NSImage

Inherits From:	NSObject
Conforms To:	NSCoding, NSCopying NSObject (NSObject)
Declared In:	AppKit/NSImage.h

Class Description

An NSImage object contains an image that can be composited anywhere without first being drawn in any particular view. It manages the image by:

- Reading image data from the application bundle, from an NSPasteboard, or from an NSData object.
- Keeping multiple representations of the same image.
- Choosing the representation that's appropriate for a particular data type.
- Choosing the representation that's appropriate for any given display device.
- Caching the representations it uses by rendering them in off-screen windows.
- Optionally retaining the data used to draw the representations, so that they can be reproduced when needed.
- Compositing the image from the off-screen cache to where it's needed on-screen.
- Reproducing the image for the printer so that it matches what's displayed on-screen, yet is the best representation possible for the printed page.
- Automatically using any filtering services installed by the user to convert image data from unsupported formats to supported formats.

Defining an Image

An image can be created from various types of data:

- Encapsulated PostScript code (EPS)
- Bitmap data in Tag Image File Format (TIFF)
- Untagged (raw) bitmap data
- Other image data supported by an NSImageRep subclass registered with the NSImage class
- Data that can be filtered to a supported type by a user-installed filter service

If data is placed in a file (for example, in an application bundle), the NSImage object can access the data whenever it's needed to create the image. If data is read from an NSData object, the NSImage object may need to store the data itself.

Images can also be defined by the program, in two ways:

- By drawing the image in an off-screen window maintained by the NSImage object. In this case, the NSImage maintains only the cached image.
- By defining a method that can be used to draw the image when needed. This allows the NSImage to delegate responsibility for producing the image to some other object.

Image Representations

An NSImage object can keep more than one representation of an image. Multiple representations permit the image to be customized for the display device. For example, different hand-tuned TIFF images can be provided for monochrome and color screens, and an EPS representation or a custom method might be used for printing. All representations are versions of the same image.

An NSImage returns an NSArray of its representations in response to a **representations** message. Each representation is a kind of NSImageRep object:

NSEPSImageRep	An image that can be recreated from EPS data that's either stored by the object or at a known location in the file system.
NSBitmapImageRep	An image that can be recreated from bitmap or TIFF data.
NSCustomImageRep	An image that can be redrawn by a method defined in the application.
NSCachedImageRep	An image that has been rendered in an off-screen cache from data or instructions that are no longer available. The image in the cache provides the only data from which the image can be reproduced.

You can define other NSImageRep subclasses for objects that render images from other types of source data. To make these new subclasses available to an NSImage object, they need to be added to the NSImageRep class registry by invoking the **registerImageRepClass:** class method. NSImage determines the data types that each subclass can support by invoking its **imageUnfilteredFileTypes** and **imageUnfilteredPasteboardTypes** methods.

Choosing Representations

The NSImage object will choose the representation that best matches the rendering device. By default, the choice is made according to the following set of ordered rules. Each rule is applied in turn until the choice of representation is narrowed to one.

1. Choose a color representation for a color device, and a gray-scale representation for a monochrome device.
2. Choose a representation with a resolution that matches the resolution of the device, or if no representation matches, choose the one with the highest resolution.

   By default, any image representation with a resolution that's an integer multiple of the device resolution is considered to match. If more than one representation matches, the NSImage will choose the one that's closest to the device resolution. However, you can force resolution matches to be exact by passing NO to the **setMatchesOnMultipleResolution:** method.

   Rule 2 prefers TIFF and bitmap representations, which have a defined resolution, over EPS representations, which don't. However, you can use the **setUsesEPSOnResolutionMismatch:** method to have the NSImage choose an EPS representation in case a resolution match isn't possible.
3. If all else fails, choose the representation with a specified bits per sample that matches the depth of the device. If no representation matches, choose the one with the highest bits per sample.

By passing NO to the **setPrefersColorMatch:** method, you can have the NSImage try for a resolution match before a color match. This essentially inverts the first and second rules above.

If these rules fail to narrow the choice to a single representationÐfor example, if the NSImage has two color TIFF representations with the same resolution and depthÐthe one that will be chosen is system dependent.

Caching Representations

When first asked to composite the image, the NSImage object chooses the representation that's best for the destination display device, as outlined above. It renders the representation in an off-screen window on the same device, then composites it from this cache to the desired location. Subsequent requests to composite the image use the same cache. Representations aren't cached until they're needed for compositing.

