Quartz Framework Reference

Graphics & Imaging > Quartz

2007-12-11

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

Simultaneously published in the United States and Canada.

Contents

Tables

Introduction

Framework	/System/Library/Frameworks/Quartz
Header file directories	/System/Library/Frameworks/Quartz/Quartz.framework/Headers
Declared in	IKFilterBrowserPanel.h IKFilterBrowserView.h IKFilterUI.h IKFilterUIView.h IKImageBrowserView.h IKImageEditPanel.h IKImageView.h IKPictureTaker.h IKSaveOptions.h IKSlideshow.h PDFAction.h PDFActionGoTo.h PDFActionNamed.h PDFActionRemoteGoTo.h PDFActionResetForm.h PDFActionURL.h PDFAnnotation.h PDFAnnotationButtonWidget.h PDFAnnotationChoiceWidget.h PDFAnnotationCircle.h PDFAnnotationFreeText.h PDFAnnotationInk.h PDFAnnotationLine.h PDFAnnotationLink.h PDFAnnotationMarkup.h PDFAnnotationPopup.h PDFAnnotationSquare.h PDFAnnotationStamp.h PDFAnnotationText.h PDFAnnotationTextWidget.h PDFBorder.h PDFDestination.h PDFDocument.h PDFOutline.h PDFPage.h PDFSelection.h PDFThumbnailView.h PDFView.h QCComposition.h QCCompositionLayer.h QCCompositionParameterView.h QCCompositionPickerPanel.h QCCompositionPickerView.h

QCCompositionRepository.h
QCPlugIn.h
QCPlugInViewController.h
QCRenderer.h
QCView.h

Companion guides

Quartz Composer Programming Guide
Quartz Composer Custom Patch Programming Guide
Image Kit Programming Guide
PDF Kit Programming Guide

This collection of documents provides the API reference for the Quartz framework; in particular, for Quartz Composer, Image Kit, and PDF Kit. The Quartz Composer API supports processing and rendering graphical data and allows developers to create custom patches for the Quartz Composer developer tool. Image Kit provides user interface support for browsing, editing, and saving images, showing slideshows, and browsing and previewing Core Image filters. PDF Kit is a technology that allows applications to display and manipulate PDF documents.

Classes

CIFilter Image Kit Additions

Inherits from	NSObject
Conforms to	NSCoding NSCopying NSObject (NSObject)
Framework	System/Library/Frameworks/Quartz.framework/ImageKit.framework
Availability	Available in Mac OS X v10.5 and later.
Declared in	IIKFilterUI.h
Companion guide	Core Image Programming Guide

Overview

This Image Kit addition to the `CIFilter` class, introduced in Mac OS X v10.5, consists of one method and a set of constants that generate a view with input parameter controls for a Core Image filter. Using this method, it is easier for applications to present a user interface for a filter than it was in Mac OS X v10.4. Then, applications could create a filter user interface only by analyzing the keys and key attributes of a filter and then writing the code to implement the user interface.

You use the `viewForUIConfiguration:excludedKeys:` method to request a view from Core Image. The view is a subclass of the `NSView` class so that you can insert it easily into any other view as a subview or into an `NSWindow` object as a content view. Core Image automatically generates the view for you unless you implement the `IKFilterCustomUIProvider` protocol, in which case calling `viewForUIConfiguration:excludedKeys:` causes Core Image to provide your custom view.

Tasks

Creating a View for a Filter

`- viewForUIConfiguration:excludedKeys:` (page 20)
Returns a filter view for the filter.

Instance Methods

viewForUIConfiguration:excludedKeys:

Returns a filter view for the filter.

```
-(IKFilterUIView*)viewForUIConfiguration:(NSDictionary*)inUIConfiguration
    excludedKeys:(NSArray*)inKeys;
```

Parameters

inUIConfiguration

A dictionary that contains values for the `IKUISizeFlavor` and `kCIUIParameterSet` keys. See "User Interface Options" (page 21) for the constants that you can provide as values for `IKUISizeFlavor`. For `kCIUIParameterSet` you can provide one of the following values: `kCIUISetBasic`, `kCIUISetIntermediate`, `kCIUISetAdvanced`, or `kCIUISetDevelopment`. When you request a user interface for a parameter set, all keys for that set and below are included. For example, the advanced set consists of all parameters in the basic, intermediate and advanced sets. The development set should contain parameters that are either experimental or for debugging purposes. You should use them only during the development of filters and client applications, and not in a shipping product.

inKeys

An array of the input keys for which you do *not* want to provide a user interface. Pass `nil` if you want all input keys to be represented in the user interface.

Return Value

An `IKFilterUIView` object. You should retain the view as long as you need it, but make sure to release it when you no longer need it as the view is retaining the filter.

Discussion

Calling this method to receive a view for a filter causes the `CIFilter` class to invoke the `provideViewForUIConfiguration:excludedKeys:` (page 397) method. If you override `provideViewForUIConfiguration:excludedKeys:` the user interface is created by your filter subclass . Otherwise, Core Image automatically generates the user interface based on the filter keys and attributes.

The algorithm used to lay out the controls for a filter operates in a manner similar to the Core Image Fun House application (`/Developer/Applications/Graphics Tools/`). Applications can retrieve a view whose control sizes complement the size of user interface elements already used in the application. It is also possible to choose which filter input parameters appear in the view. Consumer applications, for example, may want to show a small, basic set of input parameters whereas professional applications may want to provide access to all input parameters.

The controls in the view use bindings to set the values of the filter. See *Cocoa Bindings Programming Topics* if you are unfamiliar with bindings.

Availability

Available in Mac OS X v10.5 and later.

Declared In

`IKFilterUI.h`

Constants

User Interface Options

Keys or values for the size of the input parameter controls for a filter view.

```
NSString *IKUISizeFlavor;
NSString *IKUISizeMini;
NSString *IKUISizeSmall;
NSString *IKUISizeRegular;
NSString *IKUImaxSize;
NSString *IKUIFlavorAllowFallback;
```

Constants

`IKUISizeFlavor`

A key for the size of the controls in a filter view. The associated value can be `IKUISizeMini`, `IKUISizeSmall`, or `IKUISizeRegular`.

Available in Mac OS X v10.5 and later.

Declared in `IKFilterUI.h`.

`IKUISizeMini`

Controls whose size is mini, as defined by Interface Builder 2.5.

Available in Mac OS X v10.5 and later.

Declared in `IKFilterUI.h`.

`IKUISizeSmall`

Controls whose size is small, as defined by Interface Builder 2.5.

Available in Mac OS X v10.5 and later.

Declared in `IKFilterUI.h`.

`IKUISizeRegular`

Controls whose size is regular or normal, as defined by Interface Builder 2.5.

Available in Mac OS X v10.5 and later.

Declared in `IKFilterUI.h`.

`IKUImaxSize`

Controls whose dimensions are the maximum allowable for the filter view. A width or height of `0` indicates that that dimension of the view is not restricted. If the size requested is too small, the filter is expected to return a view as small as possible. It is up to the client to verify that the returned view fits into the context.

Available in Mac OS X v10.5 and later.

Declared in `IKFilterUI.h`.

`IKUIFlavorAllowFallback`

Substitute controls of another size. The associated value is a Boolean value. If the filter cannot provide a view for the requested size and a fallback is allowed, the filter can use controls of a different size.

Available in Mac OS X v10.5 and later.

Declared in `IKFilterUI.h`.

Declared In

`IKFilterUI.h`

IKFilterBrowserPanel Class Reference

Inherits from	NSPanel : NSWindow : NSResponder : NSObject
Conforms to	NSUserInterfaceValidations (NSWindow) NSAnimatablePropertyContainer (NSWindow) NSCoding (NSResponder) NSObject (NSObject)
Framework	System/Library/Frameworks/Quartz.framework/ImageKit.framework
Availability	Available in Mac OS X v10.5 and later.
Declared in	ImageKit/IKFilterBrowserPanel.h

Overview

The `IKFilterBrowserPanel` class provides a user interface that allows users to browse Core Image filters (`CIFilter`), to preview a filter, and to get additional information about the filter, such as its description.

An `IKFilterBrowserPanel` object can be displayed as:

- a separate panel, that is, a utility window that floats on top of document windows
- a modal dialog
- a sheet, that is, a dialog that is attached to its parent window and must be dismissed by the user
- a view that an application can insert into a custom user interface

An `IKFilterBrowserPanel` object can be configured through a style mask to use either the default or brushed metal look for windows. The size and number of visible controls are specified through an options dictionary. An `IKFilterBrowserPanel` object communicates selection changes through notifications.

The `IKFilterBrowserPanel` class allows the user to create filter collections that are stored with the `filterCollections` key in the `com.apple.CoreImageKit.plist` property list located in `~/Library/Preferences/`.

Tasks

Getting a Filter Name

`- filterName` (page 27)
Returns the name of the filter that is currently selected in the filter browser.

Displaying and Running the Panel

`- filterBrowserViewWithOptions:` (page 26)
Returns a view that contains a filter browser.

`- beginWithOptions:modelessDelegate:didEndSelector:contextInfo:` (page 26)
Displays the filter browser in a new utility window, unless the filter browser is already open.

`- beginSheetWithOptions:modalForWindow:modalDelegate:didEndSelector:contextInfo:` (page 25)
Displays the filter browser in a sheet—that is, a dialog that is attached to its parent window and must be dismissed by the user.

`- runModalWithOptions:` (page 28)
Displays the filter browser in a modal dialog that must be dismissed by the user but that is not attached to a window.

`- finish:` (page 27)
Closes a filter browser view.

Creating a Filter Browser Panel

`+ filterBrowserPanelWithStyleMask:` (page 24)
Creates a shared instance of the `IKFilterBrowserPanel` class.

Class Methods

filterBrowserPanelWithStyleMask:

Creates a shared instance of the `IKFilterBrowserPanel` class.

```
+ (id)filterBrowserPanelWithStyleMask:(unsigned int)styleMask;
```

Parameters

styleMask
A mask that specifies whether to use the default or brushed metal look for the window. You can select or deselect the `NSTexturedBackgroundWindowMask` style bit.

Return Value
The shared instance.

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

Declared In
`IKFilterBrowserPanel.h`

Instance Methods

beginSheetWithOptions:modalForWindow:modalDelegate:didEndSelector:contextInfo:

Displays the filter browser in a sheet—that is, a dialog that is attached to its parent window and must be dismissed by the user.

```
- (void)beginSheetWithOptions:(NSDictionary*)inOptions modalForWindow:(NSWindow
    *)docWindow modalDelegate:(id)modalDelegate didEndSelector:(SEL)didEndSelector
     contextInfo:(void *)contextInfo;
```

Parameters

inOptions
A dictionary of options that describe the configuration to use for the filter browser user interface. For the possible keys you can supply see "Filter Browser Option Keys" (page 28) and the constant `IKUISizeFlavor`.

modalForWindow
The parent window for the dialog.

modalDelegate
The object that will invoke the selector `didEndSelector` when the filter browser session terminates.

didEndSelector
The selector to invoke when the filter browser session terminates.

contextInfo
Any data that must be passed as an argument to the delegate through `didEndSelector` after the filter browser session terminates.

Discussion
When the filter browser session ends, `didEndSelector` is invoked on the modeless delegate, passing `contextInfo` as an argument. The selector `didEndSelector` must have the following signature:

```
- (void)openPanelDidEnd:(NSOpenPanel *)panel returnCode:(int)returnCode
contextInfo:(void *)contextInfo
```

The `returnCode` value passed to the selector is set to `NSOKButton` if the user validates, or to `NSCancelButton` if the user cancels.

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

See Also
- `beginWithOptions:modelessDelegate:didEndSelector:contextInfo:` (page 26)
- `runModalWithOptions` (page 28)

Declared In
IKFilterBrowserPanel.h

beginWithOptions:modelessDelegate:didEndSelector:contextInfo:

Displays the filter browser in a new utility window, unless the filter browser is already open.

```
- (void)beginWithOptions:(NSDictionary*)inOptions
    modelessDelegate:(id)modelessDelegate didEndSelector:(SEL)didEndSelector
    contextInfo:(void *)contextInfo;
```

Parameters

inOptions

A dictionary of options that describe the configuration to use for the filter browser user interface. For the possible keys you can supply see "Filter Browser Option Keys" (page 28) and the constant IKUISizeFlavor.

modelessDelegate

The object that will invoke the selector didEndSelector when the filter browser session terminates.

didEndSelector

The selector to invoke when the filter browser session terminates.

contextInfo

Any data that must be passed as an argument to the delegate through didEndSelector after the filter browser session terminates.

Discussion
When the filter browser session ends, didEndSelector is invoked on the modeless delegate, passing contextInfo as an argument. The selector didEndSelector must have the following signature:

```
- (void)openPanelDidEnd:(NSOpenPanel *)panel returnCode:(int)returnCode
contextInfo:(void *)contextInfo
```

The returnCode value passed to the selector is set to NSOKButton if the user validates, or to NSCancelButton if the user cancels.

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

See Also
- beginSheetWithOptions:modalForWindow:modalDelegate:didEndSelector:contextInfo: (page 25)
- runModalWithOptions (page 28)

Declared In
IKFilterBrowserPanel.h

filterBrowserViewWithOptions:

Returns a view that contains a filter browser.

```
- (IKFilterBrowserView*)filterBrowserViewWithOptions:(NSDictionary*)inOptions;
```

Parameters

inOptions

A dictionary of options that describe the configuration to use for the filter browser user interface. For the possible keys you can supply see "Filter Browser Option Keys" (page 28) and the constant `IKUISizeFlavor`.

Return Value
A filter browser view that is configured as specified.

Discussion
Use this method to add a view that contains the filter browser to your custom user interface. To dismiss the filter browser view, invoke the `finish` (page 27) method.

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

Declared In
`IKFilterBrowserPanel.h`

filterName

Returns the name of the filter that is currently selected in the filter browser.

```
- (NSString*)filterName;
```

Return Value
The name of the currently selected filter.

Discussion
Use this method in response to the notifications `IKFilterBrowserFilterSelectedNotification` (page 30) or `IKFilterBrowserFilterDoubleClickNotification` (page 30), or after the user makes a choice in a dialog.

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

Declared In
`IKFilterBrowserPanel.h`

finish:

Closes a filter browser view.

```
- (void)finish:(id)sender;
```

Parameters

sender

The object that invokes the action, such as an OK or Cancel button.

Discussion
Invoke this action when you want to dismiss the filter browser.

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

See Also
- `filterBrowserViewWithOptions` (page 26)

Declared In
`IKFilterBrowserPanel.h`

runModalWithOptions:

Displays the filter browser in a modal dialog that must be dismissed by the user but that is not attached to a window.

```
- (int)runModalWithOptions:(NSDictionary*)inOptions;
```

Parameters

inOptions

A dictionary of options that describe the configuration to use for the filter browser user interface. For the possible keys you can supply see "Filter Browser Option Keys" (page 28) and the constant `IKUISizeFlavor`.

Return Value
Either `NSOKButton` if the user validates, or `NSCancelButton` if the user cancels.

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

See Also
- `beginSheetWithOptions:modalForWindow:modalDelegate:didEndSelector:contextInfo:` (page 25)
- `beginWithOptions:modelessDelegate:didEndSelector:contextInfo:` (page 26)

Declared In
`IKFilterBrowserPanel.h`

Constants

Filter Browser Option Keys

Keys for filter browser options.

```
NSString *const IKFilterBrowserDefaultInputImage;
NSString *const IKFilterBrowserExcludeCategories;
NSString *const IKFilterBrowserExcludeFilters;
NSString *const IKFilterBrowserShowCategories;
NSString *const IKFilterBrowserShowPreview;
```

Constants

`IKFilterBrowserDefaultInputImage`

The key for the default input image. The associated value is the `CIImage` object to use as the default input image for the filter preview. Setting the image to `nil` causes Image Kit to use the image supplied by the framework. You can also set the input image and other parameters during the notification IKFilterBrowserWillPreviewFilterNotification (page 29).

Available in Mac OS X v10.5 and later.

Declared in `IKFilterBrowserPanel.h`.

`IKFilterBrowserExcludeCategories`

The key for excluding filter categories. The associated value is an `NSArray` object that lists the categories that you do *not* want to display in the filter browser.

Available in Mac OS X v10.5 and later.

Declared in `IKFilterBrowserPanel.h`.

`IKFilterBrowserExcludeFilters`

The key for excluding filters. The associated value is an `NSArray` object that lists the filters that you do *not* want to display in the filter browser.

Available in Mac OS X v10.5 and later.

Declared in `IKFilterBrowserPanel.h`.

`IKFilterBrowserShowCategories`

The key for showing categories. The associated value is a `BOOL` value that determines if the filter browser should show the category list.

Available in Mac OS X v10.5 and later.

Declared in `IKFilterBrowserPanel.h`.

`IKFilterBrowserShowPreview`

The associated value is a `BOOL` value that determines if the filter browser should provide a preview.

Available in Mac OS X v10.5 and later.

Declared in `IKFilterBrowserPanel.h`.

Declared In

`IKFilterBrowserPanel.h`

Notifications

IKFilterBrowserWillPreviewFilterNotification

Posted before showing a filter preview, allowing an application to set the parameters of a filter.

The selected filter is sent as the object in the notification.

Availability

Available in Mac OS X v10.5 and later.

Declared In
`IKFilterBrowserPanel.h`

IKFilterBrowserFilterSelectedNotification

Posted when the user clicks a filter name in the filter browser.

The name of the selected filter is sent as the object in the notification.

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

Declared In
`IKFilterBrowserPanel.h`

IKFilterBrowserFilterDoubleClickNotification

Posted when the user double-clicks a filter in the filter browser.

The name of the selected filter is send as the object in the notification.

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

Declared In
`IKFilterBrowserPanel.h`

IKFilterBrowserView Class Reference

Inherits from	NSView : NSResponder : NSObject
Conforms to	NSAnimatablePropertyContainer (NSView) NSCoding (NSResponder) NSObject (NSObject)
Framework	System/Library/Frameworks/Quartz.framework/ImageKit.framework
Availability	Available in Mac OS X v10.5 and later.
Declared in	ImageKit/IKFilterBrowserView.h

Overview

The `IKFilterBrowserView` class is used as a container for the elements of an `IKFilterBrowserPanel` object.

Tasks

Setting the Preview State

`- setPreviewState:` (page 32)
Sets the preview state.

Getting the Filter Name

`- filterName` (page 31)
Returns the name of the filter that is currently selected in the filter browser.

Instance Methods

filterName

Returns the name of the filter that is currently selected in the filter browser.

```
- (NSString*)filterName;
```

Return Value
The name of the currently selected filter.

Discussion
Use this method in response to the notifications `IKFilterBrowserFilterSelectedNotification` (page 30) or `IKFilterBrowserFilterDoubleClickNotification` (page 30), or after the user makes a choice in a dialog.

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

Declared In
`IKFilterBrowserView.h`

setPreviewState:

Sets the preview state.

```
- (void)setPreviewState:(BOOL)inState;
```

Parameters
inState
A state (`YES` or `NO`) that represents whether a preview is visible.

Discussion
Use this method to show and hide the preview programmatically.

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

Declared In
`IKFilterBrowserView.h`

IKFilterUIView Class Reference

Inherits from	NSView : NSResponder : NSObject
Conforms to	NSAnimatablePropertyContainer (NSView) NSCoding (NSResponder) NSObject (NSObject)
Framework	System/Library/Frameworks/Quartz.framework/ImageKit.framework
Availability	Available in Mac OS X v10.5 and later.
Declared in	ImageKit/IKFilterUIView.h

Overview

The `IKFilterUIView` class provides a view that contains input parameter controls for a Core Image filter (`CIFilter`). You need to use this class when providing a user interface for a custom filter. The class creates a view that has an object controller for the given filter. It also retains the filter.

