeToPoint:fromRect:operation:` (page 68) except that you can specify the amount of opacity to use when drawing the image.

**Important:** If you are writing new code, or updating old code, you should avoid using this method. Instead, you should use the `drawAtPoint:fromRect:operation:fraction:` or `drawInRect:fromRect:operation:fraction:` method to draw the image. Although the method itself is not deprecated, the behavior it provides is not recommended for general use.

#### Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.6.

#### See Also

- `dissolveToPoint:fromRect:fraction:` (page 73)
- `drawAtPoint:fromRect:operation:fraction:` (page 25)
- `drawInRect:fromRect:operation:fraction:` (page 26)

#### Declared In

NSImage.h

## `compositeToPoint:operation:`

Composites the entire image to the specified point in the current coordinate system. (Deprecated in Mac OS X v10.6.)

```
- (void)compositeToPoint:(NSPoint)aPoint operation:(NSCompositingOperation)op
```

#### Parameters

*aPoint*

The point at which to draw the image, specified in the current coordinate system.

*op*

The compositing operation to use when drawing the image to the screen. The supported compositing operations are described in “Constants” (page 48).

#### Discussion

This method draws the receiver's best image representation at the specified point in the currently focused view. The entire image is drawn using its current size information. During drawing, the image is composited from its offscreen window cache. Because the offscreen cache is not created until the image representation is first used, this method may need to render the image before compositing. Bitmap representations in particular are not cached until they are explicitly rendered. You can use the `lockFocus` (page 36) and `unlockFocus` (page 47) methods to force the cached version to be created.

During printing, this method ignores the `op` parameter. Even though this parameter is ignored, this method attempts to render the image as close as possible to its appearance when the compositing operation is used on the screen. In either case, the best image representation is chosen for the printing context.

**Important:** If you are writing new code, or updating old code, you should avoid using this method. Instead, you should use the `drawAtPoint:fromRect:operation:fraction:` or `drawInRect:fromRect:operation:fraction:` method to draw the image. Although the method itself is not deprecated, the behavior it provides is not recommended for general use.

**Availability**

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.6.

**See Also**

- [dissolveToPoint:fraction:](#) (page 72)
- [drawAtPoint:fromRect:operation:fraction:](#) (page 25)
- [drawInRect:fromRect:operation:fraction:](#) (page 26)

**Related Sample Code**

ColorMatching

Image Difference

RGB ValueTransformers

Sketch-112

STUCAuthoringDeviceCocoaSample

**Declared In**

NSImage.h

**compositeToPoint:operation:fraction:**

Composites the entire image at the specified opacity in the current coordinate system. (Deprecated in Mac OS X v10.6.)

```
- (void)compositeToPoint:(NSPoint)aPoint operation:(NSCompositingOperation)op
    fraction:(CGFloat)delta
```

**Parameters**

*aPoint*

The point at which to draw the image, specified in the current coordinate system.

*op*

The compositing operation to use when drawing the image to the screen. The supported compositing operations are described in “Constants” (page 48).

*delta*

The desired opacity of the image, specified as a value between 0.0 and 1.0, with 1.0 representing total opacity. Values larger than 1.0 are interpreted as 1.0. This method always expects to render something, so for values that are equal to or less than 0, this method renders at full opacity.

**Discussion**

Behaves the same as [compositeToPoint:operation:](#) (page 70) except that you can specify the amount of opacity to use when drawing the image.

**Important:** If you are writing new code, or updating old code, you should avoid using this method. Instead, you should use the `drawAtPoint:fromRect:operation:fraction:` or `drawInRect:fromRect:operation:fraction:` method to draw the image. Although the method itself is not deprecated, the behavior it provides is not recommended for general use.

**Availability**

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.6.

**See Also**

- [dissolveToPoint:fraction:](#) (page 72)
- [drawAtPoint:fromRect:operation:fraction:](#) (page 25)
- [drawInRect:fromRect:operation:fraction:](#) (page 26)

**Declared In**

NSImage.h

**dissolveToPoint:fraction:**

Composites the entire image to the specified location using the `NSCompositeSourceOver` operator. (Deprecated in Mac OS X v10.6.)

```
- (void)dissolveToPoint:(NSPoint)aPoint fraction:(CGFloat)delta
```

**Parameters**

*aPoint*

The point at which to draw the image, specified in the current coordinate system.

*delta*

The desired opacity of the image, specified as a value between 0.0 and 1.0. A value of 0.0 renders the image totally transparent while 1.0 renders it fully opaque. Values larger than 1.0 are interpreted as 1.0.