When printing, the NSImage tries not to use the cached image. Instead, it attempts to render on the printerÐusing the appropriate image data, or a delegated methodÐthe best version of the image that it can. Only as a last resort will it image the cached bitmap.

Image Size

Before an NSImage can be used, the size of the image must be set, in units of the base coordinate system. If a representation is smaller or larger than the specified size, it can be scaled to fit.

If the size of the image hasn't already been set when the NSImage is provided with a representation, the size will be set from the data. The bounding box is used to determine the size of an NSEPSImageRep. The TIFF fields ªImageLengthº and ªImageWidthº are used to determine the size of an NSBitmapImageRep.

Coordinate Systems

Images have the horizontal and vertical orientation of the base coordinate system; they can't be rotated or flipped. When composited, an image maintains this orientation, no matter what coordinate system it's composited to. (The destination coordinate system is used only to determine the location of a composited image, not its size or orientation.)

It's possible to refer to portions of an image when compositing by specifying a rectangle in the image's coordinate system, which is identical to the base coordinate system, except that the origin is at the lower left corner of the image.

Named Images

An NSImage object can be identified either by its **id** or by a name. Assigning an NSImage a name adds it to a table kept by the class object; each name in the database identifies one and only one instance of the class. When you ask for an NSImage object by name (with the **imageNamed:** method), the class object returns the one from its database, which also includes all the system bitmaps provided by the Application Kit. If there's no object in the database for the specified name, the class object tries to create one by checking for a system bitmap of the same name, checking the name of the application's own image, and then checking for the image in the application's main bundle.

If a section or file matches the name, an NSImage is created from the data stored there. You can therefore create NSImage objects simply by including EPS or TIFF data for them within the executable file, or in files inside the application's file package.

Image Filtering Services

NSImage is designed to automatically take advantage of user-installed filter services for converting unsupported image file types to supported image file types. The class method **imageFileTypes** returns an array of all file types from which NSImage can create an instance of itself. This list includes all file types supported by registered subclasses of NSImageRep, and those types that can be converted to supported file types through a user-installed filter service.

Initializing a New NSImage Instance

- (id)**initByReferencingFile:**(NSString *)*filename*	Initializes the new NSImage from the data in *filename*. The file is assumed to persist and may be reread later if the NSImage is resized or otherwise modified.
- (id)**initWithContentsOfFile:**(NSString *)*filename*	Initializes the new NSImage from the data in *filename*.
- (id)**initWithData:**(NSData *)*data*	Initializes the new NSImage from *data*.
- (id)**initWithPasteboard:**(NSPasteboard *)*pasteboard*	Initializes the new NSImage with the data in *pasteboard*.
- (id)**initWithSize:**(NSSize)*aSize*	Initializes the new NSImage to the specified size.

Setting the Size of the Image

- (void)**setSize:**(NSSize)*aSize*	Sets the size of the image to *aSize* in base coordinates.
- (NSSize)**size**	Returns the size of the image.

Referring to Images by Name

+ (id)**imageNamed:**(NSString *)*name*	Returns the NSImage object having *name*. Searches the main bundle for the image if necessary.
- (BOOL)**setName:**(NSString *)*name*	Assigns *name* to be the receiver's name. Returns NO if *name* is already in use; otherwise, returns YES.
- (NSString *)**name**	Returns the receiver's name.

Specifying the Image

- (void)**addRepresentation:**(NSImageRep *)*imageRep*	Adds *imageRep* to the receiver's list of representations.
- (void)**addRepresentations:**(NSArray *)*imageRepArray*	Adds the *imageRep*s from *imageRepArray* to the receiver's list of representations.
- (void)**lockFocus**	Prepares for drawing in the best representation.

- (void)**lockFocusOnRepresentation:**(NSImageRep *)*imageRep*
  Prepares for drawing in *imageRep*.

- (void)**unlockFocus** — Balances a previous **lockFocus** or **lockFocusOnRepresentation:**.

Using the Image

- (void)**compositeToPoint:**(NSPoint)*aPoint* **operation:**(NSCompositingOperation)*op*
  Composites the image to *aPoint* using the operation *op*.

- (void)**compositeToPoint:**(NSPoint)*aPoint* **fromRect:**(NSRect)*aRect* **operation:**(NSCompositingOperation)*op*
  Composites the *aRect* portion of the image to *aPoint* using the operation *op*.

- (void)**dissolveToPoint:**(NSPoint)*aPoint* **fraction:**(float)*aFloat*
  Composites the image to *aPoint* using the **dissolve** operator. *aFloat* is a value from 0.0 to 1.0 that determines how much of the resulting composite comes from the receiver.