Tasks

Creating and Initializing a Filter UI View

`+ viewWithFrame:filter:` (page 34)
Creates a view that contains controls for the input parameters of a filter.

`- initWithFrame:filter:` (page 34)
Initializes a view that contains controls for the input parameters of a filter.

Getting Data from the Filter View

`- filter` (page 34)
Returns the Core Image filter associated with the view.

`- objectController` (page 35)
Returns the object controller for the bindings between the filter and its view.

Class Methods

viewWithFrame:filter:

Creates a view that contains controls for the input parameters of a filter.

```
+ (id)viewWithFrame:(NSRect)frameRect filter:(CIFilter *)inFilter
```

Parameters

frameRect

The rectangle that defines the area of the view.

inFilter

A Core Image filter. The view retains the filter.

Return Value

An `IKFilterUIView` object that contains controls for the input parameters of the provided filter.

Availability

Available in Mac OS X v10.5 and later.

See Also

`- initWithFrame:filter:` (page 34)

Declared In

`IKFilterUIView.h`

Instance Methods

filter

Returns the Core Image filter associated with the view.

```
- (CIFilter *)filter
```

Return Value

The Core Image filter associated with the view.

Availability

Available in Mac OS X v10.5 and later.

Declared In

`IKFilterUIView.h`

initWithFrame:filter:

Initializes a view that contains controls for the input parameters of a filter.

```
- (id)initWithFrame:(NSRect)frameRect filter:(CIFilter *)inFilter
```

Parameters

frameRect

The rectangle that defines the area of the view.

inFilter

A Core Image filter. The view retains the filter.

Return Value

The `IKFilterUIView` object initialized with controls for the input parameters of the provided filter.

Availability

Available in Mac OS X v10.5 and later.

See Also

`+ viewWithFrame:filter:` (page 34)

Declared In

`IKFilterUIView.h`

objectController

Returns the object controller for the bindings between the filter and its view.

```
- (NSObjectController *)objectController
```

Return Value

The object controller for the bindings between the filter and its view.

Availability

Available in Mac OS X v10.5 and later.

Declared In

`IKFilterUIView.h`

IKImageBrowserView Class Reference

Inherits from	NSView : NSResponder : NSObject
Conforms to	NSAnimatablePropertyContainer (NSView) NSCoding (NSResponder) NSObject (NSObject)
Framework	/System/Library/Frameworks/Quartz.framework/ImageKit.framework
Availability	Available in Mac OS X v10.5 and later.
Declared in	ImageKit/IKImageBrowserView.h

Overview

The `IKImageBrowserView` class is a view for displaying and browsing a large amount of images and movies efficiently.

Tasks

Initializing and Setting Up an Image Browser View

- `initWithFrame:` (page 45)
  Initializes a newly allocated image browser view with the provided frame rectangle.
- `setDataSource:` (page 50)
  Sets the data source of the receiver.
- `dataSource` (page 43)
  Returns the data source of the receiver.
- `reloadData` (page 46)
  Marks the receiver as needing its data reloaded.
- `setDelegate:` (page 50)
  Sets the delegate of the receiver.
- `delegate` (page 43)
  Returns the delegate of the receiver.

Setting the Appearance

- `setCellsStyleMask:` (page 49)
  Defines the appearance style of the cells.
- `cellsStyleMask` (page 41)
  Returns the appearance style mask for the cell.
- `setConstrainsToOriginalSize:` (page 49)
  Sets whether the receiver constrains the cell's image to its original size.
- `constrainsToOriginalSize` (page 42)
  Returns whether the receiver constrains the cell's image to its original size.

Zooming and Resizing

- `setZoomValue:` (page 51)
  Sets the zoom value.
- `zoomValue` (page 52)
  Returns the current zoom value.
- `setContentResizingMask:` (page 49)
  Determines how the receiver resizes its content when zooming.
- `contentResizingMask` (page 42)
  Returns the receiver's content resizing mask, which determines how its content is resized while zooming.

Scrolling

- `scrollIndexToVisible:` (page 46)
  Scrolls the receiver to the item at the specified index.

Setting and Getting Cell Size

- `setCellSize:` (page 48)
  Sets the cell size.
- `cellSize` (page 41)
  Returns the cell size.

Getting Item Information

- `indexOfItemAtPoint:` (page 45)
  Returns the index of the item at the specified location.
- `itemFrameAtIndex:` (page 46)
  Returns the frame rectangle for the item located at the specified index.

Reordering and Groups Items

- `selectionIndexes` (page 47)
  Returns the indexes of the selected cells.
- `setSelectionIndexes:byExtendingSelection:` (page 51)
  Selects cells at the specified indexes.
- `setAllowsMultipleSelection:` (page 47)
  Controls whether the user can select more than one cell at a time.
- `allowsMultipleSelection` (page 40)
  Returns whether multiple selections are allowed.
- `setAllowsEmptySelection:` (page 47)
  Controls whether an empty selection is allowed.
- `allowsEmptySelection` (page 40)
  Returns whether an empty selection is allowed.
- `setAllowsReordering:` (page 48)
  Controls whether the user can reorder items.
- `allowsReordering` (page 40)
  Returns whether the user can reorder items.
- `setAnimates:` (page 48)
  Controls whether the receiver animates reordering and changes of the data source.
- `animates` (page 41)
  Returns whether the receiver animates reordering and changes of the data source.
- `expandGroupAtIndex:` (page 44)
  Expands a group at the specified index.
- `collapseGroupAtIndex:` (page 42)
  Collapses a group at the specified index.
- `isGroupExpandedAtIndex:` (page 45)
  Returns whether the group at the provided index is expanded.

Supporting Drag and Drop

- `setDraggingDestinationDelegate:` (page 50)
  Sets the dragging destination delegate of the receiver.
- `draggingDestinationDelegate` (page 43)
  Returns the dragging destination delegate of the receiver.
- `indexAtLocationOfDroppedItem` (page 44)
  Returns the index of the cell where the drop operation occurred.

Instance Methods

allowsEmptySelection

Returns whether an empty selection is allowed.

```
- (BOOL) allowsEmptySelection;
```

Return Value
`YES` if the receiver allows an empty selection; `NO` otherwise.

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

See Also
`- setAllowsEmptySelection:` (page 47)

Declared In
`IKImageBrowserView.h`

allowsMultipleSelection

Returns whether multiple selections are allowed.

```
- (BOOL) allowsMultipleSelection;
```

Return Value
`YES` if the receiver allows the user to select more than one cell at a time; `NO` otherwise.

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

See Also
`- setAllowsEmptySelection:` (page 47)

Declared In
`IKImageBrowserView.h`

allowsReordering

Returns whether the user can reorder items.

```
- (BOOL) allowsReordering;
```

Return Value
`YES` if the user can reorder items; `NO` otherwise.

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

See Also
- setAllowsReordering: (page 48)

Declared In
IKImageBrowserView.h

animates

Returns whether the receiver animates reordering and changes of the data source.

```
- (BOOL) animates;
```

Return Value
YES if the receiver animates reordering and changes of the data source; NO otherwise.

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

See Also
- setAnimates: (page 48)

Declared In
IKImageBrowserView.h

cellSize

Returns the cell size.

```
- (NSSize) cellSize;
```

Return Value
The current size for the cells in the image browser view.

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

Declared In
IKImageBrowserView.h

cellsStyleMask

Returns the appearance style mask for the cell.

```
- (NSUInteger) cellsStyleMask;
```

Return Value
The appearance style mask for the cell.

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

See Also
- `setCellsStyleMask:` (page 49)

Declared In
`IKImageBrowserView.h`

collapseGroupAtIndex:

Collapses a group at the specified index.

```
- (void) collapseGroupAtIndex:(NSUInteger) index;
```

Parameters

index
The index of the group you want to collapse.

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

See Also
- `expandGroupAtIndex:` (page 44)
- `isGroupExpandedAtIndex:` (page 45)

Declared In
`IKImageBrowserView.h`

constrainsToOriginalSize

Returns whether the receiver constrains the cell's image to its original size.

```
- (BOOL) constrainsToOriginalSize;
```

Return Value
`NO` if the image is not constrained; otherwise `YES`.

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

See Also
- `setConstrainsToOriginalSize:` (page 49)

Declared In
`IKImageBrowserView.h`

contentResizingMask

Returns the receiver's content resizing mask, which determines how its content is resized while zooming.

```
- (NSUInteger) contentResizingMask;
```

Return Value
The content resizing mask.

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

See Also
`- setContentResizingMask:` (page 49)

Declared In
`IKImageBrowserView.h`

dataSource

Returns the data source of the receiver.

```
- (id ) dataSource;
```

Return Value
The data source (`IKImageBrowserDataSource`). The data source is not retained by the receiver.

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

See Also
`- setDataSource:` (page 50)

Declared In
`IKImageBrowserView.h`

delegate

Returns the delegate of the receiver.

```
- (id) delegate;
```

Return Value
The delegate.

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

See Also
`- setDelegate:` (page 50)

Declared In
`IKImageBrowserView.h`

draggingDestinationDelegate

Returns the dragging destination delegate of the receiver.

```
- (id) draggingDestinationDelegate;
```

Return Value
The receiver's dragging destination delegate.

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

See Also
- `setDraggingDestinationDelegate:` (page 50)

Declared In
`IKImageBrowserView.h`

expandGroupAtIndex:

Expands a group at the specified index.

```
- (void) expandGroupAtIndex:(NSUInteger) index;
```

Parameters
index
The index of the group you want to expand.

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

See Also
- `collapseGroupAtIndex:` (page 42)
- `isGroupExpandedAtIndex:` (page 45)

Declared In
`IKImageBrowserView.h`

indexAtLocationOfDroppedItem

Returns the index of the cell where the drop operation occurred.

```
- (NSUInteger) indexAtLocationOfDroppedItem;
```

Return Value
The index of the cell where the drop operation occurred.

Discussion
The returned index is valid until the next drop occurs.

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

Declared In
`IKImageBrowserView.h`

indexOfItemAtPoint:

Returns the index of the item at the specified location.

```
- (NSInteger) indexOfItemAtPoint: (NSPoint)point;
```

Parameters

point

The location of the item.

Return Value

The index of the item or `NSNotFound` if no item at this location.

Availability

Available in Mac OS X v10.5 and later.

Declared In

`IKImageBrowserView.h`

initWithFrame:

Initializes a newly allocated image browser view with the provided frame rectangle.

```
- (id) initWithFrame:(NSRect) frame;
```

Parameters

frame

The rectangle for the image browser.

Return Value

The initialized object.

Availability

Available in Mac OS X v10.5 and later.

Declared In

`IKImageBrowserView.h`

isGroupExpandedAtIndex:

Returns whether the group at the provided index is expanded.

```
- (BOOL) isGroupExpandedAtIndex:(NSUInteger) index;
```

Parameters

index

The index you want to check.

Return Value

`YES` if the group is expanded; `NO` otherwise.

Availability

Available in Mac OS X v10.5 and later.

See Also
- `expandGroupAtIndex:` (page 44)
- `collapseGroupAtIndex:` (page 42)

Declared In
`IKImageBrowserView.h`

itemFrameAtIndex:

Returns the frame rectangle for the item located at the specified index.

```
- (NSRect) itemFrameAtIndex: (NSInteger)index;
```

Parameters
index
The index of the item whose frame rectangle you want to obtain.

Return Value
The frame rectangle of the item.

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

Declared In
`IKImageBrowserView.h`

reloadData

Marks the receiver as needing its data reloaded.

```
- (void) reloadData;
```

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

Declared In
`IKImageBrowserView.h`

scrollIndexToVisible:

Scrolls the receiver to the item at the specified index.

```
- (void) scrollIndexToVisible:(NSInteger) index;
```

Parameters
index
The index of the item to scroll to.

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

Declared In
`IKImageBrowserView.h`

selectionIndexes

Returns the indexes of the selected cells.

```
- (NSIndexSet *) selectionIndexes;
```

Return Value
The indexes of the selected cells.

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

See Also
`- setSelectionIndexes:byExtendingSelection:` (page 51)

Declared In
`IKImageBrowserView.h`

setAllowsEmptySelection:

Controls whether an empty selection is allowed.

```
- (void) setAllowsEmptySelection: (BOOL) flag;
```

Parameters
flag
A `BOOL` value that specifies whether to allow an empty selection.

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

See Also
`- allowsEmptySelection` (page 40)

Declared In
`IKImageBrowserView.h`

setAllowsMultipleSelection:

Controls whether the user can select more than one cell at a time.

```
- (void) setAllowsMultipleSelection: (BOOL) flag;
```

Parameters
flag
A `BOOL` value that specifies whether to allow multiple selections.

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

See Also
`- allowsMultipleSelection` (page 40)

Declared In
IKImageBrowserView.h

setAllowsReordering:

Controls whether the user can reorder items.

```
- (void) setAllowsReordering: (BOOL) flag;
```

Parameters

flag

A BOOL value that specifies whether the user can reorder items.

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

See Also
- allowsReordering (page 40)

Declared In
IKImageBrowserView.h

setAnimates:

Controls whether the receiver animates reordering and changes of the data source.

```
- (void) setAnimates: (BOOL) flag;
```

Parameters

flag

A BOOL value that specifies whether the receiver animates reordering and changes of the data source.

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

See Also
- animates (page 41)

Declared In
IKImageBrowserView.h

setCellSize:

Sets the cell size.

```
- (void) setCellSize:(NSSize) size;
```

Parameters

size

The size to set.

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

Declared In
IKImageBrowserView.h

setCellsStyleMask:

Defines the appearance style of the cells.

```
- (void)  setCellsStyleMask:(NSUInteger) mask;
```

Parameters

mask

An integer bit mask. A mask can be specified by combining any of the options described in "Cell Appearance Style Masks" (page 52) using the C bitwise OR operator.

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

See Also
- cellsStyleMask (page 41)

Declared In
IKImageBrowserView.h

setConstrainsToOriginalSize:

Sets whether the receiver constrains the cell's image to its original size.

```
- (void) setConstrainsToOriginalSize: (BOOL) flag;
```

Parameters

flag

A flag that specifies whether to constrain the image. The default value is NO.

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

See Also
- constrainsToOriginalSize (page 42)

Declared In
IKImageBrowserView.h

setContentResizingMask:

Determines how the receiver resizes its content when zooming.

```
- (void) setContentResizingMask:(NSUInteger) mask;
```

Parameters

mask

A resizing mask. You specify a mask by combining any of the following options using the C bitwise OR operator: NSViewWidthSizable, NSViewHeightSizable. Other values are ignored.

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

See Also
- `contentResizingMask` (page 42)

Declared In
`IKImageBrowserView.h`

setDataSource:

Sets the data source of the receiver.

```
- (void) setDataSource:(id) source;
```

Parameters

source
A data source (`IKImageBrowserDataSource`).

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

See Also
- `dataSource` (page 43)

Declared In
`IKImageBrowserView.h`

setDelegate:

Sets the delegate of the receiver.

```
- (void) setDelegate: (id) aDelegate;
```

Parameters

aDelegate
The delegate must implement the `IKImageBrowserDelegate` informal protocol.

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

See Also
- `delegate` (page 43)

Declared In
`IKImageBrowserView.h`

setDraggingDestinationDelegate:

Sets the dragging destination delegate of the receiver.

```
- (void) setDraggingDestinationDelegate:(id) delegate;
```

Parameters

delegate

The delegate (`NSDraggingDestination`) to set.

Availability

Available in Mac OS X v10.5 and later.

See Also

`- draggingDestinationDelegate` (page 43)

Declared In

`IKImageBrowserView.h`

setSelectionIndexes:byExtendingSelection:

Selects cells at the specified indexes.

```
- (void) setSelectionIndexes:(NSIndexSet *) indexes byExtendingSelection:(BOOL)
    extendSelection;
```

Parameters

indexes

The indexes of the cells you want to select.

extendSelection

A `BOOL` value that specifies whether to extend the current selection. Pass `YES` to extends the selection; `NO` replaces the current selection.

Availability

Available in Mac OS X v10.5 and later.

See Also

`- selectionIndexes` (page 47)

Declared In

`IKImageBrowserView.h`

setZoomValue:

Sets the zoom value.

```
- (void) setZoomValue:(float)aValue;
```

Parameters

aValue

The zoom value. This value should be greater or equal to zero and less or equal than one. A zoom value of zero corresponds to the minimum size (40x40 pixels). A zoom value of one means images fits the browser bounds. Other values are interpolated.

Availability

Available in Mac OS X v10.5 and later.

See Also

`- zoomValue` (page 52)

Declared In
IKImageBrowserView.h

zoomValue

Returns the current zoom value.

```
- (float) zoomValue;
```

Return Value
The zoom value.

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

See Also
- setZoomValue: (page 51)

Declared In
IKImageBrowserView.h

Constants

Cell Appearance Style Masks

Masks for the appearance style bit field.

```
enum{
    IKCellsStyleNone              =0,
    IKCellsStyleShadowed          =1,
    IKCellsStyleOutlined          =2,
    IKCellsStyleTitled            =4,
    IKCellsStyleSubtitled         =8
};
```

Constants

IKCellsStyleNone
No style.
Available in Mac OS X v10.5 and later.
Declared in IKImageBrowserView.h.

IKCellsStyleShadowed
Cells use shadows.
Available in Mac OS X v10.5 and later.
Declared in IKImageBrowserView.h.

IKCellsStyleOutlined
Cells are outlined.
Available in Mac OS X v10.5 and later.
Declared in IKImageBrowserView.h.

`IKCellsStyleTitled`
Cells display a title.

Available in Mac OS X v10.5 and later.

Declared in `IKImageBrowserView.h`.

`IKCellsStyleSubtitled`
Cells display a subtitle.

Available in Mac OS X v10.5 and later.

Declared in `IKImageBrowserView.h`.

Declared In
`IKImageBrowserView.h`

Group Style Attributes

Attributes for the group style.

```
enum{
    IKGroupBezelStyle,
    IKGroupDisclosureStyle,
};
```

Constants
`IKGroupBezelStyle`
A bezel style.

Available in Mac OS X v10.5 and later.

Declared in `IKImageBrowserView.h`.

`IKGroupDisclosureStyle`
A disclosure triangle.

Available in Mac OS X v10.5 and later.

Declared in `IKImageBrowserView.h`.

Discussion
These constants affect the appearance of a group.

Declared In
`IKImageBrowserView.h`

View Options Keys

Keys for image browser view options.

```
NSString * const IKImageBrowserBackgroundColorKey;
NSString * const IKImageBrowserSelectionColorKey;
NSString * const IKImageBrowserCellsOutlineColorKey;
NSString * const IKImageBrowserCellsTitleAttributesKey;
NSString * const IKImageBrowserCellsHighlightedTitleAttributesKey;
NSString * const IKImageBrowserCellsSubtitleAttributesKey;
```

Constants

`IKImageBrowserBackgroundColorKey`
A key for the background color of the image browser view. The associated value is an `NSColor` object.

Available in Mac OS X v10.5 and later.

Declared in `IKImageBrowserView.h`.

`IKImageBrowserSelectionColorKey`
A key for the color that indicates a selection. The associated value is an `NSColor` object.

Available in Mac OS X v10.5 and later.

Declared in `IKImageBrowserView.h`.

`IKImageBrowserCellsOutlineColorKey`
A key for the outline color for an item in the image browser view. The associated value is an `NSColor` object.

Available in Mac OS X v10.5 and later.

Declared in `IKImageBrowserView.h`.

`IKImageBrowserCellsTitleAttributesKey`
A key for title attribute of an item in the image browser view. The associated value is an `NSDictionary` object.

Available in Mac OS X v10.5 and later.

Declared in `IKImageBrowserView.h`.

`IKImageBrowserCellsHighlightedTitleAttributesKey`
A key for the highlighted title attribute for an item in the image browser view. The associated value is an `NSDictionary` object.