**Discussion**

Except for the choice of compositing operator, this method behaves in the same way as the [compositeToPoint:operation:](#) (page 70) method. During printing, the *delta* parameter is ignored.

If the source image contains alpha information, this operation may promote the destination `NSWindow` object to contain alpha information.

To slowly dissolve this image onto another, you can invoke this method (or the [dissolveToPoint:fromRect:fraction:](#) (page 73) method) repeatedly with an ever-increasing *delta* value. Because the *delta* parameter refers to the visible fraction of the source image, increasing the value causes the source image to replace the destination content gradually. You should generally perform this type of operation using a buffered window or other offscreen drawing environment.

**Special Considerations**

If you are writing new code, or updating old code, you should avoid using this method. Instead, you should use the `drawAtPoint:fromRect:operation:fraction:` or `drawInRect:fromRect:operation:fraction:` method to draw the image. Although the method itself is not deprecated, the behavior it provides is not recommended for general use.



## Deprecated NSImage Methods

**Availability**

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.6.

**Declared In**

NSImage.h

**dissolveToPoint:fromRect:fraction:**

Composites a portion of the image to the specified location using the `NSCompositeSourceOver` operator. (Deprecated in Mac OS X v10.6.)

```
- (void)dissolveToPoint:(NSPoint)aPoint fromRect:(NSRect)srcRect  
    fraction:(CGFloat)delta
```

**Parameters**

*aPoint*

The point at which to draw the image, specified in the current coordinate system.

*srcRect*

The portion of the image you want to draw, specified in the image's coordinate system.

*delta*

The desired opacity of the image, specified as a value between 0.0 and 1.0. A value of 0.0 renders the image totally transparent while 1.0 renders it fully opaque. Values larger than 1.0 are interpreted as 1.0.

**Discussion**

Except for the choice of compositing operator, this method behaves in the same way as the [compositeToPoint:fromRect:operation:](#) (page 68) method. During printing, the *delta* parameter is ignored.

If the source image contains alpha information, this operation may promote the destination `NSWindow` object to contain alpha information.

**Special Considerations**

If you are writing new code, or updating old code, you should avoid using this method. Instead, you should use the `drawAtPoint:fromRect:operation:fraction:` or `drawInRect:fromRect:operation:fraction:` method to draw the image. Although the method itself is not deprecated, the behavior it provides is not recommended for general use.

**Availability**

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.6.

**See Also**

- [dissolveToPoint:fraction:](#) (page 72)

**Declared In**

NSImage.h

## isCachedSeparately

Returns a Boolean value indicating whether each image representation caches its contents in a separate offscreen window. (Deprecated in Mac OS X v10.6. NSImage no longer caches to windows. There is no replacement method)

- (BOOL)isCachedSeparately

### Return Value

YES if the image representations cache their content in separate offscreen windows; otherwise, NO. The default value is NO.

### Discussion

If this method returns NO, it means that the image may be cached in a shared window but is not required to be. Images are cached in a shared window if they have the same general attributes, such as color space, resolution, and bit depth.

### Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.6.

### Declared In

NSImage.h

## isDataRetained

Returns a Boolean value indicating whether the receiver retains its source image data. (Deprecated in Mac OS X v10.6. In Mac OS v10.6, NSImage no longer discards data in such a way that the original can no longer be reconstructed. There is no replacement method.)

- (BOOL)isDataRetained

### Return Value

YES if the image retains its source data; otherwise, NO. The default value is NO with some exceptions, which are covered in the discussion.

### Discussion

For image objects initialized using either the [initWithReferencingFile:](#) (page 29) or [initWithReferencingURL:](#) (page 30) method, this value is YES by default. The reason is that for these methods, data retention simply involves retaining the filename or URL.

Data retention increases the memory used by the NSImage object and its image representations.

### Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.6.

### Declared In

NSImage.h

## isFlipped

Returns a Boolean value indicating whether the image uses a flipped coordinate system. (Deprecated in Mac OS X v10.6. The flipped property of an image was widely misunderstood and has been deprecated. Use [drawInRect:fromRect:operation:fraction:respectFlipped:hints:](#) (page 27) to draw respecting a context's flipped status and [lockFocusFlipped:](#) (page 37) to draw into a flipped image.)

- (BOOL)isFlipped

### Return Value

YES if the image's coordinate system is flipped; otherwise, NO. The default is NO.

### Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.6.

### See Also

- [setFlipped:](#) (page 78)

### Declared In

NSImage.h

## lockFocusOnRepresentation:

Prepares the specified image representation to receive drawing commands. (Deprecated in Mac OS X v10.6. Use the code fragment shown in the special considerations below.)