- (void)**dissolveToPoint:**(NSPoint)*aPoint* **fromRect:**(NSRect)*aRect* **fraction:**(float)*aFloat*
  Composites the *aRect* portion of the image to *aPoint* using the **dissolve** operator. *aFloat* is a value from 0.0 to 1.0 that determines how much of the resulting composite comes from the receiver.

Choosing Which Image Representation to Use

- (void)**setPrefersColorMatch:**(BOOL)*flag* — Determines whether color matches are preferred.

- (BOOL)**prefersColorMatch** — Returns whether color matches are preferred.

- (void)**setUsesEPSOnResolutionMismatch:**(BOOL)*flag*
  Sets whether to use EPS representations on mismatch.

- (BOOL)**usesEPSOnResolutionMismatch** — Returns whether to use EPS representations on mismatch.

- (void)**setMatchesOnMultipleResolution:**(BOOL)*flag*
  Sets whether resolution multiples match.

- (BOOL)**matchesOnMultipleResolution** — Returns whether resolution multiples match.

Getting the Representations

- (NSImageRep *)**bestRepresentationForDevice:**(NSDictionary *)*deviceDescription*	Returns the best representation for the device described by *deviceDescription*. If *deviceDescription* is **nil**, the current device is assumed. See **NSGraphics.h** for appropriate dictionary keys and values.
- (NSArray *)**representations**	Returns an array of all the representations.
- (void)**removeRepresentation:**(NSImageRep *)*imageRep*	Removes *imageRep* from the receiver's list of representations.

Determining How the Image is Stored

- (void)**setCachedSeparately:**(BOOL)*flag*	Sets whether representations are cached separately.
- (BOOL)**isCachedSeparately**	Returns whether representations are cached separately.
- (void)**setDataRetained:**(BOOL)*flag*	Sets whether image data is retained by the object after the image is cached.
- (BOOL)**isDataRetained**	Returns whether image data is retained.
- (void)**setCacheDepthMatchesImageDepth:**(BOOL)*flag*	Sets whether the default depth limit applies to caches.
- (BOOL)**cacheDepthMatchesImageDepth**	Returns whether the default depth limit applies to caches.

Determining How the Image is Drawn

- (BOOL)**isValid**	Returns YES to indicate that the receiver's image is valid.
- (void)**setScalesWhenResized:**(BOOL)*flag*	If flag is YES, representations are scaled to fit.
- (BOOL)**scalesWhenResized**	Returns whether representations are scaled to fit.
- (void)**setBackgroundColor:**(NSColor *)*aColor*	Sets the background color of the image to *aColor*.
- (NSColor *)**backgroundColor**	Returns the background color of the image.
- (BOOL)**drawRepresentation:**(NSImageRep *)*imageRep* **inRect:**(NSRect)*aRect*	Overridden to have *imageRep* draw the representation in *aRect*.

- (void)**recache**	Invalidates caches of all representations, so they will be redrawn.

Assigning a Delegate

- (void)**setDelegate:**(id)*anObject*	Makes *anObject* the delegate of the NSImage.
- (id)**delegate**	Returns the delegate of the NSImage.

Producing TIFF Data for the Image

- (NSData *)**TIFFRepresentation**	Returns a data object containing TIFF for all representations, using their default compressions.
- (NSData *)**TIFFRepresentationUsingCompression:**(NSTIFFCompression)*comp* **factor:**(float)*aFloat*	Returns a data object containing TIFF for all the representations.

Managing NSImageRep Subclasses

+ (NSArray *)**imageUnfilteredFileTypes**	Returns an array of file types recognized by the NSImage without filtering. This list comes from all registered NSImageReps.
+ (NSArray *)**imageUnfilteredPasteboardTypes**	Returns an array of pasteboard types recognized by the NSImage.

Testing Image Data Sources

+ (BOOL)**canInitWithPasteboard:**(NSPasteboard *)*pasteboard*	Returns YES if the receiver can create a representation from *pasteboard*; otherwise, returns NO.
+ (NSArray *)**imageFileTypes**	Returns an array of supported image data file types.
+ (NSArray *)**imagePasteboardTypes**	Returns an array of supported pasteboard types.

Methods Implemented by the Delegate

- (NSImage *)**imageDidNotDraw:**(id)*sender* **inRect:**(NSRect)*aRect*	Responds to message that *image* couldn't be composited into *aRect*.