Available in Mac OS X v10.5 and later.

Declared in `IKImageBrowserView.h`.

`IKImageBrowserCellsSubtitleAttributesKey`
A key for a subtitle attribute for an item in the image browser view. The associated value is an `NSDictionary` object.

Available in Mac OS X v10.5 and later.

Declared in `IKImageBrowserView.h`.

Discussion
You can set and retrieve values for these keys using the methods `setValue:forKey` and `valueForKey:`.

Declared In
`IKImageBrowserView.h`

Group Keys

Keys for group attributes.

```
NSString * const IKImageBrowserGroupRangeKey;
NSString * const IKImageBrowserGroupBackgroundColorKey;
NSString * const IKImageBrowserGroupTitleKey;
NSString * const IKImageBrowserGroupStyleKey;
```

Constants

`IKImageBrowserGroupRangeKey`

A key for the range of a group. The associated value is an `NSValue` object.

Available in Mac OS X v10.5 and later.

Declared in `IKImageBrowserView.h`.

`IKImageBrowserGroupBackgroundColorKey`

A key for the background color of a group. The associated value is an `NSColor` object.

Available in Mac OS X v10.5 and later.

Declared in `IKImageBrowserView.h`.

`IKImageBrowserGroupTitleKey`

A key for the title of a group. The associated value is an `NSString` object.

Available in Mac OS X v10.5 and later.

Declared in `IKImageBrowserView.h`.

`IKImageBrowserGroupStyleKey`

A key for the style of a group. The associated value is one of the constants defined in "Group Style Attributes" (page 53).

Available in Mac OS X v10.5 and later.

Declared in `IKImageBrowserView.h`.

Declared In

`IKImageBrowserView.h`

IKImageEditPanel Class Reference

Inherits from	NSPanel : NSWindow : NSResponder : NSObject
Conforms to	NSUserInterfaceValidations (NSWindow) NSAnimatablePropertyContainer (NSWindow) NSCoding (NSResponder) NSObject (NSObject)
Framework	System/Library/Frameworks/Quartz.framework/ImageKit.framework
Availability	Available in Mac OS X v10.5 and later.
Declared in	ImageKit/IKImageEditPanel.h

Overview

The `IKImageEditPanel` class provides a panel, that is, a utility window that floats on top of document windows, optimized for image editing.

Tasks

Creating an Image Editing Panel

`+ sharedImageEditPanel` (page 58)
Creates a shared instance of an image editing panel.

Getting, Setting, and Reloading Data

`- setDataSource:` (page 59)
Sets a data source for an image editing panel.

`- dataSource` (page 58)
Returns the data source associated with an image editing panel.

`- reloadData` (page 58)
Reloads the data from the data associated with an image editing panel.

Class Methods

sharedImageEditPanel

Creates a shared instance of an image editing panel.

```
+ (IKImageEditPanel*) sharedImageEditPanel;
```

Return Value
An `IKImageEditPanel` object.

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

Declared In
`IKImageEditPanel.h`

Instance Methods

dataSource

Returns the data source associated with an image editing panel.

```
- (id)dataSource;
```

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

See Also
`- setDataSource:` (page 59)

Declared In
`IKImageEditPanel.h`

reloadData

Reloads the data from the data associated with an image editing panel.

```
- (void)reloadData;
```

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

Declared In
`IKImageEditPanel.h`

setDataSource:

Sets a data source for an image editing panel.

```
- (void)setDataSource: (id<IKImageEditPanelDataSource>)dataSource;
```

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

See Also
- dataSource (page 58)

Declared In
IKImageEditPanel.h

IKImageView Class Reference

Inherits from	NSView : NSResponder : NSObject
Conforms to	NSAnimatablePropertyContainer (NSView) NSCoding (NSResponder) NSObject (NSObject)
Framework	System/Library/Frameworks/Quartz.framework/ImageKit.framework
Availability	Available in Mac OS X v10.5 and later.
Declared in	ImageKit/IKImageView.h

Overview

The `IKImageView` class provides an efficient way to display images in a view while at the same time supporting a number of image editing operations such as rotating, zooming, and cropping. It supports drag and drop, so that the user can drag an image to the view. If possible, image rendering uses hardware acceleration to achieve optimal performance. The `IKImageView` class is implemented as a subclass of `NSView`. Similar to `NSImageView`, the `IKImageView` class is used to display a single image.

You can provide an images for the view in any of these formats:

- File reference (`NSURL`, `CFURLRef`, or a path)
- `CGImageSourceRef`
- Data (`NSData` or `CFDataRef`)
- Image (`NSImage`, `CGImageRef`, or `CIImage`)

Providing a file reference is the preferred way to set the the image for a view because in addition to the actual image data, `IKImageView` also handles the image metadata embedded in the file. The image view automatically fetches the metadata from a file reference, whereas for the other sources (except for a `CGImageSourceRef` source), it cannot. For images set from other sources, you need to set the metadata separately.

`IKImageView` supports multi-frame images (TIFF, GIF, and so forth) and animated images.

Tasks

Getting and Setting Image View Characteristics

`delegate` (page 65) *property*
Specifies the delegate object of the receiver.

`zoomFactor` (page 67) *property*
Specifies the zoom factor for the image view.

`rotationAngle` (page 66) *property*
Specifies the rotation angle for the image view.

`currentToolMode` (page 64) *property*
Specifies the current tool mode for the image view.

`autoresizes` (page 64) *property*
Specifies the automatic resizing state for the image view.

`hasHorizontalScroller` (page 65) *property*
Specifies the horizontal scroll bar state for the image view.

`hasVerticalScroller` (page 66) *property*
Specifies the vertical scroll bar state for the image view.

`autohidesScrollers` (page 64) *property*
Specifies the automatic-hiding scroll bar state for the image view.

`supportsDragAndDrop` (page 66) *property*
Specifies the drag-and-drop support state for the image view.

`editable` (page 65) *property*
Specifies the editable state for the image view.

`doubleClickOpensImageEditPanel` (page 65) *property*
Specifies the image-opening state of the editing pane in the image view.

`imageCorrection` (page 66) *property*
Specifies a Core Image filter for image correction.

`backgroundColor` (page 64) *property*
Specifies the background color for the image view.

- `imageSize` (page 70)
Returns the size of the image in the image view.

- `imageProperties` (page 70)
Returns the metadata for the image in the view.

Getting and Setting Images

- `image` (page 69)
Returns the image associated with the view, after any image corrections.

- `setImage:imageProperties:` (page 72)
Sets the image to display in an image view.

- `setImageWithURL:` (page 72)
  Initializes an image view with the image specified by a URL.

Manipulating the Image in a View

- `setRotationAngle:centerPoint:` (page 73)
  Sets the rotation angle at the provided origin.
- `setImageZoomFactor:centerPoint:` (page 72)
  Sets the zoom factor at the provided origin.
- `zoomImageToFit:` (page 74)
  Zooms the image so that it fits in the image view.
- `zoomImageToActualSize:` (page 74)
  Zooms the image so that it is displayed using its true size.
- `zoomImageToRect:` (page 74)
  Zooms the image so that it fits in the specified rectangle.
- `flipImageHorizontal:` (page 69)
  Flips an image along the horizontal axis.
- `flipImageVertical:` (page 69)
  Flips an image along the vertical axis.

Working With Core Animation

- `setOverlay:forType:` (page 73)
  Sets an overlay type for a Core Animation layer.
- `overlayForType:` (page 70)
  Returns the Core Animation layer associated with a layer type.

Scrolling

- `scrollToPoint:` (page 71)
  Scrolls the view to the specified point.
- `scrollToRect:` (page 71)
  Scrolls the view so that it includes the provided rectangular area.

Converting Points and Rectangles

- `convertViewPointToImagePoint:` (page 68)
  Converts an image view coordinate to an image coordinate.
- `convertViewRectToImageRect:` (page 68)
  Converts an image view rectangle to an image rectangle.
- `convertImagePointToViewPoint:` (page 67)
  Converts an image coordinate to an image view coordinate.

`- convertImageRectToViewRect:` (page 67)
Converts an image rectangle to an image view rectangle.

Properties

For more about Objective-C properties, see "Properties" in *The Objective-C 2.0 Programming Language*.

autohidesScrollers

Specifies the automatic-hiding scroll bar state for the image view.

```
@property BOOL autohidesScrollers;
```

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

Declared In
`IKImageView.h`

autoresizes

Specifies the automatic resizing state for the image view.

```
@property BOOL autoresizes;
```

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

Declared In
`IKImageView.h`

backgroundColor

Specifies the background color for the image view.

```
@property NSColor *  backgroundColor;
```

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

Declared In
`IKImageView.h`

currentToolMode

Specifies the current tool mode for the image view.

```
@property NSString*  currentToolMode;
```

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

Declared In
`IKImageView.h`

delegate

Specifies the delegate object of the receiver.

```
@property id delegate;
```

Discussion
An IKImageView object's delegate is inserted in the responder chain after the image view itself and is informed of various actions by the image view through delegation messages.

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

Declared In
`IKImageView.h`

doubleClickOpensImageEditPanel

Specifies the image-opening state of the editing pane in the image view.

```
@property BOOL doubleClickOpensImageEditPanel;
```

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

Declared In
`IKImageView.h`

editable

Specifies the editable state for the image view.

```
@property BOOL editable;
```

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

Declared In
`IKImageView.h`

hasHorizontalScroller

Specifies the horizontal scroll bar state for the image view.

```
@property BOOL hasHorizontalScroller;
```

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

Declared In
`IKImageView.h`

hasVerticalScroller

Specifies the vertical scroll bar state for the image view.

```
@property BOOL hasVerticalScroller;
```

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

Declared In
`IKImageView.h`

imageCorrection

Specifies a Core Image filter for image correction.

```
@property CIFilter * imageCorrection;
```

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

Declared In
`IKImageView.h`

rotationAngle

Specifies the rotation angle for the image view.

```
@property CGFloat rotationAngle;
```

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

Declared In
`IKImageView.h`

supportsDragAndDrop

Specifies the drag-and-drop support state for the image view.

```
@property BOOL supportsDragAndDrop;
```

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

Declared In
`IKImageView.h`

zoomFactor

Specifies the zoom factor for the image view.

```
@property CGFloat zoomFactor;
```

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

Declared In
`IKImageView.h`

Instance Methods

convertImagePointToViewPoint:

Converts an image coordinate to an image view coordinate.

```
- (NSPoint)convertImagePointToViewPoint: (NSPoint)imagePoint;
```

Parameters
imagePoint
A point specified in coordinates relative to the image.

Return Value
A point specified in coordinates relative to the image view.

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

See Also
`- convertViewPointToImagePoint:` (page 68)

Declared In
`IKImageView.h`

convertImageRectToViewRect:

Converts an image rectangle to an image view rectangle.

```
- (NSRect)convertImageRectToViewRect: (NSRect)imageRect;
```

Parameters

imageRect

An rectangle specified in coordinates relative to the image.

Return Value

An rectangle specified in coordinates relative to the image view.

Availability

Available in Mac OS X v10.5 and later.

See Also

- convertViewRectToImageRect: (page 68)

Declared In

IKImageView.h

convertViewPointToImagePoint:

Converts an image view coordinate to an image coordinate.

```
- (NSPoint)convertViewPointToImagePoint: (NSPoint)viewPoint;
```

Parameters

viewPoint

A point specified in coordinates relative to the image view.

Return Value

The point specified in coordinates relative to the image.

Availability

Available in Mac OS X v10.5 and later.

See Also

- convertImagePointToViewPoint: (page 67)

Declared In

IKImageView.h

convertViewRectToImageRect:

Converts an image view rectangle to an image rectangle.

```
- (NSRect)convertViewRectToImageRect: (NSRect)viewRect;
```

Parameters

viewRect

An rectangle specified in coordinates relative to the image view.

Return Value

The rectangle specified in coordinates relative to the image.

Availability

Available in Mac OS X v10.5 and later.

See Also
- convertImageRectToViewRect: (page 67)

Declared In
IKImageView.h

flipImageHorizontal:

Flips an image along the horizontal axis.

```
- (void)flipImageHorizontal: (id)sender;
```

Parameters

sender
The object initiating the action.

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

See Also
- flipImageVertical: (page 69)

Declared In
IKImageView.h

flipImageVertical:

Flips an image along the vertical axis.

```
- (void)flipImageVertical: (id)sender;
```

Parameters

sender
The object initiating the action.

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

See Also
- flipImageHorizontal: (page 69)

Declared In
IKImageView.h

image

Returns the image associated with the view, after any image corrections.

```
- (CGImageRef)image;
```

Return Value
The image.

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

See Also
- `setImage:imageProperties:` (page 72)
- `setImageWithURL:` (page 72)

Declared In
`IKImageView.h`

imageProperties

Returns the metadata for the image in the view.

```
- (NSDictionary*)imageProperties;
```

Return Value
A dictionary of metadata that specifies the image properties.

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

Declared In
`IKImageView.h`

imageSize

Returns the size of the image in the image view.

```
- (NSSize)imageSize;
```

Return Value
The size of the image.

Discussion
The image size changes whenever an image is rotates or cropped.

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

Declared In
`IKImageView.h`

overlayForType:

Returns the Core Animation layer associated with a layer type.

```
- (CALayer*)overlayForType: (NSString*)layerType;
```

Parameters
layerType
A layer type. See "Overlay Types" (page 76).

Return Value
The Core Animation layer.

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

See Also
- `setOverlay:forType:` (page 73)

Declared In
`IKImageView.h`

scrollToPoint:

Scrolls the view to the specified point.

```
- (void)scrollToPoint:(NSPoint)point;
```

Parameters

point
The point to scroll to.

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

See Also
- `scrollToRect:` (page 71)

Declared In
`IKImageView.h`

scrollToRect:

Scrolls the view so that it includes the provided rectangular area.

```
- (void)scrollToRect:(NSRect)rect;
```

Parameters

rect
The rectangular area to include in the view.

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

See Also
- `scrollToPoint:` (page 71)

Declared In
`IKImageView.h`

setImage:imageProperties:

Sets the image to display in an image view.

```
- (void)setImage: (CGImageRef)image  imageProperties: (NSDictionary*)metaData;
```

Parameters

image

The image to set.

metaData

A dictionary that contains metadata that describes the image.

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

See Also
- `image` (page 69)
- `imageProperties` (page 70)
- `setImageWithURL:` (page 72)

Declared In
`IKImageView.h`

setImageWithURL:

Initializes an image view with the image specified by a URL.

```
- (void)setImageWithURL: (NSURL*)url;
```

Parameters

url

The URL that specifies the location of the image.

Discussion
This method is the preferred initializer for RAW images. If you use this method for a TIFF file that contains multiple images, only the first image is displayed.

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

See Also
- `setImage:imageProperties:` (page 72)

Declared In
`IKImageView.h`

setImageZoomFactor:centerPoint:

Sets the zoom factor at the provided origin.

```
- (void)setImageZoomFactor: (CGFloat)zoomFactor centerPoint: (NSPoint)centerPoint;
```

Parameters

zoomFactor

The zoom factor to apply to the image.

centerPoint

The point that specifies the origin of the zoom factor.

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

See Also
`@property zoomFactor` (page 67)

Declared In
`IKImageView.h`

setOverlay:forType:

Sets an overlay type for a Core Animation layer.

```
- (void)setOverlay: (CALayer*)layer forType: (NSString*)layerType;
```

Parameters

layer

A Core Animation layer object.

layerType

A layer type. See "Overlay Types" (page 76).

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

See Also
`- overlayForType:` (page 70)

Declared In
`IKImageView.h`

setRotationAngle:centerPoint:

Sets the rotation angle at the provided origin.

```
- (void)setRotationAngle: (CGFloat)rotationAngle  centerPoint: (NSPoint)centerPoint;
```

Parameters

rotationAngle

The rotation angle to apply to the image.

centerPoint

The point that specifies the origin of the rotation angle.

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

See Also
`@property rotationAngle` (page 66)

Declared In
`IKImageView.h`

zoomImageToActualSize:

Zooms the image so that it is displayed using its true size.

```
- (void)zoomImageToActualSize: (id)sender;
```

Parameters

sender
The object initiating the action.

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

See Also
- `zoomImageToFit:` (page 74)
- `zoomImageToRect:` (page 74)

Declared In
`IKImageView.h`

zoomImageToFit:

Zooms the image so that it fits in the image view.

```
- (void)zoomImageToFit: (id)sender;
```

Parameters

sender
The object initiating the action.

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

See Also
- `zoomImageToActualSize:` (page 74)
- `zoomImageToRect:` (page 74)

Declared In
`IKImageView.h`

zoomImageToRect:

Zooms the image so that it fits in the specified rectangle.

```
- (void)zoomImageToRect: (NSRect)rect;
```

Parameters

rect

The rectangle to fit the image in.

Availability

Available in Mac OS X v10.5 and later.

See Also

- `zoomImageToFit:` (page 74)
- `zoomImageToActualSize:` (page 74)

Declared In

`IKImageView.h`

Constants

Tool Modes

Image Kit tools modes.

```
NSString *const IKToolModeMove;
NSString *const IKToolModeSelect;
NSString *const IKToolModeCrop;
NSString *const IKToolModeRotate;
NSString *const IKToolModeAnnotate;
```

Constants

`IKToolModeMove`

The move tool.

Available in Mac OS X v10.5 and later.

Declared in `IKImageView.h`.

`IKToolModeSelect`

The selection tool.

Available in Mac OS X v10.5 and later.

Declared in `IKImageView.h`.

`IKToolModeCrop`

The crop tool.

Available in Mac OS X v10.5 and later.

Declared in `IKImageView.h`.

`IKToolModeRotate`

The rotation tool.

Available in Mac OS X v10.5 and later.

Declared in `IKImageView.h`.

`IKToolModeAnnotate`

The annotation tool.

Available in Mac OS X v10.5 and later.

Declared in `IKImageView.h`.

Declared In
`IKImageView.h`

Overlay Types

A layer level.

```
NSString *const IKOverlayTypeBackground;
NSString *const IKOverlayTypeImage;
```

Constants

`IKOverlayTypeBackground`

A background.

Available in Mac OS X v10.5 and later.

Declared in `IKImageView.h`.

`IKOverlayTypeImage`

An image.

Available in Mac OS X v10.5 and later.

Declared in `IKImageView.h`.

Declared In
`IKImageView.h`

IKPictureTaker Class Reference

Inherits from	NSPanel : NSWindow : NSResponder : NSObject
Conforms to	NSUserInterfaceValidations (NSWindow) NSAnimatablePropertyContainer (NSWindow) NSCoding (NSResponder) NSObject (NSObject)
Framework	System/Library/Frameworks/Quartz.framework/ImageKit.framework
Availability	Available in Mac OS X v10.5 and later.
Declared in	ImageKit/IKPictureTaker.h

Overview

The `IKPictureTaker` class represents a panel that allows users to choose images by browsing the file system. The picture taker panel provides an Open Recent menu, supports image cropping, and supports taking snapshots from an iSight or other digital camera.

Tasks

Getting and Setting Images

`- setInputImage:` (page 81)
Set the image input for the picture taker.

`- inputImage` (page 80)
Returns the input image associated with the picture taker.

`- outputImage` (page 80)
Returns the edited image.

Managing the Picture Taker

`+ pictureTaker` (page 78)
Returns a shared `IKPictureTaker` instance, creating it if necessary.

`- beginPictureTakerSheetForWindow:withDelegate:didEndSelector:contextInfo:` (page 78)
Opens a picture taker as a sheet whose parent is the specified window.

- `beginPictureTakerWithDelegate:didEndSelector:contextInfo:` (page 79)
  Opens a picture taker pane.
- `popUpRecentsMenuForView:withDelegate:didEndSelector:contextInfo:` (page 81)
  Displays the Open Recent popup menu associated with the picture taker.
- `runModal` (page 81)
  Opens a modal picture taker dialog.