- (void)lockFocusOnRepresentation:(NSImageRep \*)*imageRepresentation*

### Parameters

*imageRepresentation*

An image representation belonging to the receiver, or `nil` if you want the receiver to choose which image representation to use.

### Discussion

This method sets the current drawing context to the area of the offscreen window used to cache the specified image representation's contents. Subsequent drawing commands are composited to this offscreen window. If the offscreen drawing area already has some content, any new drawing commands are composited with that content. This method does not modify the original image data directly.

If *imageRepresentation* is `nil`, this method acts like the [lockFocus](#) (page 36) method, setting the focus to the best representation for the NSImage object.

A successful `lockFocusOnRepresentation:` message must be balanced with a matching [unlockFocus](#) (page 47) message to the same NSImage object. These messages bracket the code that draws the image.

If `lockFocusOnRepresentation:` is unable to focus on the specified image representation, it raises an `NSImageCacheException`.

### Special Considerations

This method is deprecated as it did not set up *imageRepresentation* as a drawing destination, it set the image up as a drawing destination, then drew *imageRepresentation* into it. You can replace this functionality with the following code fragment

## Deprecated NSImage Methods

```
[image lockFocus];
[imageRepresentation drawInRect:NSMakeRect(0,0,[image size].width, [image
size].height)];
```

```
image unlockFocus;
```

**Availability**

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.6.

**See Also**

- [isValid](#) (page 35)

**Declared In**

NSImage.h

**scalesWhenResized**

Returns a Boolean value indicating whether image representations are scaled to fit the receiver's size. (Deprecated in Mac OS X v10.6. This method was related to caching behavior. In Mac OS X v10.6 and later image caching is no longer necessary and as a result there is no replacement necessary.)

```
- (BOOL)scalesWhenResized
```

**Return Value**

YES if image representations are scaled to fit the receiver; otherwise, NO. The default value is NO.

**Discussion**

Images are not resized during drawing if this method returns YES. They are only resized when you change the size by sending the receiver a [setSize:](#) (page 43) message.

**Availability**

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.6.

**See Also**

- [setScaleWhenResized:](#) (page 79)

**Declared In**

NSImage.h

**setCacheDepthMatchesImageDepth:**

Sets whether the receiver's offscreen window caches use the same bit depth as the image data itself. (Deprecated in Mac OS X v10.6. NSImage no longer caches to windows. A cache is now generated appropriate for the destination where an image is drawn. There is no replacement method.)

```
- (void)setCacheDepthMatchesImageDepth:(BOOL)flag
```

**Parameters**

*flag*

YES if the offscreen caches use the same bit-depth associated with the image data; otherwise, NO to indicate they should use the default bit depth.

## Deprecated NSImage Methods

**Discussion**

This method does not cause the receiver to recache itself. The default depth limit is equal to the bit depth of the deepest screen on the system.

**Availability**

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.6.

**See Also**

- [cacheDepthMatchesImageDepth](#) (page 68)
- [LockFocus](#) (page 36)
- [recache](#) (page 38)

**Declared In**

NSImage.h

**setCachedSeparately:**

Sets whether each image representation uses a separate offscreen window to cache its contents. (Deprecated in Mac OS X v10.6. NSImage no longer caches to windows. There is no replacement method)

```
- (void)setCachedSeparately:(BOOL)flag
```

**Parameters**

*flag*

YES if you want each of the receiver's image representation objects to use a separate offscreen window for caching; otherwise, NO.

**Discussion**

If you specify NO, a representation can be cached together with other images, though in practice it might not be. This method does not invalidate any existing caches.

If you plan to resize an NSImage object frequently, it is usually more efficient to cache its representations separately. In some situations, you might also want to enable separate caching if you plan to use the [compositeToPoint:fromRect:operation:](#) (page 68) or [compositeToPoint:fromRect:operation:fraction:](#) (page 69) methods to draw the image.

**Availability**

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.6.

**See Also**

- [recache](#) (page 38)

**Declared In**

NSImage.h

**setDataRetained:**

Sets whether the receiver retains its source image data. (Deprecated in Mac OS X v10.6. In Mac OS v10.6, NSImage no longer discards data in such a way that the original can no longer be reconstructed. There is no replacement method.)

## Deprecated NSImage Methods

```
- (void)setDataRetained:(BOOL)flag
```

**Parameters**

*flag*

YES if you want the source image data to be retained; otherwise NO.

**Discussion**

Retention of the source image data is important if the source of the image data could change, be moved, or be deleted. Data retention is also useful if you plan to resize an image frequently; otherwise, resizing occurs on a cached copy of the image, which can lose image quality during successive scaling operations. With data retention enabled, the image is resized from the original source data.