Getting and Setting Mirroring

- `setMirroring:` (page 82)
  Controls whether the receiver enables video mirroring during snapshots.
- `mirroring` (page 80)
  Returns whether video mirroring is enabled during snapshots.

Class Methods

pictureTaker

Returns a shared `IKPictureTaker` instance, creating it if necessary.

```
+ (IKPictureTaker *) pictureTaker;
```

Return Value
An `IKPictureTaker` object.

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

Declared In
`IKPictureTaker.h`

Instance Methods

beginPictureTakerSheetForWindow:withDelegate:didEndSelector:contextInfo:

Opens a picture taker as a sheet whose parent is the specified window.

```
- (void) beginPictureTakerSheetForWindow:(NSWindow *)aWindow withDelegate:(id)
    delegate didEndSelector:(SEL) didEndSelector contextInfo:(void *) contextInfo;
```

Parameters
aWindow
The parent window of the picture taker sheet.

delegate
The object that will invoke the selector `didEndSelector` when the picture taker session terminates.

didEndSelector
The selector to invoke when the picture taker session terminates.

contextInfo
Any data that must be passed as an argument to the delegate through `didEndSelector` after the picture taker session terminates.

Discussion
The `didEndSelector` method should have the following signature:

```
- (void)pictureTakerDidEnd:(IKPictureTaker *)sheet returnCode:(NSInteger)returnCode
contextInfo:(void *)contextInfo;
```

The `returnCode` value is set to `NSOKButton` if the user validates, or to `NSCancelButton` if the user cancels.

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

See Also
`- beginPictureTakerWithDelegate:didEndSelector:contextInfo:` (page 79)

Declared In
`IKPictureTaker.h`

beginPictureTakerWithDelegate:didEndSelector:contextInfo:

Opens a picture taker pane.

```
- (void) beginPictureTakerWithDelegate:(id) delegate didEndSelector:(SEL)
    didEndSelector contextInfo:(void *) contextInfo;
```

Parameters
delegate
The object that will invoke the selector `didEndSelector` when the picture taker session terminates.

didEndSelector
The selector to invoke when the picture taker session terminates.

contextInfo
Any data that must be passed as an argument to the delegate through `didEndSelector` after the picture taker session terminates.

Discussion
The `didEndSelector` method should have the following signature:

```
- (void)pictureTakerDidEnd:(IKPictureTaker *)sheet returnCode:(NSInteger)returnCode
contextInfo:(void *)contextInfo;
```

The `returnCode` value is set to `NSOKButton` if the user validates, or to `NSCancelButton` if the user cancels.

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

See Also
`- beginPictureTakerSheetForWindow:withDelegate:didEndSelector:contextInfo:` (page 78)

Declared In
IKPictureTaker.h

inputImage

Returns the input image associated with the picture taker.

```
- (NSImage*) inputImage;
```

Return Value
The input image.

Discussion
The input image is never modified by the picture taker.

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

See Also
- setInputImage: (page 81)

Declared In
IKPictureTaker.h

mirroring

Returns whether video mirroring is enabled during snapshots.

```
- (BOOL) mirroring;
```

Return Value
Returns YES if video mirroring is enabled, NO otherwise.

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

Declared In
IKPictureTaker.h

outputImage

Returns the edited image.

```
- (NSImage*) outputImage;
```

Return Value
The edited image.

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

Declared In
IKPictureTaker.h

popUpRecentsMenuForView:withDelegate:didEndSelector:contextInfo:

Displays the Open Recent popup menu associated with the picture taker.

```
- (void) popUpRecentsMenuForView:(NSView *) aView withDelegate:(id) delegate
    didEndSelector:(SEL) didEndSelector contextInfo:(void *) contextInfo;
```

Parameters

delegate
The object that will invoke the selector `didEndSelector` when the picture taker session terminates.

didEndSelector
The selector to invoke when the picture taker session terminates.

contextInfo
Any data that must be passed as an argument to the delegate through `didEndSelector` after the picture taker session terminates.

Discussion
The `didEndSelector` method should have the following signature:

```
- (void)pictureTakerDidEnd:(IKPictureTaker *)sheet returnCode:(NSInteger)returnCode
contextInfo:(void *)contextInfo;
```

The `returnCode` value is set to `NSOKButton` if the user validates, or to `NSCancelButton` if the user cancels.

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

Declared In
`IKPictureTaker.h`

runModal

Opens a modal picture taker dialog.

```
- (NSInteger) runModal;
```

Return Value
Returns `NSOKButton` if the user edits or chooses an image; `NSCancelButton` if the user cancels or does not change the default image.

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

Declared In
`IKPictureTaker.h`

setInputImage:

Set the image input for the picture taker.

```
- (void) setInputImage:(NSImage *) image;
```

Parameters

image

An `NSImage` object.

Discussion

The input image is never modified by the picture taker.

Availability

Available in Mac OS X v10.5 and later.

See Also

`- inputImage` (page 80)

Declared In

`IKPictureTaker.h`

setMirroring:

Controls whether the receiver enables video mirroring during snapshots.

```
- (void) setMirroring:(BOOL)b;
```

Parameters

b

The default setting is `YES`.

Availability

Available in Mac OS X v10.5 and later.

Declared In

`IKPictureTaker.h`

Constants

Picture Taker Keys

Keys for customizing the picture taker appearance and behavior.

```
NSString *const IKPictureTakerAllowsVideoCaptureKey;
NSString *const IKPictureTakerAllowsFileChoosingKey;
NSString *const IKPictureTakerShowRecentPictureKey;
NSString *const IKPictureTakerUpdateRecentPictureKey;
NSString *const IKPictureTakerAllowsEditingKey;
NSString *const IKPictureTakerShowEffectsKey;
NSString *const IKPictureTakerInformationalTextKey;
NSString *const IKPictureTakerImageTransformsKey;
NSString *const IKPictureTakerOutputImageMaxSizeKey;
NSString *const IKPictureTakerCropAreaSizeKey;
NSString *const IKPictureTakerShowAddressBookPictureKey;
NSString *const IKPictureTakerShowEmptyPictureKey;
```

Constants

`IKPictureTakerAllowsVideoCaptureKey`

A key for allowing video capture. The associated value is an `NSNumber` value (`BOOL`) whose default value is `YES`.

Available in Mac OS X v10.5 and later.

Declared in `IKPictureTaker.h`.

`IKPictureTakerAllowsFileChoosingKey`

A key for allowing the user to choose a file. The associated value is an `NSNumber` object that contains a `BOOL` value whose default value is `YES`.

Available in Mac OS X v10.5 and later.

Declared in `IKPictureTaker.h`.

`IKPictureTakerUpdateRecentPictureKey`

A key for allowing a recent picture to be updated. The associated value is an `NSNumber` object that contains a `BOOL` value whose default value is `YES`.

Available in Mac OS X v10.5 and later.

Declared in `IKPictureTaker.h`.

`IKPictureTakerAllowsEditingKey`

A key for allowing image editing. The associated value is an `NSNumber` object that contains a `BOOL` value whose default value is `YES`.

Available in Mac OS X v10.5 and later.

Declared in `IKPictureTaker.h`.

`IKPictureTakerShowEffectsKey`

A key for showing effects. The associated value is an `NSNumber` object that contains a `BOOL` value whose default value is `NO`.

Available in Mac OS X v10.5 and later.

Declared in `IKPictureTaker.h`.

`IKPictureTakerInformationalTextKey`

A key for informational text. The associated value is an `NSString` or `NSAttributedString` object whose default value is `"Drag Image Here"`.

Available in Mac OS X v10.5 and later.

Declared in `IKPictureTaker.h`.

`IKPictureTakerImageTransformsKey`

A n image transformation key. The associated value is an `NSDictionary` object that can be serialized.

Available in Mac OS X v10.5 and later.

Declared in `IKPictureTaker.h`.

`IKPictureTakerOutputImageMaxSizeKey`
A key for the maximum size of the output image. The associated value is an `NSValue` object (`NSSize`).

Available in Mac OS X v10.5 and later.

Declared in `IKPictureTaker.h`.

`IKPictureTakerCropAreaSizeKey`
A key for the cropping area size. The associated value is an `NSValue` object (`NSSize`).

Available in Mac OS X v10.5 and later.

Declared in `IKPictureTaker.h`.

`IKPictureTakerShowAddressBookPictureKey`
A key for showing the address book picture. The associated value is a Boolean value packages as an `NSNumber` object. The default value is `NO`. If set to YES, the picture taker automatically adds the address book image for the Me user at the end of the Recent Pictures pop-up menu.

Available in Mac OS X v10.5 and later.

Declared in `IKPictureTaker.h`.

`IKPictureTakerShowEmptyPictureKey`
A key for showing an empty picture. The associated value is an `NSImage` object. The default value is `nil`. If set to an image, the picture taker automatically shows an image at the end of the Recent Pictures pop-up menu. that means "no picture."

Available in Mac OS X v10.5 and later.

Declared in `IKPictureTaker.h`.

Discussion
You can set picture taker options using `setValue:forKey` (`NSKeyValueCoding`).

Declared In
`IKPictureTaker.h`

IKSaveOptions Class Reference

Inherits from	NSObject
Conforms to	NSObject (NSObject)
Framework	System/Library/Frameworks/Quartz.framework/ImageKit.framework
Availability	Available in Mac OS X v10.5 and later.
Declared in	ImageKit/IKSaveOptions.h

Overview

The `IKSaveOptions` class initializes, adds, and manages user interface options for saving image data.

Tasks

Initializing and Adding Options

- `initWithImageProperties:imageUTType:` (page 87)
  Initializes a save options accessory pane for the provided image properties and uniform type identifier.
- `addSaveOptionsAccessoryViewToSavePanel:` (page 86)
  Adds a save options accessory view to a save panel.

Retrieving User Settings

- `imageProperties` (page 86)
  Returns a dictionary of image properties that reflects the user's selection.
- `imageUTType` (page 86)
  Returns the uniform type identifier that reflects the user's selection.
- `userSelection` (page 87)
  Returns a dictionary that contains the save options selected by the user.

Instance Methods

addSaveOptionsAccessoryViewToSavePanel:

Adds a save options accessory view to a save panel.

```
- (void)addSaveOptionsAccessoryViewToSavePanel: (NSSavePanel*)savePanel;
```

Parameters

savePanel

The panel you want to extend.

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

Declared In
IKSaveOptions.h

imageProperties

Returns a dictionary of image properties that reflects the user's selection.

```
- (NSDictionary *)imageProperties;
```

Return Value
A dictionary of updated image properties.

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

Declared In
IKSaveOptions.h

imageUTType

Returns the uniform type identifier that reflects the user's selection.

```
- (NSString *)imageUTType;
```

Return Value
A string that specifies the uniform type identifier of the selection.

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

Declared In
IKSaveOptions.h

initWithImageProperties:imageUTType:

Initializes a save options accessory pane for the provided image properties and uniform type identifier.

```
- (id)initWithImageProperties: (NSDictionary*)imageProperties imageUTType:
   (NSString*)imageUTType;
```

Parameters

imageProperties

A dictionary of image properties.

imageUTType

A string that specifies a uniform type identifier, such as `JPEG`. See *Uniform Type Identifiers Overview*.

Return Value
The initialized object.

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

Declared In
`IKSaveOptions.h`

userSelection

Returns a dictionary that contains the save options selected by the user.

```
- (NSDictionary *)userSelection;
```

Return Value
A dictionary of save options.

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

Declared In
`IKSaveOptions.h`

IKSlideshow Class Reference

Inherits from	NSObject
Conforms to	NSObject (NSObject)
Framework	System/Library/Frameworks/Quartz.framework/ImageKit.framework
Availability	Available in Mac OS X v10.5 and later.
Declared in	ImageKit/IKSlideshow.h

Overview

The `IKSlideshow` class encapsulates a data source and options for a slideshow.

Tasks

Getting Slideshow Data

`- indexOfCurrentSlideshowItem` (page 92)
Returns the index of the current slideshow item.

Reloading Data

`- reloadData` (page 92)
Reloads the data for a slideshow.

`- reloadSlideshowItemAtIndex:` (page 92)
Reloads the data for a slideshow, starting at the specified index.

Exporting Slideshow Items

`+ canExportToApplication:` (page 90)
Finds out whether the slideshow can export its contents to an application.

`+ exportSlideshowItem:toApplication:` (page 91)
Exports a slideshow item to the application that has the provided bundle identifier.

Creating a Shared Instance of a Slideshow

`+ sharedSlideshow` (page 91)
Returns a shared instance of a slideshow.

Running and Stopping a Slideshow

`- runSlideshowWithDataSource:inMode:options:` (page 93)
Runs a slideshow that contains the specified kind of items, provided from a data source.

`- stopSlideshow:` (page 93)
Stops a slideshow.

`autoPlayDelay` (page 90) *property*
Controls the interval of time before a slideshow starts to play automatically.

Properties

For more about Objective-C properties, see "Properties" in *The Objective-C 2.0 Programming Language*.

autoPlayDelay

Controls the interval of time before a slideshow starts to play automatically.

```
@property NSTimeInterval autoPlayDelay;
```

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

Declared In
`IKSlideshow.h`

Class Methods

canExportToApplication:

Finds out whether the slideshow can export its contents to an application.

```
+ (BOOL)canExportToApplication:(NSString *)applicationBundleIdentifier
```

Parameters
`applicationBundleIdentifier`
The bundle identifier of the application that you want to export the slideshow to. See "`Bundle Identifiers`" (page 93).

Return Value
`YES` if the slideshow can be exported to the specified application; `NO` otherwise.

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

See Also
`+ exportSlideshowItem:ToApplication:` (page 91)

Declared In
`IKSlideshow.h`

exportSlideshowItem:toApplication:

Exports a slideshow item to the application that has the provided bundle identifier.

```
+ (void)exportSlideshowItem:(id)item toApplication:(NSString
    *)applicationBundleIdentifier
```

Parameters

item
The item to export

applicationBundleIdentifier
The bundle identifier of the application that you want to export the item to.

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

See Also
`+ canExportToApplication:` (page 90)

Declared In
`IKSlideshow.h`

sharedSlideshow

Returns a shared instance of a slideshow.

```
+ (IKSlideshow *)sharedSlideshow
```

Return Value
A slideshow object.

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

Declared In
`IKSlideshow.h`

Instance Methods

indexOfCurrentSlideshowItem

Returns the index of the current slideshow item.

```
- (NSUInteger)indexOfCurrentSlideshowItem
```

Return Value
The index of the current item in the slideshow.

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

Declared In
`IKSlideshow.h`

reloadData

Reloads the data for a slideshow.

```
- (void)reloadData
```

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

See Also
`- reloadSlideshowItemAtIndex:` (page 92)

Declared In
`IKSlideshow.h`

reloadSlideshowItemAtIndex:

Reloads the data for a slideshow, starting at the specified index.

```
- (void)reloadSlideshowItemAtIndex:(NSUInteger)index
```

Parameters
index
The index that species where to reload the slideshow data.

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

See Also
`- reloadData` (page 92)

Declared In
`IKSlideshow.h`

runSlideshowWithDataSource:inMode:options:

Runs a slideshow that contains the specified kind of items, provided from a data source.

```
- (void)runSlideshowWithDataSource:(id < IKSlideshowDataSource >)dataSource
    inMode:(NSString *)slideshowMode options:(NSDictionary *)slideshowOptions
```

Parameters

dataSource

The data source to use for the slideshow.

slideshowMode

A constant that indicate what kind of items are in the slideshow—`IKSlideshowModeImages`, `IKSlideshowModePDF`, or `IKSlideshowModeQuickLook`. See "Slideshow Modes" (page 94).

slideshowOptions

A dictionary of slideshow options. See "Slideshow Option Keys" (page 94).

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

Declared In
`IKSlideshow.h`

stopSlideshow:

Stops a slideshow.

```
- (void)stopSlideshow:(id)sender
```

Parameters

sender

The object sending the message to stop the slideshow.

Discussion
This method is invoked when the user clicks a button or issues a stop command.

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

Declared In
`IKSlideshow.h`

Constants

Bundle Identifiers

Identifiers for exporting slideshow items to an application.

```
NSString *const IK_iPhotoBundleIdentifier;
```

Constants

`IK_iPhotoBundleIdentifier`

The iPhoto application—`com.apple.iPhoto`.

Available in Mac OS X v10.5 and later.

Declared in `IKSlideshow.h`.

Declared In

`IKSlideshow.h`

Slideshow Modes

The kind of items in the slideshow.

```
extern NSString *const IKSlideshowModeImages;
extern NSString *const IKSlideshowModePDF;
extern NSString *const IKSlideshowModeOther;
```

Constants

`IKSlideshowModeImages`

All items in the slideshow are images.

Available in Mac OS X v10.5 and later.

Declared in `IKSlideshow.h`.

`IKSlideshowModePDF`

All items in the slideshow are PDF documents.

Available in Mac OS X v10.5 and later.

Declared in `IKSlideshow.h`.

`IKSlideshowModeOther`

There are a mixture of items in the slideshow (image, PDF, text, HTML, and so on).

Available in Mac OS X v10.5 and later.

Declared in `IKSlideshow.h`.

Declared In

`IKSlideshow.h`

Slideshow Option Keys

Keys for slideshow options.

```
NSString *const IKSlideshowWrapAround;
NSString *const IKSlideshowStartPaused;
NSString *const IKSlideshowStartIndex;
NSString *const IKSlideshowPDFDisplayBox;
NSString *const IKSlideshowPDFDisplayMode;
NSString *const IKSlideshowPDFDisplaysAsBook;
```

Constants

`IKSlideshowWrapAround`

A key for starting the slideshow over after the last slide shows. The associated value is a `Boolean` data type.

Available in Mac OS X v10.5 and later.

Declared in `IKSlideshow.h`.

`IKSlideshowStartPaused`

A key for starting in a paused state. The associated value is a `Boolean` data type.

Available in Mac OS X v10.5 and later.

Declared in `IKSlideshow.h`.

`IKSlideshowStartIndex`

A key for the slideshow item index. The associated value is an index.

Available in Mac OS X v10.5 and later.

Declared in `IKSlideshow.h`.

`IKSlideshowPDFDisplayBox`

A key for the PDF display box. The associated value is a type of display box, such as `kPDFDisplayBoxMediaBox` or `kPDFDisplayBoxMediaBox`. See *PDFPage Class Reference* for more information.

Available in Mac OS X v10.5 and later.

Declared in `IKSlideshow.h`.

`IKSlideshowPDFDisplayMode`

A key for the PDF display mode. The associated value is a PDF display mode constant, such as `kPDFDisplaySinglePage` or `kPDFDisplayTwoUp`. See *PDFView Class Reference* for more information.

Available in Mac OS X v10.5 and later.

Declared in `IKSlideshow.h`.

`IKSlideshowPDFDisplaysAsBook`

A key for displaying the slideshow as a book. The associated value is a `Boolean` data type.

Available in Mac OS X v10.5 and later.

Declared in `IKSlideshow.h`.

Declared In

`IKSlideshow.h`

PDFAction Class Reference

Inherits from	NSObject
Conforms to	NSObject (NSObject)
Framework	Library/Frameworks/Quartz.framework/Frameworks/PDFKit.framework
Declared in	PDFKit/PDFAction.h
Availability	Available in Mac OS X v10.5 and later.

Overview

`PDFAction`, a subclass of `NSObject`, represents an action that is performed when, for example, a PDF annotation is activated or an outline item is clicked.

A `PDFAction` object represents an action associated with a PDF element, such as an annotation or a link, that the viewer application can perform. See the Adobe PDF Specification for more about actions and action types.

`PDFAction` is an abstract superclass of the following concrete classes:

- `PDFActionGoTo`
- `PDFActionNamed`
- `PDFActionRemoteGoTo`
- `PDFActionResetForm`
- `PDFActionURL`

Tasks

Getting the Action Type

`- type` (page 98)
Returns the type of the action.

Instance Methods

type

Returns the type of the action.

```
- (NSString *)type
```

Return Value
The type of the PDF action.

Discussion
The PDF action type returned by this method may not correspond precisely to the name of a `PDFAction` subclass. For example, a `PDFActionURL` object might return “URI” or “Launch,” depending on the original action as defined by the Adobe PDF Specification. In the PDF Kit, these two actions are handled in the single `PDFActionURL` subclass, and the more familiar term “URL” is used instead.

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

Declared In
`PDFAction.h`

PDFActionGoTo Class Reference

Inherits from	PDFAction : NSObject
Conforms to	NSCopying NSObject (NSObject)
Framework	Library/Frameworks/Quartz.framework/Frameworks/PDFKit.framework
Declared in	PDFKit/PDFActionGoTo.h
Availability	Available in Mac OS X v10.5 and later.

Overview

`PDFActionGoTo`, a subclass of `PDFAction`, defines methods for getting and setting the destination of a go-to action.

A `PDFActionGoTo` object represents the action of going to a specific location within the PDF document.

Tasks

Accessing the Destination

- `destination` (page 100)
  Returns the destination associated with the action.
- `setDestination` (page 100)
  Sets the destination of the go-to action.

Initializing the Action

- `initWithDestination` (page 100)
  Initializes the go-to action.

Instance Methods

destination

Returns the destination associated with the action.

```
- (PDFDestination *)destination
```

Return Value
The destination specified by the go-to action.

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

See Also
- `setDestination` (page 100)

Declared In
`PDFActionGoTo.h`

initWithDestination

Initializes the go-to action.

```
- (id)initWithDestination:(PDFDestination *) destination;
```

Parameters
destination
The destination with which to initialize the go-to action.

Return Value
An initialized `PDFActionGoTo` instance, or `NULL` if the object could not be initialized.

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

setDestination

Sets the destination of the go-to action.

```
- (void)setDestination:(PDFDestination *)destination
```

Parameters
destination
The destination of the go-to action.

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

See Also
- `destination` (page 100)

PDFActionNamed Class Reference

Inherits from	PDFAction : NSObject
Conforms to	NSCopying NSObject (NSObject)
Framework	Library/Frameworks/Quartz.framework/Frameworks/PDFKit.framework
Declared in	PDFKit/PDFActionNamed.h
Availability	Available in Mac OS X v10.5 and later.

Overview

`PDFActionNamed` defines methods used to work with actions in PDF documents, some of which are named in the Adobe PDF Specification.

A `PDFActionNamed` object represents an action with a defined name, such as “Go back” or “Zoom in.“

Tasks

Accessing the Name of the Action

- `name` (page 102)
  Returns the name of the named action.
- `setName:` (page 102)
  Sets the name of the named action.

Initializing the Action

- `initWithName:` (page 102)
  Initializes the `PDFActionName` object with the specified named action.

Instance Methods

initWithName:

Initializes the `PDFActionName` object with the specified named action.

```
- (id)initWithName:(PDFActionNamedName)name
```

Parameters

name

The action name used to initialize the named action.

Return Value
An initialized `PDFActionNamed` instance, or `NULL` if the object could not be initialized.

Discussion
See "`Named Action Names`" for the names of named actions you can specify.

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

Declared In
`PDFActionNamed.h`

name

Returns the name of the named action.

```
- (PDFActionNamedName)name
```

Return Value
The name of the named action.

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

See Also
`- setName`

Declared In
`PDFActionNamed.h`

setName:

Sets the name of the named action.

```
- (void)setName:(PDFActionNamedName)name
```

Parameters

name

The action name to which to set the named action.

Discussion
See "`Named Action Names`" for the names of named actions you can specify.

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

See Also
`- name`

Declared In
`PDFActionNamed.h`

Constants

Named Action Names

Names of supported actions.

```
enum
{ kPDFActionNamedNone = 0,
    kPDFActionNamedNextPage = 1,
    kPDFActionNamedPreviousPage = 2,
    kPDFActionNamedFirstPage = 3,
    kPDFActionNamedLastPage = 4,
    kPDFActionNamedGoBack = 5,
    kPDFActionNamedGoForward = 6,
    kPDFActionNamedGoToPage = 7,
    kPDFActionNamedFind = 8,
    kPDFActionNamedPrint = 9,
    kPDFActionNamedZoomIn = 10,
    kPDFActionNamedZoomOut = 11
};
```

Constants

`kPDFActionNamedNone`
The action has no name.
Available in Mac OS X v10.5 and later.
Declared in `PDFActionNamed.h`.

`kPDFActionNamedNextPage`
The Next Page action.
Available in Mac OS X v10.5 and later.
Declared in `PDFActionNamed.h`.

`kPDFActionNamedPreviousPage`
The Previous Page action.
Available in Mac OS X v10.5 and later.
Declared in `PDFActionNamed.h`.

`kPDFActionNamedFirstPage`
The First Page action.

Available in Mac OS X v10.5 and later.

Declared in `PDFActionNamed.h`.

`kPDFActionNamedLastPage`
The Last Page action.

Available in Mac OS X v10.5 and later.

Declared in `PDFActionNamed.h`.

`kPDFActionNamedGoBack`
The Go Back action.

Available in Mac OS X v10.5 and later.

Declared in `PDFActionNamed.h`.

`kPDFActionNamedGoForward`
The Go Forward action.

Available in Mac OS X v10.5 and later.

Declared in `PDFActionNamed.h`.

`kPDFActionNamedGoToPage`
The Go to Page action.

Available in Mac OS X v10.5 and later.

Declared in `PDFActionNamed.h`.

`kPDFActionNamedFind`
The Find action.

Available in Mac OS X v10.5 and later.

Declared in `PDFActionNamed.h`.

`kPDFActionNamedPrint`
The Print action.

Available in Mac OS X v10.5 and later.

Declared in `PDFActionNamed.h`.

`kPDFActionNamedZoomIn`
The Zoom In action.

Available in Mac OS X v10.5 and later.

Declared in `PDFActionNamed.h`.

`kPDFActionNamedZoomOut`
The Zoom Out action.

Available in Mac OS X v10.5 and later.

Declared in `PDFActionNamed.h`.

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

Declared In
`PDFActionNamed.h`

PDFActionRemoteGoTo Class Reference

Inherits from	PDFAction : NSObject
Conforms to	NSCopying NSObject (NSObject)
Framework	Library/Frameworks/Quartz.framework/Frameworks/PDFKit.framework
Declared in	PDFKit/PDFActionRemoteGoTo.h
Availability	Available in Mac OS X v10.5 and later.

Overview

`PDFActionRemoteGoTo`, a subclass of `PDFAction`, defines methods for getting and setting the destination of a go-to action that targets another document.

Tasks

Initializing the Remote Go-to Action

- `initWithPageIndex:atPoint:fileURL:` (page 106)
  Initializes the remote go-to action with the specified page index, point, and document URL.

Accessing the Page Index of the Referenced Document

- `pageIndex` (page 106)
  Returns the zero-based page index referenced by the remote go-to action.
- `setPageIndex:` (page 107)
  Sets the zero-based page index referenced by the remote go-to action.

Accessing a Point on the Referenced Page

- `point` (page 107)
  Returns the point, in page space, on the page referenced by the remote go-to action.

- `setPoint:` (page 107)
  Sets the point, in page space, on the page referenced by the remote go-to action.

Accessing the URL of the Referenced Document

- `URL` (page 108)
  Returns the URL of the document referenced by the remote go-to action.
- `setURL:` (page 108)
  Sets the URL of the document referenced by the remote go-to action.

Instance Methods

initWithPageIndex:atPoint:fileURL:

Initializes the remote go-to action with the specified page index, point, and document URL.

```
- (id)initWithPageIndex:(NSUInteger)pageIndex atPoint:(NSPoint)point fileURL:(NSURL *)url
```

Parameters

pageIndex
  The page index of the remote document.

point
  The point on the page in the remote document.

url
  The URL of the remote PDF document.

Return Value
An initialized `PDFActionRemoteGoTo` instance, or `NULL` if the object could not be initialized..

Discussion
The `PDFActionRemoteGoTo` object uses a zero-based page index, not a `PDFPage` object. This simplifies the handling of remote destinations for documents that may not be instantiated yet.

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

Declared In
`PDFActionRemoteGoTo.h`

pageIndex

Returns the zero-based page index referenced by the remote go-to action.

```
- (NSUInteger)pageIndex
```

Return Value
The page index referenced by the remote go-to action.

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

See Also
- `setPageIndex:` (page 107)

Declared In
`PDFActionRemoteGoTo.h`

point

Returns the point, in page space, on the page referenced by the remote go-to action.

```
- (NSPoint)point
```

Return Value
The point on the page of the remote document referenced by the action. If either the x value or the y value of the point is `kPDFDestinationUnspecifiedValue`, no position on the page is specified.

Discussion
Page space is a 72-dpi coordinate system with the origin at the lower-left corner of the current page.

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

Declared In
`PDFActionRemoteGoTo.h`

setPageIndex:

Sets the zero-based page index referenced by the remote go-to action.

```
- (void)setPageIndex:(NSUInteger)pageIndex
```

Parameters
pageIndex
The page index in the remote document to go to.

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

See Also
- `pageIndex` (page 106)

Declared In
`PDFActionRemoteGoTo.h`

setPoint:

Sets the point, in page space, on the page referenced by the remote go-to action.

```
- (void)setPoint:(NSPoint)point
```

Parameters

point

The point on the remote page to go to. If either the x value or the y value of the point is `kPDFDestinationUnspecifiedValue`, no position on the page is specified.

Discussion

Page space is a 72-dpi coordinate system with the origin at the lower-left corner of the current page.

Availability

Available in Mac OS X v10.5 and later.

Declared In

`PDFActionRemoteGoTo.h`

setURL:

Sets the URL of the document referenced by the remote go-to action.

```
- (void)setURL:(NSURL *)url
```

Parameters

url

The URL of the remote document to go to.

Availability

Available in Mac OS X v10.5 and later.

Declared In

`PDFActionRemoteGoTo.h`

URL

Returns the URL of the document referenced by the remote go-to action.

```
- (NSURL *)URL
```

Return Value

The URL of the remote document referenced by the action.

Availability

Available in Mac OS X v10.5 and later.

Declared In

`PDFActionRemoteGoTo.h`

PDFActionResetForm Class Reference

Inherits from	PDFAction : NSObject
Conforms to	NSCopying NSObject (NSObject)
Framework	Library/Frameworks/Quartz.framework/Frameworks/PDFKit.framework
Declared in	PDFKit/PDFActionResetForm.h
Availability	Available in Mac OS X v10.5 and later.

Overview

`PDFActionResetForm`, a subclass of `PDFAction`, defines methods for getting and clearing fields in a PDF form.

A `PDFActionResetForm` object represents an action associated with a PDF form.

Tasks

Initializing a Reset Form Action

- `init` (page 110)
  Initializes a reset form action.

Accessing and Changing Fields

- `fields` (page 110)
  Returns an array of fields associated with the reset action.
- `setFields:` (page 111)
  Sets the array of fields associated with the reset action.

Determining Whether Fields Are Cleared When the Action Is Performed

- `fieldsIncludedAreCleared` (page 110)
  Returns whether the fields associated with the reset action are cleared when the action is performed.

- setFieldsIncludedAreCleared: (page 111)
  Sets whether the fields associated with the reset action are cleared when the action is performed.

Instance Methods

fields

Returns an array of fields associated with the reset action.

```
- (NSArray *)fields
```

Return Value
An array of NSString objects that corresponds to the fieldNames property of widget annotations (such as PDFAnnotationButtonWidget) on the PDF page. This method can return NULL.

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

See Also
- setFields

Declared In
PDFActionResetForm.h

fieldsIncludedAreCleared

Returns whether the fields associated with the reset action are cleared when the action is performed.

```
- (BOOL)fieldsIncludedAreCleared
```

Discussion
If YES, the reset action's fields are cleared when the action is performed. If NO, the fields are excluded from the reset action; that is, they are not cleared, but all other fields in the document are cleared.

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

See Also
- setFieldsIncludedAreCleared:

Declared In
PDFActionResetForm.h

init

Initializes a reset form action.

```
- (id)init
```

Return Value
An initialized `PDFActionResetForm` instance, or `NULL` if the object could not be initialized.

Discussion
Initially, there are no fields and `fieldsIncludedAreCleared` (page 110) returns `YES`.

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

Declared In
`PDFActionResetForm.h`

setFields:

Sets the array of fields associated with the reset action.

```
- (void)setFields:(NSArray *)fields
```

Parameters

fields

An array of strings that represent field names.

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

See Also
`- fields`

Declared In
`PDFActionResetForm.h`

setFieldsIncludedAreCleared:

Sets whether the fields associated with the reset action are cleared when the action is performed.

```
- (void)setFieldsIncludedAreCleared:(BOOL)include
```

Parameters

include

Pass `YES` to clear the fields associated with the action when the reset action is performed. Pass `NO` to exclude from the reset action only the fields associated with the action.

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

See Also
`- fieldsIncludedAreCleared`

Declared In
`PDFActionResetForm.h`

PDFActionURL Class Reference

Inherits from	PDFAction : NSObject
Conforms to	NSCopying NSObject (NSObject)
Framework	Library/Frameworks/Quartz.framework/Frameworks/PDFKit.framework
Declared in	PDFKit/PDFActionURL.h
Availability	Available in Mac OS X v10.5 and later.

Overview

`PDFActionURL`, a subclass of `PDFAction`, defines methods for getting and setting the URL associated with a URL action.

Tasks

Initializing a URL Action

- `initWithURL:` (page 114)
  Initializes a URL action with the specified URL.

Accessing and Changing the URL

- `URL` (page 114)
  Returns the URL associated with the URL action.
- `setURL:` (page 114)
  Sets the URL associated with the URL action.

Instance Methods

initWithURL:

Initializes a URL action with the specified URL.

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

Parameters

url

The URL to set the action to.

Return Value

An initialized `PDFActionURL` instance, or `NULL` if the object could not be initialized.

Availability

Available in Mac OS X v10.5 and later.

Declared In

`PDFActionURL.h`

setURL:

Sets the URL associated with the URL action.

```
- (void)setURL:(NSURL *)url
```

Parameters

url

The URL to set the action to.

Availability

Available in Mac OS X v10.5 and later.

See Also

`- URL`

Declared In

`PDFActionURL.h`

URL

Returns the URL associated with the URL action.

```
- (NSURL *)URL
```

Return Value

The URL associated with the action, or `NULL` if no URL is specified.

Availability

Available in Mac OS X v10.5 and later.

See Also
`- setURL:`

Declared In
`PDFActionURL.h`

PDFAnnotation Class Reference

Inherits from	NSObject
Conforms to	NSObject (NSObject)
Framework	Library/Frameworks/Quartz.framework/Frameworks/PDFKit.framework
Declared in	PDFKit/PDFAnnotation.h
Availability	Available in Mac OS X v10.4 and later.

Overview

`PDFAnnotation`, a subclass of `NSObject`, represents an annotation in a PDF document, which associates an object (such as a note or a sound) with a location in a PDF document.

In addition to its primary textual content, a PDF file can contain annotations that represent links, form elements, highlighting circles, textual notes, and so on. Each annotation is associated with a specific location on a page and may offer interactivity with the user. See the Adobe PDF Specification for more on annotations.

You are not likely to work with a `PDFAnnotation` object by itself, because the specific subclasses, such as `PDFAnnotationCircle`, are much more useful. When a PDF file is being parsed, however, any unknown or unsupported annotation is represented as a `PDFAnnotation` object.

`PDFAnnotation` is an abstract superclass of the following concrete classes:

- `PDFAnnotationButtonWidget`
- `PDFAnnotationCircle`
- `PDFAnnotationFreeText`
- `PDFAnnotationInk`
- `PDFAnnotationLine`
- `PDFAnnotationLink`
- `PDFAnnotationMarkup`
- `PDFAnnotationPopup`
- `PDFAnnotationSquare`
- `PDFAnnotationStamp`
- `PDFAnnotationText`
- `PDFAnnotationTextWidget`

Tasks

Initializing an Annotation

- `initWithBounds:` (page 122)
  Initializes a PDF annotation object.

Accessing Information About an Annotation

- `page` (page 123)
  Returns the page that the annotation is associated with.
- `modificationDate` (page 122)
  Returns the modification date of the annotation.
- `setModificationDate:` (page 125)
  Sets the modification date of the annotation.
- `userName` (page 129)
  Returns the name of the user who created the annotation.
- `setUserName:` (page 127)
  Sets the name of the user who created the annotation.
- `popup` (page 123)
  Returns the pop-up annotation associated with an annotation.
- `setPopup:` (page 126)
  Sets the pop-up annotation associated with an annotation.
- `mouseUpAction` (page 122)
  Returns the optional action performed when a user releases the mouse button within an annotation.
- `setMouseUpAction:` (page 126)
  Sets the action performed when a user releases the mouse button within an annotation.
- `type` (page 129)
  Returns the type of the annotation.
- `contents` (page 120)
  Returns the textual content (if any) associated with the annotation.
- `setContents:` (page 125)
  Specifies the textual content associated with the annotation.
- `toolTip` (page 128)
  Returns text for display as a help tag.

Managing Annotation Display Characteristics

- `bounds` (page 120)
  Returns the bounding box for the annotation in page space.
- `setBounds:` (page 124)
  Sets the bounding box for the annotation.

- `border` (page 119)
  Returns the border style for the annotation.
- `setBorder:` (page 124)
  Sets the border style for the annotation.
- `color` (page 120)
  Returns the stroke color for the annotation.
- `setColor:` (page 124)
  Sets the stroke color for the annotation.
- `hasAppearanceStream` (page 121)
  Returns a Boolean value that indicates whether the annotation has an appearance stream associated with it.

Managing Annotation Drawing and Output

- `drawWithBox:` (page 121)
  Draws the annotation on its associated page.
- `shouldDisplay` (page 128)
  Returns a Boolean value indicating whether the annotation should be displayed.
- `setShouldDisplay:` (page 126)
  Specifies whether the annotation should be displayed.
- `shouldPrint` (page 128)
  Returns a Boolean value indicating whether the annotation should appear when the document is printed.
- `setShouldPrint:` (page 127)
  Specifies whether the annotation should appear when the document is printed.

Instance Methods

border

Returns the border style for the annotation.

```
- (PDFBorder *)border
```

Return Value
The border style for the annotation. See "Constants" (page 195) in the PDFBorder class for possible values.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `setBorder:` (page 124)

Declared In
`PDFAnnotation.h`

bounds

Returns the bounding box for the annotation in page space.

```
- (NSRect)bounds
```

Return Value
The bounding box for the annotation in page space.

Discussion
Page space is a 72-dpi coordinate system with the origin at the lower-left corner of the current page.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `setBounds:` (page 124)

Declared In
`PDFAnnotation.h`

color

Returns the stroke color for the annotation.

```
- (NSColor *)color
```

Return Value
The stroke color for the annotation.

Discussion
Where this color is used depends on the type of annotation.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `setColor:` (page 124)

Declared In
`PDFAnnotation.h`

contents

Returns the textual content (if any) associated with the annotation.

```
- (NSString *)contents
```

Return Value
A string representing the textual content associated with the annotation.

Discussion
Textual content is typically associated with `PDFAnnotationText` and `PDFAnnotationFreeText` annotations.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `setContents:` (page 125)
- `toolTip` (page 128)

Declared In
`PDFAnnotation.h`

drawWithBox:

Draws the annotation on its associated page.

```
- (void)drawWithBox:(PDFDisplayBox)box
```

Parameters

box
The bounding box used to draw the annotation in.