If the responsibility for drawing the image is delegated to another object, there is no reason to retain the image data. Similarly, if the source of the image data is not expected to change or you do not plan to resize the image, you do not need to retain the data. In fact, retaining the data leads to increased memory usage, which could have a negative impact on performance.

If you create your image object using the [initWithReferencingFile:](#) (page 29) method, the only data retained is the name of the source file.

**Availability**

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.6.

**Declared In**

NSImage.h

**setFlipped:**

Sets whether the polarity of the y axis is inverted when drawing an image. (Deprecated in Mac OS X v10.6. The flipped property of an image was widely misunderstood and has been deprecated. Use [drawInRect:fromRect:operation:fraction:respectFlipped:hints:](#) (page 27) to draw respecting a context's flipped status and [lockFocusFlipped:](#) (page 37) to draw into a flipped image.)

```
- (void)setFlipped:(BOOL)flag
```

**Parameters**

*flag*

YES if you want the image data to be inverted before drawing; otherwise, NO.

**Discussion**

If *flag* is YES, the y-axis of the image's internal coordinate system is inverted, with the origin in the upper-left corner and the positive y axis extending downward. This method affects only the coordinate system used internally by the image and the orientation of the image when it is drawn; it does not affect the coordinate system used to specify the position of an image in a view. This method does not cause the receiver to recache itself.

If you set *flag* to YES and then lock focus and draw into the image, the content you draw is cached in the inverted (flipped) orientation. Changing the value for *flag* does not affect the orientation of the cached image.

**Availability**

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.6.

**See Also**

- [isFlipped](#) (page 75)
- [recache](#) (page 38)

**Related Sample Code**

ImageMap  
ImageMapExample  
PhotoSearch  
Sketch-112  
WebKitPluginStarter

**Declared In**

NSImage.h

**setScaleWhenResized:**

Sets whether different-sized image representations are scaled to fit the receiver's size. (Deprecated in Mac OS X v10.6. This method was related to caching behavior. In Mac OS X v10.6 and later image caching is no longer necessary and as a result there is no replacement necessary.)

```
- (void)setScaleWhenResized:(BOOL)flag
```

**Parameters**

*flag*

YES if image representations are scaled to fit; otherwise NO.

**Discussion**

Most images (especially those loaded from files and URLs) contain only a single image representation whose size is the same as the receiver. It is possible to add image representations using the [addRepresentation:](#) (page 21) or [addRepresentations:](#) (page 21) methods but doing so is rarely necessary because modern hardware is powerful enough to resize and scale images quickly. The only reason to consider creating new representations is if each representation contains a customized version of the image at a specific size. (TIFF images may also contain a thumbnail version of an image, which is stored using a separate image representation.) If you pass YES in the *flag* parameter, and subsequently send a [setSize:](#) (page 43) message to the receiver, all such image representations would be scaled to the same size. Scaling of bitmap images usually results in the interpolation of the bitmap data.

This method does not invalidate the caches of any of the receiver's image representations. The caches are not invalidated until you change the image size using a [setSize:](#) (page 43) message. Scaling affects only the cached offscreen data for a given image representation.

**Availability**

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.6.

**See Also**

- [scaleWhenResized](#) (page 76)

**Related Sample Code**

CocoaDragAndDrop  
CompositeLab  
FunkyOverlayWindow

## APPENDIX A

### Deprecated NSImage Methods

MyCustomColorPicker  
STUCAuthoringDeviceCocoaSample

#### **Declared In**

NSImage.h



# Document Revision History

---

This table describes the changes to *NSImage Class Reference*.

Date	Notes
2010-08-03	Updated deprecated methods to indicate replacements where available.
2010-01-22	Add images to reference for icons.
2009-05-29	Updated for Mac OS X v10.6. Delegate methods moved to <i>NSImageDelegate Protocol Reference</i> .
2009-01-06	Updated the description of the <code>alignmentRect</code> method.
2008-10-15	Updated descriptions of the <code>imageNamed:</code> and compositing methods.
2007-10-31	Updated for Mac OS X v10.5. Fixed a delegate method name.
	Documented constants representing standard system images.
2006-10-03	Updated descriptions for the <code>imageFileTypes</code> , <code>imageUnfilteredFileTypes</code> , <code>imagePasteboardTypes</code> , and <code>imageUnfilteredPasteboardTypes</code> methods.
	Updated the drawing routine parameter descriptions to reflect the use of <code>NSZeroRect</code> to specify the entire image.
2006-06-28	Added a link to an explanation of how bundles are searched in relation to the <code>imageNamed:</code> method.
2006-05-23	First publication of this content as a separate document.

## REVISION HISTORY

### Document Revision History