Discussion
The annotation is drawn relative to the origin of *box* in page space.

Page space is a 72 dpi coordinate system with the origin at the lower-left corner of the current page.

For additional information see the "Constants" section in the `PDFPage` class.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `boundsForBox:` (page 241)

Declared In
`PDFAnnotation.h`

hasAppearanceStream

Returns a Boolean value that indicates whether the annotation has an appearance stream associated with it.

```
- (BOOL) hasAppearanceStream
```

Return Value
`YES` if the annotation has an appearance stream; otherwise `NO`.

Discussion
An appearance stream is a sequence of draw instructions used to render a PDF item. If an appearance stream exists, PDF Kit draws the annotation using the stream, which may override existing set parameters (such as the stroke color set with `setColor`).

Availability
Available in Mac OS X v10.4 and later.

Declared In
PDFAnnotation.h

initWithBounds:

Initializes a PDF annotation object.

```
- (id)initWithBounds:(NSRect)bounds
```

Parameters

bounds

The bounding box of the annotation, in page space.

Return Value
An initialized `PDFAnnotation` instance, or `NULL` if the object could not be initialized.

Discussion
Subclasses of `PDFAnnotation` should use this method to initialize annotation instances. Provide *bounds* in page space. Invoking `initWithBounds:` directly on a `PDFAnnotation` object creates an illegal `NULL` type.

Page space is a 72 dpi coordinate system with the origin at the lower-left corner of the current page.

Availability
Available in Mac OS X v10.4 and later.

Declared In
PDFAnnotation.h

modificationDate

Returns the modification date of the annotation.

```
- (NSDate *)modificationDate
```

Return Value
The modification date of the annotation, or `NULL` if there is no modification date.

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

See Also
- setModificationDate

Declared In
PDFAnnotation.h

mouseUpAction

Returns the optional action performed when a user releases the mouse button within an annotation.

```
- (PDFAction *)mouseUpAction
```

Return Value
The PDF action performed when a user releases the mouse button within an annotation.

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

See Also
- setMouseUpAction

Declared In
PDFAnnotation.h

page

Returns the page that the annotation is associated with.

```
- (PDFPage *)page
```

Return Value
The PDF page associated with the annotation.

Discussion
The addAnnotation: (page 239) method in the PDFPage class lets you associate an annotation with a page.

Availability
Available in Mac OS X v10.4 and later.

Declared In
PDFAnnotation.h

popup

Returns the pop-up annotation associated with an annotation.

```
- (PDFAnnotationPopup *)popup
```

Return Value
The pop-up annotation associated with the annotation, or NULL if no pop-up exists.

Discussion
Pop-up annotations are not used with links or widgets. The bounds and open state of the pop-up annotation indicate the placement and open state of the pop-up window.

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

See Also
- setPopup

Declared In
PDFAnnotation.h

setBorder:

Sets the border style for the annotation.

```
- (void)setBorder:(PDFBorder *)border
```

Parameters

border

The border style for the annotation. See "Constants" (page 195) in the PDFBorder class for the available styles. The border style attribute is optional.

Availability
Available in Mac OS X v10.4 and later.

See Also
- border (page 119)

Declared In
PDFAnnotation.h

setBounds:

Sets the bounding box for the annotation.

```
- (void)setBounds:(NSRect)bounds
```

Parameters

bounds

The bounding box for the annotation. Use page space for *bounds*. The bounds attribute is required for all annotations.

Discussion
Page space is a 72-dpi coordinate system with the origin at the lower-left corner of the current page.

Availability
Available in Mac OS X v10.4 and later.

See Also
- bounds (page 120)

Declared In
PDFAnnotation.h

setColor:

Sets the stroke color for the annotation.

```
- (void)setColor:(NSColor *)color
```

Parameters

color

The stroke color for the annotation.

Discussion
Where this color is used depends on the annotation type.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `color` (page 120)

Declared In
`PDFAnnotation.h`

setContents:

Specifies the textual content associated with the annotation.

```
- (void)setContents:(NSString *)contents
```

Parameters

contents
A string representing the textual contents associated with the annotation.

Discussion
Textual content is typically associated with `PDFAnnotationText` and `PDFAnnotationFreeText` annotations. For most annotation types, `PDFView` displays the associated textual content as a help tag.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `contents` (page 120)
- `toolTip` (page 128)

Declared In
`PDFAnnotation.h`

setModificationDate:

Sets the modification date of the annotation.

```
- (void)setModificationDate:(NSDate *)date
```

Parameters

date
The modification date to associate with the annotation.

Discussion
The modification date is optional.

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

See Also
- `modificationDate`

Declared In
`PDFAnnotation.h`

setMouseUpAction:

Sets the action performed when a user releases the mouse button within an annotation.

```
- (void)setMouseUpAction:(PDFAction *)action
```

Parameters

action

The PDF action to be performed when a user releases the mouse button within an annotation.

Discussion

The mouse-up action is optional.

Availability

Available in Mac OS X v10.5 and later.

See Also

- mouseUpAction

Declared In

PDFAnnotation.h

setPopup:

Sets the pop-up annotation associated with an annotation.

```
- (void)setPopup:(PDFAnnotationPopup *)popup
```

Parameters

popup

The pop-up annotation to associate with the annotation.

Discussion

A pop-up annotation is not associated with links or widgets. The bounds and open state of the pop-up annotation indicate the placement and open state of the pop-up window.

Availability

Available in Mac OS X v10.5 and later.

See Also

- popup

Declared In

PDFAnnotation.h

setShouldDisplay:

Specifies whether the annotation should be displayed.

```
- (void)setShouldDisplay:(BOOL)display
```

Parameters

display

Set this value to YES to display the annotation or NO otherwise.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `shouldDisplay` (page 128)

Declared In
`PDFAnnotation.h`

setShouldPrint:

Specifies whether the annotation should appear when the document is printed.

```
- (void)setShouldPrint:(BOOL)print
```

Parameters

print
Set this value to `YES` to ensure the annotation appears when the document is printed or `NO` otherwise.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `shouldPrint` (page 128)

Declared In
`PDFAnnotation.h`

setUserName:

Sets the name of the user who created the annotation.

```
- (void)userName:(NSString *)name
```

Parameters

name
The name of the user who created the annotation.

Discussion
The user name is optional.

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

See Also
- `userName`

Declared In
`PDFAnnotation.h`

shouldDisplay

Returns a Boolean value indicating whether the annotation should be displayed.

```
- (BOOL)shouldDisplay
```

Return Value
`YES` if the annotation should be displayed; otherwise `NO`.

Discussion
`PDFPage` respects this flag when drawing.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- setShouldDisplay:` (page 126)

Declared In
`PDFAnnotation.h`

shouldPrint

Returns a Boolean value indicating whether the annotation should appear when the document is printed.

```
- (BOOL)shouldPrint
```

Return Value
`YES` if the annotation should appear when the PDF document is printed; otherwise `NO`.

Discussion
`PDFPage` respects this flag when printing.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- setShouldPrint:` (page 127)

Declared In
`PDFAnnotation.h`

toolTip

Returns text for display as a help tag.

```
- (NSString *)toolTip
```

Return Value
A string that contains help tag content, or `NULL` if there is no text associated with the annotation.

Discussion
This method is equivalent to sending the message `[self contents]`. PDF Kit's annotation subclasses override this behavior as appropriate. For example, a `PDFAnnotationLink` object displays a URL or page destination for its help tag.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`PDFAnnotation.h`

type

Returns the type of the annotation.

```
- (NSString *)type
```

Return Value
The type of the annotation. Types include `Line`, `Link`, `Text`, and so on, referring to the `PDFAnnotation` subclasses. In the Adobe PDF Specification, this attribute is called `Subtype`, and the common "type" for all annotations in the PDF Specification is `Annot`.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`PDFAnnotation.h`

userName

Returns the name of the user who created the annotation.

```
- (NSString *)userName
```

Return Value
The name of the user who created the annotation, or NULL if no user name is set.

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

See Also
`- setUserName`

Declared In
`PDFAnnotation.h`

PDFAnnotationButtonWidget Class Reference

Inherits from	PDFAnnotation : NSObject
Conforms to	NSObject (NSObject)
Framework	Library/Frameworks/Quartz.framework/Frameworks/PDFKit.framework
Availability	Available in Mac OS X v10.4 and later.
Declared in	PDFKit/PDFAnnotationButtonWidget.h

Overview

A `PDFAnnotationButtonWidget` object provides user interactivity on a page of a PDF document. There are three types of buttons available: push button, radio button, and checkbox.

`PDFAnnotationButtonWidget` inherits general annotation behavior from the `PDFAnnotation` class. If you use a `PDFAnnotationButtonWidget` object, your application must handle hit testing, unless you are simply using `PDFView` to display content. This is because `PDFView` automatically handles hit testing for you.

Tasks

Getting and Setting the Control Type

- `controlType` (page 134)
  Returns the type of the control.
- `setControlType:` (page 137)
  Sets the type of the control.
- `parentID` (page 136)
  Gets the ID of the parent object. (Deprecated. If you need to find other buttons in the same group, such as a group of radio buttons, you do not need information about the parent object. Instead, look for button widget objects that return the same value in `fieldName`.)

Getting and Setting the Control's State

- `state` (page 140)
  Returns the state of the control.

- `setState:` (page 139)
  Sets the state of the control.

Getting and Setting the Control's Appearance

- `isHighlighted` (page 135)
  Returns a Boolean value that indicates whether the control is highlighted when it is drawn.
- `setHighlighted:` (page 138)
  Sets the control's highlighting when it is drawn.
- `backgroundColor` (page 133)
  Returns the background color of the control.
- `setBackgroundColor:` (page 136)
  Sets the control's background color.

Getting and Setting the Control Label Font Attributes

- `font` (page 135)
  Returns the font used in the control's label.
- `setFont:` (page 138)
  Sets the font of the control's label.
- `fontColor` (page 135)
  Returns the font color used in the control's label.
- `setFontColor:` (page 138)
  Sets the font color used in the control's label.

Getting and Setting the Control Label Text

- `caption` (page 133)
  Returns the text of the label on a push button control.
- `setCaption:` (page 137)
  Sets the text of the label on a push button control.

Managing Radio Button Behavior

- `allowsToggleToOff` (page 133)
  Returns a Boolean value indicating whether a radio button behaves in a toggling manner.

Managing Control State Values and Form Fields

- `onStateValue` (page 136)
  Returns the string associated with the on state of a radio button or checkbox control.

- `setOnStateValue:` (page 139)
  Sets the string that is associated with the on state of a radio button or checkbox control.
- `fieldName` (page 134)
  Returns the internal name of a field (used for reset-form actions).
- `setFieldName:` (page 137)
  Sets the internal name of a field (used for reset-form actions).

Instance Methods

allowsToggleToOff

Returns a Boolean value indicating whether a radio button behaves in a toggling manner.

```
- (BOOL)allowsToggleToOff
```

Return Value
`YES` if clicking a radio button control that is already in the on state toggles it to the off state; otherwise `NO`.

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

Declared In
`PDFAnnotationButtonWidget.h`

backgroundColor

Returns the background color of the control.

```
- (NSColor *)backgroundColor
```

Return Value
The color drawn in the background of the control.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- setBackgroundColor:` (page 136)

Declared In
`PDFAnnotationButtonWidget.h`

caption

Returns the text of the label on a push button control.

```
- (NSString *)caption
```

Return Value
The label drawn on a push button control.

Discussion
This method applies only to the label drawn on a control of type `kPDFWidgetPushButtonControl`.

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

See Also
`- setCaption:`

Declared In
`PDFAnnotationButtonWidget.h`

controlType

Returns the type of the control.

```
- (PDFWidgetControlType)controlType
```

Return Value
The type of control the button represents. See "Constants" (page 140) for the various control types.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- setControlType`

Declared In
`PDFAnnotationButtonWidget.h`

fieldName

Returns the internal name of a field (used for reset-form actions).

```
- (NSString *)fieldName
```

Return Value
The internal name of a field.

Discussion
The internal name of a field is an optional value.

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

See Also
`- setFieldName`

Declared In
`PDFAnnotationButtonWidget.h`

font

Returns the font used in the control's label.

```
- (NSFont *)font
```

Return Value
The font used in the control's label.

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

See Also
`- setFont:`

Declared In
`PDFAnnotationButtonWidget.h`

fontColor

Returns the font color used in the control's label.

```
- (NSColor *)fontColor
```

Return Value
The font color used in the control's label.

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

See Also
`- setFontColor:`

Declared In
`PDFAnnotationButtonWidget.h`

isHighlighted

Returns a Boolean value that indicates whether the control is highlighted when it is drawn.

```
- (BOOL)isHighlighted
```

Return Value
`YES` if the control is highlighted when it is drawn; otherwise `NO`.

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

See Also
`- setHighlighted:`

Declared In
`PDFAnnotationButtonWidget.h`

onStateValue

Returns the string associated with the on state of a radio button or checkbox control.

```
- (NSString *)onStateValue
```

Return Value
The string associated with the on state of a radio button or checkbox control.

Discussion
This is a required string for controls of types `kPDFWidgetRadioButtonControl` and `kPDFWidgetCheckBoxControl`. The off state is always labeled "Off".

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

See Also
`- setOnStateValue`

Declared In
`PDFAnnotationButtonWidget.h`

parentID

Gets the ID of the parent object. (**Deprecated.** If you need to find other buttons in the same group, such as a group of radio buttons, you do not need information about the parent object. Instead, look for button widget objects that return the same value in `fieldName`.)

```
- (unsigned)parentID
```

Discussion
For more information about the field names of annotations, see the Adobe PDF specification.

Availability
Available in Mac OS X v10.4 through Mac OS X v10.4.

Declared In
`PDFAnnotationButtonWidget.h`

setBackgroundColor:

Sets the control's background color.

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

Parameters
color
The color to be drawn in the control's background.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- backgroundColor` (page 133)

Declared In
PDFAnnotationButtonWidget.h

setCaption:

Sets the text of the label on a push button control.

```
- (void)setCaption:(NSString *)name
```

Parameters

name

The text to be used as the label on a push button control.

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

See Also
- caption

Declared In
PDFAnnotationButtonWidget.h

setControlType:

Sets the type of the control.

```
- (void)setControlType:(PDFWidgetControlType)type
```

Parameters

type

The type of control for the button. "Constants" (page 140) lists the various control types you can send for this value.

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

See Also
- controlType

Declared In
PDFAnnotationButtonWidget.h

setFieldName:

Sets the internal name of a field (used for reset-form actions).

```
- (void)setFieldName:(NSString *)name
```

Parameters

name

The internal name of a field. This is an optional value.

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

See Also
- fieldName

Declared In
PDFAnnotationButtonWidget.h

setFont:

Sets the font of the control's label.

```
- (void)setFont:(NSFont *)font
```

Parameters

font
The desired font for the control's label.

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

See Also
- font

Declared In
PDFAnnotationButtonWidget.h

setFontColor:

Sets the font color used in the control's label.

```
- (void)setFontColor:(NSColor *)color
```

Parameters

color
The desired font color of the control's label.

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

See Also
- fontColor

Declared In
PDFAnnotationButtonWidget.h

setHighlighted:

Sets the control's highlighting when it is drawn.

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

Parameters

flag

Set this value to YES to cause the control to be highlighted when it is drawn or NO otherwise.

Availability
Available in Mac OS X v10.4 and later.

See Also
- isHighlighted

Declared In
PDFAnnotationButtonWidget.h

setOnStateValue:

Sets the string that is associated with the on state of a radio button or checkbox control.

```
- (void)setOnStateValue:(NSString *)name
```

Discussion
Required for controls of types kPDFWidgetRadioButtonControl and kPDFWidgetCheckBoxControl, the value of *name* describes the on state of the control (the off state is always labeled "Off"). Although "On" is an acceptable string for the on state of a single checkbox, a group of two or more radio buttons should have a unique string associated with each control.

For example, a form might display a group of 3 radio buttons that allow users to indicate an account type, such as savings, checking, or investment. The strings associated with the on states of these buttons could be "Savings," "Checking," and "Investment." In this example, these 3 radio buttons also would share a field name string, such as "AccountType."

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

See Also
- onStateValue
- fieldName

Declared In
PDFAnnotationButtonWidget.h

setState:

Sets the state of the control.

```
- (void)setState:(int)value
```

Parameters

value

The state the control should be in.

Discussion
A control's state (for example, checked or unchecked) affects how it is drawn. Note that push buttons are always in the on state.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `state` (page 140)

Declared In
`PDFAnnotationButtonWidget.h`

state

Returns the state of the control.

```
- (int)state
```

Return Value
`NSOnState` if the control is on; `NSOffState` otherwise.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `setState:` (page 139)

Declared In
`PDFAnnotationButtonWidget.h`

Constants

Types of PDF Annotation Buttons

The types of annotation buttons.

```
typedef NSInteger PDFWidgetControlType;
enum {
    kPDFWidgetUnknownControl = -1,
    kPDFWidgetPushButtonControl = 0,
    kPDFWidgetRadioButtonControl = 1,
    kPDFWidgetCheckBoxControl = 2
};
```

Constants
`kPDFWidgetUnknownControl`
Unknown control type.

Available in Mac OS X v10.4 and later.

Declared in `PDFAnnotationButtonWidget.h`.

`kPDFWidgetPushButtonControl`
Push button control.

Available in Mac OS X v10.4 and later.

Declared in `PDFAnnotationButtonWidget.h`.

`kPDFWidgetRadioButtonControl`
Radio button control.

Available in Mac OS X v10.4 and later.

Declared in `PDFAnnotationButtonWidget.h`.

`kPDFWidgetCheckBoxControl`
Check box control.

Available in Mac OS X v10.4 and later.

Declared in `PDFAnnotationButtonWidget.h`.

PDFAnnotationChoiceWidget Class Reference

Inherits from	PDFAnnotation : NSObject
Conforms to	NSObject (NSObject)
Framework	Library/Frameworks/Quartz.framework/Frameworks/PDFKit.framework
Declared in	PDFKit/PDFAnnotationChoiceWidget.h
Availability	Available in Mac OS X v10.5 and later.

Overview

A `PDFAnnotationChoiceWidget` object provides user interactivity on a page of a PDF document, in the form of pop-up menus and lists.

`PDFAnnotationChoiceWidget` inherits general annotation behavior from the `PDFAnnotation` class. If you use a `PDFAnnotationChoiceWidget` object, your application must handle hit testing, unless you are simply using `PDFView` to display content. This is because `PDFView` automatically handles hit testing for you.

Tasks

Getting and Setting the String Value

- `stringValue` (page 149)
  Returns the selection in the widget annotation.
- `setStringValue:` (page 148)
  Sets the selection in the widget annotation.

Managing Font and Background Color Characteristics

- `backgroundColor` (page 144)
  Returns the color of the widget annotation background.
- `setBackgroundColor:` (page 146)
  Sets the background color of the widget annotation.
- `font` (page 145)
  Returns the font used to display the text in the widget annotation.

- `setFont:` (page 147)
  Sets the font used to display the text in the widget annotation.
- `fontColor` (page 146)
  Returns the font color used to display the text in the widget annotation.
- `setFontColor:` (page 147)
  Sets the font color used to display the text in the widget annotation.

Managing the Associated Field Name

- `fieldName` (page 145)
  Returns the internal field name associated with the widget annotation.
- `setFieldName:` (page 147)
  Sets the internal field name associated with the widget annotation's value.

Determining the Type of Choice Widget Annotation

- `isListChoice` (page 146)
  Returns a Boolean value indicating whether the widget annotation is a list.
- `setIsListChoice:` (page 148)
  Sets whether the widget annotation is a list.

Accessing the Items in the Choice Widget Annotation

- `choices` (page 145)
  Returns an array of strings that represent the items available in the list or pop-up menu of the choice widget annotation.
- `setChoices:` (page 146)
  Sets the items available in the list or pop-up menu of the choice widget annotation.

Instance Methods

backgroundColor

Returns the color of the widget annotation background.

```
- (NSColor *)backgroundColor
```

Return Value
The color of the widget annotation background.

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

Declared In
PDFAnnotationChoiceWidget.h

choices

Returns an array of strings that represent the items available in the list or pop-up menu of the choice widget annotation.

```
- (NSArray *)choices
```

Return Value
An array of strings that represent the items in the list or pop-up menu choice widget annotation.

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

Declared In
PDFAnnotationChoiceWidget.h

fieldName

Returns the internal field name associated with the widget annotation.

```
- (NSString *)fieldName
```

Return Value
The internal field name associated with the widget annotation.

Discussion
If the widget annotation is backed by PDF form data, it can associate an optional field name with a value or other data.

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

Declared In
PDFAnnotationChoiceWidget.h

font

Returns the font used to display the text in the widget annotation.

```
- (NSFont *)font
```

Return Value
The font used to display the text in the widget annotation.

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

Declared In
PDFAnnotationChoiceWidget.h

fontColor

Returns the font color used to display the text in the widget annotation.

```
- (NSColor *)fontColor
```

Return Value
The color of the font used for the text in the widget annotation.

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

Declared In
`PDFAnnotationChoiceWidget.h`

isListChoice

Returns a Boolean value indicating whether the widget annotation is a list.

```
- (BOOL)isListChoice
```

Return Value
`YES` if the widget annotation is a list, `NO` otherwise.

Discussion
A choice widget annotation can be either a list or a pop-up menu.

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

Declared In
`PDFAnnotationChoiceWidget.h`

setBackgroundColor:

Sets the background color of the widget annotation.

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

Parameters
color
The color to use in the background of the widget annotation.

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

Declared In
`PDFAnnotationChoiceWidget.h`

setChoices:

Sets the items available in the list or pop-up menu of the choice widget annotation.

```
- (void)setChoices:(NSArray *)options
```

Parameters

options

Send an array of strings, each of which represents an item in the list or pop-up menu of the choice annotation widget.

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

Declared In
PDFAnnotationChoiceWidget.h

setFieldName:

Sets the internal field name associated with the widget annotation's value.

```
- (void)setFieldName:(NSString *)name
```

Parameters

name

The name to be used as the internal field name associated with the widget annotation.

Discussion
If the widget annotation is backed by PDF form data, it can associate an optional field name with a value or other data.

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

Declared In
PDFAnnotationChoiceWidget.h

setFont:

Sets the font used to display the text in the widget annotation.

```
- (void)setFont:(NSFont *)font
```

Parameters

font

The font to be used for the text in the widget annotation.

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

Declared In
PDFAnnotationChoiceWidget.h

setFontColor:

Sets the font color used to display the text in the widget annotation.

```
- (void)setFontColor:(NSColor *)color
```

Parameters

color

The color of the font to be used for the text in the widget annotation.

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

Declared In
PDFAnnotationChoiceWidget.h

setIsListChoice:

Sets whether the widget annotation is a list.

```
- (void)setIsListChoice:(BOOL)isList
```

Parameters

isList

Send YES to set the choice widget annotation is a list, NO otherwise.

Discussion
A choice widget annotation can be either a list or a pop-up menu.

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

Declared In
PDFAnnotationChoiceWidget.h

setStringValue:

Sets the selection in the widget annotation.

```
- (void)setStringValue:(NSString *)value
```

Parameters

value

The string that represents the selection in the widget annotation.

Discussion
If the widget annotation object is backed by PDF form data, this method updates the value associated with the appropriate field in the form object.

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

Declared In
PDFAnnotationChoiceWidget.h

stringValue

Returns the selection in the widget annotation.

```
- (NSString *)stringValue
```

Return Value
The string that represents the selection in the widget annotation.

Discussion
If the widget annotation object is backed by PDF form data, this method returns the value associated with the appropriate field in the form object, if possible.

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

Declared In
`PDFAnnotationChoiceWidget.h`

PDFAnnotationCircle Class Reference

Inherits from	PDFAnnotation : NSObject
Conforms to	NSObject (NSObject)
Framework	Library/Frameworks/Quartz.framework/Frameworks/PDFKit.framework
Declared in	PDFKit/PDFAnnotationCircle.h
Availability	Available in Mac OS X v10.4 and later.

Overview

A PDFAnnotationCircle object displays an ellipse on a page. Circle annotations are like square annotations (instances of the PDFAnnotationSquare class) apart from the shape.

The `setLineWidth:` (page 194) and `setStyle:` (page 194) methods of the annotation's associated PDFBorder object determines the stroke thickness and style. The `setColor:` (page 124) method of the PDFAnnotation class determines the stroke color.

Tasks

Accessor Methods

- `interiorColor` (page 151)
  Returns the fill color used for drawing the annotation.
- `setInteriorColor:` (page 152)
  Sets the fill color used for drawing the annotation.

Instance Methods

interiorColor

Returns the fill color used for drawing the annotation.

```
- (NSColor *)interiorColor
```

Availability
Available in Mac OS X v10.4 and later.

See Also
- `setInteriorColor:` (page 152)

Declared In
`PDFAnnotationCircle.h`

setInteriorColor:

Sets the fill color used for drawing the annotation.

```
- (void)setInteriorColor:(NSColor *)color
```

Availability
Available in Mac OS X v10.4 and later.

See Also
- `interiorColor` (page 151)

Declared In
`PDFAnnotationCircle.h`

PDFAnnotationFreeText Class Reference

Inherits from	PDFAnnotation : NSObject
Conforms to	NSObject (NSObject)
Framework	Library/Frameworks/Quartz.framework/Frameworks/PDFKit.framework
Declared in	PDFKit/PDFAnnotationFreeText.h
Availability	Available in Mac OS X v10.4 and later.

Overview

A `PDFAnnotationFreeText` object displays text on a page.

Unlike a `PDFAnnotationText` object, a `PDFAnnotationFreeText` object has no open or closed state; its text is always visible. The text annotation performed in Preview uses `PDFAnnotationFreeText`.

The `PDFAnnotation` class's `contents` (page 120) and `setContents:` (page 125) methods let you get and set the textual content for a `PDFAnnotationFreeText` object.

Tasks

Managing Text Alignment

- `alignment` (page 154)
  Returns the horizontal alignment of text within the bounds of the annotation.
- `setAlignment:` (page 155)
  Sets the horizontal alignment of text within the bounds of the annotation.

Managing Font and Font Color

- `font` (page 154)
  Returns the font used for the annotation's text field.
- `setFont:` (page 155)
  Sets the font used in the text field of the annotation.
- `fontColor` (page 154)
  Returns the font color used in the text field of the annotation.

- setFontColor: (page 155)
  Sets the font color used in the text field of the annotation.

Instance Methods

alignment

Returns the horizontal alignment of text within the bounds of the annotation.

```
- (NSTextAlignment)alignment
```

Return Value
The horizontal alignment of text within the bounds of the annotation. Supported values are `NSLeftTextAlignment`, `NSRightTextAlignment`, and `NSCenterTextAlignment`.

Availability
Available in Mac OS X v10.4 and later.

See Also
- setAlignment: (page 155)

Declared In
PDFAnnotationFreeText.h

font

Returns the font used for the annotation's text field.

```
- (NSFont *)font
```

Return Value
The font used for the annotation's text field.

Availability
Available in Mac OS X v10.4 and later.

See Also
- setFont: (page 155)

Declared In
PDFAnnotationFreeText.h

fontColor

Returns the font color used in the text field of the annotation.

```
- (NSColor *)fontColor
```

Return Value
The font color used in the text field of the annotation.

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

See Also
- `setFontColor:`

Declared In
`PDFAnnotationFreeText.h`

setAlignment:

Sets the horizontal alignment of text within the bounds of the annotation.

```
- (void)setAlignment:(NSTextAlignment)alignment
```

Parameters

alignment
Send `NSLeftTextAlignment`, `NSRightTextAlignment`, or `NSCenterTextAlignment` to set the horizontal alignment of text within the bounds of the annotation.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `alignment` (page 154)

Declared In
`PDFAnnotationFreeText.h`

setFont:

Sets the font used in the text field of the annotation.

```
- (void)setFont:(NSFont *)font
```

Parameters

font
The font to be used in the text field of the annotation.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `font` (page 154)

Declared In
`PDFAnnotationFreeText.h`

setFontColor:

Sets the font color used in the text field of the annotation.

```
- (void)setFontColor:(NSColor *)color
```

Parameters

color

The font color to be used in the text field of the annotation.

Availability

Available in Mac OS X v10.5 and later.

See Also

```
- fontColor
```

Declared In

`PDFAnnotationFreeText.h`

PDFAnnotationInk Class Reference

Inherits from	PDFAnnotation : NSObject
Conforms to	NSObject (NSObject)
Framework	Library/Frameworks/Quartz.framework/Frameworks/PDFKit.framework
Declared in	PDFKit/PDFAnnotationInk.h
Availability	Available in Mac OS X v10.4 and later.

Overview

A PDFAnnotationInk object displays one or more disjoint Bezier paths on a page. This is typically used to represent a freehand jotting or "scribble" of handwritten text.

The `setLineWidth:` (page 194) and `setStyle:` (page 194) methods of the annotation's associated PDFBorder object determines the stroke thickness and style. The `setColor:` (page 124) method of the PDFAnnotation class determines the stroke color.

Tasks

Accessor Methods

- `paths` (page 158)
  Returns an array containing the Bezier paths that make up an annotation.

Working with Bezier Paths

- `addBezierPath:` (page 158)
  Adds a Bezier path to an annotation.
- `removeBezierPath:` (page 158)
  Removes a Bezier path from an annotation.

Instance Methods

addBezierPath:

Adds a Bezier path to an annotation.

```
- (void)addBezierPath:(NSBezierPath *)path
```

Availability
Available in Mac OS X v10.4 and later.

See Also
- `removeBezierPath:` (page 158)
- `paths` (page 158)

Declared In
`PDFAnnotationInk.h`

paths

Returns an array containing the Bezier paths that make up an annotation.

```
- (NSArray *)paths
```

Availability
Available in Mac OS X v10.4 and later.

See Also
- `addBezierPath:` (page 158)
- `removeBezierPath:` (page 158)

Declared In
`PDFAnnotationInk.h`

removeBezierPath:

Removes a Bezier path from an annotation.

```
- (void)removeBezierPath:(NSBezierPath *)path
```

Availability
Available in Mac OS X v10.4 and later.

See Also
- `addBezierPath:` (page 158)
- `paths` (page 158)

Declared In
`PDFAnnotationInk.h`

PDFAnnotationLine Class Reference

Inherits from	PDFAnnotation : NSObject
Conforms to	NSObject (NSObject)
Framework	Library/Frameworks/Quartz.framework/Frameworks/PDFKit.framework
Declared in	PDFKit/PDFAnnotationLine.h
Availability	Available in Mac OS X v10.4 and later.

Overview

A `PDFAnnotationLine` object displays a single line on a page.

The `setLineWidth:` (page 194) and `setStyle:` (page 194) methods of the annotation's associated `PDFBorder` object determines the stroke thickness and style. The `setColor:` (page 124) method of the `PDFAnnotation` class determines the stroke color.

Tasks

Specifying the Starting and Ending Points

- `startPoint` (page 163)
  Returns the starting point for the line.
- `setStartPoint:` (page 163)
  Sets the starting point for the line.
- `endPoint` (page 160)
  Returns the ending point for the line in page space.
- `setEndPoint:` (page 161)
  Sets the ending point for the line.

Specifying the Line Ending Styles

- `startLineStyle` (page 163)
  Returns the line ending style for the starting point of the line.

- `setStartLineStyle:` (page 162)
  Sets the line ending style for the starting point of the line.
- `endLineStyle` (page 160)
  Returns the line ending style for the ending point of the line.
- `setEndLineStyle:` (page 161)
  Sets the line ending style for the ending point of the line.

Specifying the Color of Line-end Ornaments

- `interiorColor` (page 161)
  Returns the color used to fill the ornament at the ends of the line.
- `setInteriorColor:` (page 162)
  Sets the color used to fill the ornament at the ends of the line.

Instance Methods

endLineStyle

Returns the line ending style for the ending point of the line.

```
- (PDFLineStyle)endLineStyle
```

Return Value
The line ending style for the ending point of the line.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `setEndLineStyle:` (page 161)
- `startLineStyle` (page 163)

Declared In
`PDFAnnotationLine.h`

endPoint

Returns the ending point for the line in page space.

```
- (NSPoint)endPoint
```

Return Value
The ending point for the line, in page space.

Discussion
Page space is a 72-dpi coordinate system with the origin at the lower-left corner of the current page.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `setEndPoint:` (page 161)
- `startPoint` (page 163)

Declared In
`PDFAnnotationLine.h`

interiorColor

Returns the color used to fill the ornament at the ends of the line.

```
- (NSColor *) interiorColor
```

Return Value
The color used in the line-end ornament at the ends of the line.

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

Declared In
`PDFAnnotationLine.h`

setEndLineStyle:

Sets the line ending style for the ending point of the line.

```
- (void)setEndLineStyle:(PDFLineStyle)style
```

Parameters
style
The line ending style for the ending point of the line.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `endLineStyle` (page 160)
- `setStartLineStyle:` (page 162)

Declared In
`PDFAnnotationLine.h`

setEndPoint:

Sets the ending point for the line.

```
- (void)setEndPoint:(NSPoint)point
```

Parameters

point

The ending point for the line, in page space.

Discussion

Page space is a 72-dpi coordinate system with the origin at the lower-left corner of the current page.

Availability

Available in Mac OS X v10.4 and later.

See Also

- endPoint (page 160)
- setStartPoint: (page 163)

Declared In

PDFAnnotationLine.h

setInteriorColor:

Sets the color used to fill the ornament at the ends of the line.

```
- (void)setInteriorColor:(NSColor *)color
```

Parameters

color

The color to be used to fill in the ornament at the ends of the line.

Discussion

The ornament at the end of a line is optional (for more information, see the Adobe PDF Specification 1.4).

Availability

Available in Mac OS X v10.5 and later.

Declared In

PDFAnnotationLine.h

setStartLineStyle:

Sets the line ending style for the starting point of the line.

```
- (void)setStartLineStyle:(PDFLineStyle)style
```

Parameters

style

Availability

Available in Mac OS X v10.4 and later.

See Also

- startLineStyle (page 163)
- setEndLineStyle: (page 161)

Declared In

PDFAnnotationLine.h

setStartPoint:

Sets the starting point for the line.

```
- (void)setStartPoint:(NSPoint)point
```

Parameters

point

The starting point for the line, in page space.

Discussion

Page space is a 72-dpi coordinate system with the origin at the lower-left corner of the current page.

Availability

Available in Mac OS X v10.4 and later.

See Also

- `startPoint` (page 163)
- `setEndPoint:` (page 161)

Declared In

`PDFAnnotationLine.h`

startLineStyle

Returns the line ending style for the starting point of the line.

```
- (PDFLineStyle)startLineStyle
```

Return Value

The line ending style for the starting point of the line.

Availability

Available in Mac OS X v10.4 and later.

See Also

- `setStartLineStyle:` (page 162)
- `endLineStyle` (page 160)

Declared In

`PDFAnnotationLine.h`

startPoint

Returns the starting point for the line.

```
- (NSPoint)startPoint
```

Return Value

The starting point for the line, in page space.

Discussion

Page space is a 72-dpi coordinate system with the origin at the lower-left corner of the current page.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `setStartPoint:` (page 163)
- `endPoint` (page 160)

Declared In
`PDFAnnotationLine.h`

Constants

The following constants specify the available line ending styles:

Constant	Description
`kPDFLineStyleNone`	No line ending. Available in Mac OS X v10.4 and later. Declared in `PDFAnnotationLine.h`.
`kPDFLineStyleSquare`	A square line ending filled with the annotation's interior color, if any. Available in Mac OS X v10.4 and later. Declared in `PDFAnnotationLine.h`.
`kPDFLineStyleCircle`	A circular line ending filled with the annotation's interior color, if any. Available in Mac OS X v10.4 and later. Declared in `PDFAnnotationLine.h`.
`kPDFLineStyleDiamond`	A diamond-shaped line ending filled with the annotation's interior color, if any. Available in Mac OS X v10.4 and later. Declared in `PDFAnnotationLine.h`.
`kPDFLineStyleOpenArrow`	An open arrowhead line ending, composed from two short lines meeting in an acute angle at the line end. Available in Mac OS X v10.4 and later. Declared in `PDFAnnotationLine.h`.
`kPDFLineStyle-ClosedArrow`	A closed arrowhead line ending, consisting of a triangle with the acute vertex at the line end and filled with the annotation's interior color, if any. Available in Mac OS X v10.4 and later. Declared in `PDFAnnotationLine.h`.

PDFAnnotationLink Class Reference

Inherits from	PDFAnnotation : NSObject
Conforms to	NSObject (NSObject)
Framework	Library/Frameworks/Quartz.framework/Frameworks/PDFKit.framework
Declared in	PDFKit/PDFAnnotationLink.h
Availability	Available in Mac OS X v10.4 and later.

Overview

A PDFAnnotationLink object represents either a hypertext link to another location in the document (specified as a PDFDestination object) or a URL.

Tasks

Working with Link Destinations

- `destination` (page 166)
  Gets the destination for the link when the destination was specified as a PDFDestination object.
- `setDestination:` (page 166)
  Sets the destination for the link as a PDFDestination object.
- `URL` (page 167)
  Gets the destination for the link when the destination was specified as a URL.
- `setURL:` (page 167)
  Sets the destination for the link as a URL.

Highlighting the Link

- `setHighlighted:` (page 166)
  Sets the highlighting state for the link.

Instance Methods

destination

Gets the destination for the link when the destination was specified as a PDFDestination object.

```
- (PDFDestination *)destination
```

Availability
Available in Mac OS X v10.4 and later.

See Also
- `setDestination:` (page 166)

Declared In
`PDFAnnotationLink.h`

setDestination:

Sets the destination for the link as a PDFDestination object.

```
- (void)setDestination:(PDFDestination *)destination
```

Availability
Available in Mac OS X v10.4 and later.

See Also
- `destination` (page 166)

Declared In
`PDFAnnotationLink.h`

setHighlighted:

Sets the highlighting state for the link.

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

Discussion
For typical PDF interaction, when a user clicks (mouse-down) on a link, set highlighting to `YES` and redraw the link. On the subsequent mouse-up event, set highlighting to `NO` and redraw again.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`PDFAnnotationLink.h`

setURL:

Sets the destination for the link as a URL.

```
- (void)setURL:(NSURL *)url
```

Availability
Available in Mac OS X v10.4 and later.

See Also
- URL (page 167)

Declared In
PDFAnnotationLink.h

URL

Gets the destination for the link when the destination was specified as a URL.

```
- (NSURL *)URL
```

Availability
Available in Mac OS X v10.4 and later.

See Also
- setURL: (page 167)

Declared In
PDFAnnotationLink.h

PDFAnnotationMarkup Class Reference

Inherits from	PDFAnnotation : NSObject
Conforms to	NSObject (NSObject)
Framework	Library/Frameworks/Quartz.framework/Frameworks/PDFKit.framework
Declared in	PDFKit/PDFAnnotationMarkup.h
Availability	Available in Mac OS X v10.4 and later.

Overview

A PDFAnnotationMarkup object appears as highlighting, underlining, or a strikethrough style applied to the text of a document.

The `setLineWidth:` (page 194) and `setStyle:` (page 194) methods of the annotation's associated PDFBorder object determines the stroke thickness and style. The `setColor:` (page 124) method of the PDFAnnotation class determines the stroke color.

Tasks

Working with Markup Boundaries

- `quadrilateralPoints` (page 170)
  Gets the array of quadrilateral points defining the bounds of the markup.
- `setQuadrilateralPoints:` (page 171)
  Sets the array of quadrilateral points defining the bounds of the markup.

Working with Markup Style

- `markupType` (page 170)
  Gets the markup style.
- `setMarkupType:` (page 170)
  Sets the markup style.

Instance Methods

markupType

Gets the markup style.

```
- (PDFMarkupType)markupType
```

Discussion
Refer to "Constants" (page 171) for the available markup styles.

Availability
Available in Mac OS X v10.4 and later.

See Also
- setMarkupType: (page 170)

Declared In
PDFAnnotationMarkup.h

quadrilateralPoints

Gets the array of quadrilateral points defining the bounds of the markup.

```
- (NSArray *)quadrilateralPoints
```

Discussion
Each quadrilateral encompasses a word or a contiguous group of words. The quadrilateral points are ordered counterclockwise, with the first point closest to the origin in page space.

Page space is a 72 dpi coordinate system with the origin at the lower-left corner of the current page.

Availability
Available in Mac OS X v10.4 and later.

See Also
- setQuadrilateralPoints: (page 171)

Declared In
PDFAnnotationMarkup.h

setMarkupType:

Sets the markup style.

```
- (void)setMarkupType:(PDFMarkupType)type
```

Discussion
Refer to "Constants" (page 171) for the available markup styles.

Availability
Available in Mac OS X v10.4 and later.

See Also
- markupType (page 170)

Declared In
PDFAnnotationMarkup.h

setQuadrilateralPoints:

Sets the array of quadrilateral points defining the bounds of the markup.

```
- (void)setQuadrilateralPoints:(NSArray *)points
```

Discussion
The points defined by each quadrilateral array should encompass a word or a contiguous group of words. The quadrilateral points are ordered counterclockwise, with the first point closest to the origin in page space.

Page space is a 72 dpi coordinate system with the origin at the lower-left corner of the current page.

Availability
Available in Mac OS X v10.4 and later.

See Also
- quadrilateralPoints (page 170)

Declared In
PDFAnnotationMarkup.h

Constants

The styles available for markup annotations in PDF Kit:

Constant	Description
kPDFMarkupTypeHighlight	Highlight style for the markup. Available in Mac OS X v10.5 and later. Declared in PDFAnnotationMarkup.h.
kPDFMarkupTypeStrikeOut	Strikethrough style for the markup. Available in Mac OS X v10.5 and later. Declared in PDFAnnotationMarkup.h.
kPDFMarkupTypeUnderline	Underline style for the markup. Available in Mac OS X v10.5 and later. Declared in PDFAnnotationMarkup.h.

PDFAnnotationPopup Class Reference

Inherits from	PDFAnnotation : NSObject
Conforms to	NSObject (NSObject)
Framework	Library/Frameworks/Quartz.framework/Frameworks/PDFKit.framework
Declared in	PDFKit/PDFAnnotationPopup.h
Availability	Available in Mac OS X v10.5 and later.

Overview

A `PDFAnnotationPopup` object provides user interactivity on a PDF page in the form of a pop-up menu.

Tasks

Accessing and Setting the Open State

`- isOpen` (page 173)
Returns a Boolean value indicating whether the pop-up is open.

`- setIsOpen:` (page 174)
Sets the open state of the pop-up menu.

Instance Methods

isOpen

Returns a Boolean value indicating whether the pop-up is open.

```
- (BOOL)isOpen
```

Return Value
`YES` if the pop-up is open; `NO` otherwise.

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

See Also
- `setIsOpen:` (page 174)

Declared In
`PDFAnnotationPopup.h`

setIsOpen:

Sets the open state of the pop-up menu.

```
- (void)setIsOpen:(BOOL)isOpen
```

Parameters

isOpen

Pass `YES` to set the pop-up menu to open; `NO` otherwise.

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

See Also
- `isOpen` (page 173)

Declared In
`PDFAnnotationPopup.h`

PDFAnnotationSquare Class Reference

Inherits from	PDFAnnotation : NSObject
Conforms to	NSObject (NSObject)
Framework	Library/Frameworks/Quartz.framework/Frameworks/PDFKit.framework
Declared in	PDFKit/PDFAnnotationSquare.h
Availability	Available in Mac OS X v10.4 and later.

Overview

A PDFAnnotationSquare object displays a rectangle on a page. Square annotations are like circle annotations (instances of the PDFAnnotationCircle class) apart from the shape.

The `setLineWidth:` (page 194) and `setStyle:` (page 194) methods of the annotation's associated PDFBorder object determines the stroke thickness and style. The `setColor:` (page 124) method of the PDFAnnotation class determines the stroke color.

Tasks

Accessor Methods

- `interiorColor` (page 175)
  Gets the fill color used for drawing the annotation.
- `setInteriorColor:` (page 176)
  Sets the fill color used for drawing the annotation.

Instance Methods

interiorColor

Gets the fill color used for drawing the annotation.

```
- (NSColor *)interiorColor
```

Availability
Available in Mac OS X v10.4 and later.

See Also
- `setInteriorColor:` (page 176)

Declared In
`PDFAnnotationSquare.h`

setInteriorColor:

Sets the fill color used for drawing the annotation.

```
- (void)setInteriorColor:(NSColor *)color
```

Availability
Available in Mac OS X v10.4 and later.

See Also
- `interiorColor` (page 175)

Declared In
`PDFAnnotationSquare.h`

PDFAnnotationStamp Class Reference

Inherits from	PDFAnnotation : NSObject
Conforms to	NSObject (NSObject)
Framework	Library/Frameworks/Quartz.framework/Frameworks/PDFKit.framework
Declared in	PDFKit/PDFAnnotationStamp.h
Availability	Available in Mac OS X v10.5 and later.

Overview

A `PDFAnnotationStamp` object allows you to display a word or phrase, such as "Confidential," in a PDF page.

A `PDFAnnotationStamp` object should have an appearance stream associated with it; otherwise, nothing useful is rendered.

Tasks

Accessing and Setting the Stamp Annotation

- `name` (page 177)
  Returns name associated with the stamp annotation.
- `setName:` (page 178)
  Sets the name associated with the stamp annotation.

Instance Methods

name

Returns name associated with the stamp annotation.

```
- (NSString *)name
```

Discussion
Note that the name value of the stamp annotation is not necessarily identical to the user-visible appearance of the stamp annotation. For example, a stamp annotation that displays “Confidential” on a PDF page may not have a name value of “Confidential”.

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

See Also
`- setName:`

Declared In
`PDFAnnotationStamp.h`

setName:

Sets the name associated with the stamp annotation.

```
- (NSString *)setName:(NSString *)name
```

Discussion
The name must be representable in ASCII. You can set a stamp annotation’s name to help you identify it, but that name is not displayed on the PDF page. You must provide the string you want displayed on the page, such as “Draft” or “Top Secret”, in the appearance stream for the annotation.

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

See Also
`- name`

Declared In
`PDFAnnotationStamp.h`

PDFAnnotationText Class Reference

Inherits from	PDFAnnotation : NSObject
Conforms to	NSObject (NSObject)
Framework	Library/Frameworks/Quartz.framework/Frameworks/PDFKit.framework
Declared in	PDFKit/PDFAnnotationText.h
Availability	Available in Mac OS X v10.4 and later.

Overview

A `PDFAnnotationText` object displays as an icon (such as a "sticky note") attached to a specified point in the PDF document.

Each `PDFAnnotationText` object has a `PDFAnnotationPopup` object associated with it. In its closed state, the annotation appears as an icon. In its open state, it displays as a pop-up window containing the text of the note. Note that your application must do the work to put up a window containing the text in response to a `PDFViewAnnotationHitNotification` (page 303). Currently, text annotations do not scale and rotate with the page.

Tasks

Managing the Annotation's State

- `windowIsOpen` (page 181)
  Returns a Boolean value indicating whether the annotation's note window is open. (Deprecated. Call `isOpen` (page 173) on the annotation's pop-up instead.)
- `setWindowIsOpen:` (page 180)
  Sets the open/closed state of the annotation to the specified value. (Deprecated. Call `setIsOpen:` (page 174) on the annotation's pop-up instead.)

Managing the Annotation Icon's Type

- `iconType` (page 180)
  Returns the icon type for the annotation.

- setIconType: (page 180)
  Sets the icon type for the annotation.

Instance Methods

iconType

Returns the icon type for the annotation.

```
- (PDFTextAnnotationIconType)iconType
```

Return Value
The icon type of the annotation. See "Constants" (page 181) for a list of possible return values.

Availability
Available in Mac OS X v10.4 and later.

See Also
- setIconType: (page 180)

Declared In
PDFAnnotationText.h

setIconType:

Sets the icon type for the annotation.

```
- (void)setIconType:(PDFTextAnnotationIconType)type
```

Parameters
type
The icon type for the annotation. See "Constants" (page 181) for a list of the available icon types.

Availability
Available in Mac OS X v10.4 and later.

See Also
- iconType (page 180)

Declared In
PDFAnnotationText.h

setWindowIsOpen:

Sets the open/closed state of the annotation to the specified value. (**Deprecated.** Call setIsOpen: (page 174) on the annotation's pop-up instead.)

```
- (void)setWindowIsOpen:(BOOL)isOpen
```

Discussion
This method does not actually open or close the annotation. Use it to record annotation state.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `windowIsOpen` (page 181)

Declared In
`PDFAnnotationText.h`

windowIsOpen

Returns a Boolean value indicating whether the annotation's note window is open. (**Deprecated.** Call `isOpen` (page 173) on the annotation's pop-up instead.)

```
- (BOOL)windowIsOpen
```

Availability
Available in Mac OS X v10.4 and later.

See Also
- `setWindowIsOpen:` (page 180)

Declared In
`PDFAnnotationText.h`

Constants

Text annotations can use the following icon types:

Constant	Description
`kPDFTextAnnotationIconComment`	Comment annotation icon. Available in Mac OS X v10.4 and later. Declared in `PDFAnnotationText.h`.
`kPDFTextAnnotationIconKey`	Key annotation icon. Available in Mac OS X v10.4 and later. Declared in `PDFAnnotationText.h`.
`kPDFTextAnnotationIconNote`	Note annotation icon. Available in Mac OS X v10.4 and later. Declared in `PDFAnnotationText.h`.

Constant	Description
`kPDFTextAnnotationIconHelp`	Help annotation icon. Available in Mac OS X v10.4 and later. Declared in `PDFAnnotationText.h`.
`kPDFTextAnnotationIconNewParagraph`	New Paragraph annotation icon. Available in Mac OS X v10.4 and later. Declared in `PDFAnnotationText.h`.
`kPDFTextAnnotationIconParagraph`	Paragraph annotation icon. Available in Mac OS X v10.4 and later. Declared in `PDFAnnotationText.h`.
`kPDFTextAnnotationIconInsert`	Insert annotation icon. Available in Mac OS X v10.4 and later. Declared in `PDFAnnotationText.h`.

PDFAnnotationTextWidget Class Reference

Inherits from	PDFAnnotation : NSObject
Conforms to	NSObject (NSObject)
Framework	Library/Frameworks/Quartz.framework/Frameworks/PDFKit.framework
Declared in	PDFKit/PDFAnnotationTextWidget.h
Availability	Available in Mac OS X v10.4 and later.

Overview

A `PDFAnnotationTextWidget` object allows you to manage the appearance and content of text fields.

`PDFAnnotationTextWidget` objects support interactive forms in a PDF document. This object is comparable to an editable `NSTextField` in Cocoa or an edit text view in Carbon.

Tasks

Working with Annotation Strings

- `stringValue` (page 190)
  Returns the string assigned to the annotation.
- `setStringValue:` (page 190)
  Sets the string for the annotation.
- `maximumLength` (page 186)
  Returns the maximum number of characters allowed in the annotation string.
- `setMaximumLength:` (page 189)
  Sets the maximum number of characters allowed in the annotation string.

Managing the Font and Font Color

- `font` (page 185)
  Returns the font used for the annotation's text field.
- `setFont:` (page 188)
  Sets the font used in the text field of the annotation.

- `fontColor` (page 186)
  Returns the font color used for the annotation's text field.
- `setFontColor:` (page 189)
  Sets the font color used for the annotation's text field.

Managing Background Color, Alignment, and Rotation

- `backgroundColor` (page 185)
  Returns the background color of the annotation text field.
- `setBackgroundColor:` (page 187)
  Sets the background color of the annotation text field.
- `alignment` (page 184)
  Returns the text alignment setting for the annotation.
- `setAlignment:` (page 187)
  Sets the text alignment for the annotation.
- `rotation` (page 187)
  Returns the rotation angle of the annotation text field in degrees.
- `setRotation:` (page 189)
  Sets the rotation angle of the annotation text field in degrees.

Working with Field Names

- `fieldName` (page 185)
  Returns the internal name for the annotation text field.
- `setFieldName:` (page 188)
  Sets the internal field name for the annotation text field.

Instance Methods

alignment

Returns the text alignment setting for the annotation.

```
- (NSTextAlignment)alignment
```

Return Value
The text alignment value for the annotation. Supported alignment values are `NSLeftTextAlignment`, `NSRightTextAlignment`, and `NSCenterTextAlignment`.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `setAlignment:` (page 187)

Declared In
PDFAnnotationTextWidget.h

backgroundColor

Returns the background color of the annotation text field.

```
- (NSColor *)backgroundColor
```

Return Value
The background color of the annotation's text field.

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

See Also
- setBackgroundColor:

Declared In
PDFAnnotationTextWidget.h

fieldName

Returns the internal name for the annotation text field.

```
- (NSString *)fieldName
```

Return Value
The internal name for the annotation text field.

Discussion
Field names are optional, internal names that identify text fields in a PDF form. You use field names with the PDFActionResetForm action.

Note that multiple PDFAnnotationTextWidget objects with the same field name always have the same text associated with that field name. When text is entered into one of the objects, the text associated with that field name is changed in all objects. If you need to ensure unique text for a PDFAnnotationTextWidget object, you must give it a unique field name (you can use setFieldName: (page 188) to do this).

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

See Also
- setFIeldName:

Declared In
PDFAnnotationTextWidget.h

font

Returns the font used for the annotation's text field.

```
- (NSFont *)font
```

Return Value
The font used for text in the annotation's text field.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `setFont:` (page 188)

Declared In
`PDFAnnotationTextWidget.h`

fontColor

Returns the font color used for the annotation's text field.

```
- (NSColor *)fontColor
```

Return Value
The font color used for text in the annotation's text field.

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

See Also
- `setFontColor:`

Declared In
`PDFAnnotationTextWidget.h`

maximumLength

Returns the maximum number of characters allowed in the annotation string.

```
- (NSUInteger)maximumLength
```

Return Value
The maximum number of characters allowed in the annotations string. A return value of `0` means that there is no specified maximum.

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

See Also
- `setMaximumLength:`

Declared In
`PDFAnnotationTextWidget.h`

rotation

Returns the rotation angle of the annotation text field in degrees.

```
- (int)rotation
```

Return Value
The rotation angle of the annotation text field in degrees.

Discussion
Note that the rotation value is a positive multiple of 90, such as 0, 90, 180, or 270. The rotation of annotation text fields with negative rotation is converted to a corresponding positive rotation. For example, -90 is changed to 270.

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

See Also
`- setRotation:`

Declared In
`PDFAnnotationTextWidget.h`

setAlignment:

Sets the text alignment for the annotation.

```
- (void)setAlignment:(NSTextAlignment)alignment
```

Parameters
alignment
The text-alignment value to be used for the annotation. Possible values are `NSLeftTextAlignment`, `NSRightTextAlignment`, and `NSCenterTextAlignment`.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- alignment` (page 184)

Declared In
`PDFAnnotationTextWidget.h`

setBackgroundColor:

Sets the background color of the annotation text field.

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

Parameters
color
The color to be used in the background of the annotation's text field.

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

See Also
- backgroundColor

Declared In
PDFAnnotationTextWidget.h

setFieldName:

Sets the internal field name for the annotation text field.

```
- (void)setFieldName:(NSString *)name
```

Parameters

name
The internal field name to be used for the annotation text field.

Discussion
Field names are optional, internal names that identify text fields in a PDF form. You use field names with the PDFActionResetForm action.

Note that multiple PDFAnnotationTextWidget objects with the same field name always have the same text associated with that field name. When text is entered into one of the objects, the text associated with that field name is changed in all objects. If you need to ensure unique text for a PDFAnnotationTextWidget object, you must give it a unique field name.

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

See Also
- fieldName

Declared In
PDFAnnotationTextWidget.h

setFont:

Sets the font used in the text field of the annotation.

```
- (void)setFont:(NSFont *)font
```

Parameters

font
The font to be used in the annotation's text field.

Availability
Available in Mac OS X v10.4 and later.

See Also
- font (page 185)

Declared In
PDFAnnotationTextWidget.h

setFontColor:

Sets the font color used for the annotation's text field.

```
- (void)setFontColor:(NSColor *)color
```

Parameters

color

The font color to be used in the annotation's text field.

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

See Also
- fontColor

Declared In
PDFAnnotationTextWidget.h

setMaximumLength:

Sets the maximum number of characters allowed in the annotation string.

```
- (void)setMaximumLength:(NSUInteger)maxLen
```

Parameters

maxLen

The maximum number of characters allowed in the annotation string. Pass 0 to indicate that there is no specified maximum.

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

See Also
- maximumLength

Declared In
PDFAnnotationTextWidget.h

setRotation:

Sets the rotation angle of the annotation text field in degrees.

```
- (void)setRotation:(int)rotation
```

Parameters

rotation

The rotation angle to be applied to the annotation text field, in degrees. The rotation angle must be a positive or negative multiple of 90 (negative angles are converted to their positive equivalents; for example -90 is changed to 270).

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

See Also
- rotation

Declared In
PDFAnnotationTextWidget.h

setStringValue:

Sets the string for the annotation.

```
- (void)setStringValue:(NSString *)value
```

Parameters

value

The string to be assigned to the annotation.

Availability
Available in Mac OS X v10.4 and later.

See Also
- stringValue (page 190)

Declared In
PDFAnnotationTextWidget.h

stringValue

Returns the string assigned to the annotation.

```
- (NSString *)stringValue
```

Return Value
The string assigned to the annotation.

Availability
Available in Mac OS X v10.4 and later.

See Also
- setStringValue: (page 190)

Declared In
PDFAnnotationTextWidget.h

PDFBorder Class Reference

Inherits from	NSObject
Conforms to	NSObject (NSObject)
Framework	Library/Frameworks/Quartz.framework/Frameworks/PDFKit.framework
Declared in	PDFKit/PDFBorder.h
Availability	Available in Mac OS X v10.4 and later.

Overview

A PDFBorder object, when used, adds an optional border to an annotation. Borders are drawn completely within the annotation rectangle.

Tasks

Working with Border Styles and Characteristics

- `style` (page 195)
  Gets the border style.
- `setStyle:` (page 194)
  Sets the border style.
- `lineWidth` (page 193)
  Gets the line width for the border, in points.
- `setLineWidth:` (page 194)
  Sets the line width (in points) for the border.
- `horizontalCornerRadius` (page 192)
  Gets the horizontal corner radius (in points) used for a rounded-rectangle border.
- `setHorizontalCornerRadius:` (page 193)
  Sets the horizontal corner radius (in points) used for a rounded-rectangle border.
- `verticalCornerRadius` (page 195)
  Gets the vertical corner radius used for a rounded-rectangle border, in points.
- `setVerticalCornerRadius:` (page 194)
  Sets the vertical corner radius (in points) used for a rounded-rectangle border.

- `dashPattern` (page 192)
  Gets the dash pattern for the border as an array of NSNumber objects.
- `setDashPattern:` (page 193)
  Sets the dash pattern for the border.

Drawing Borders

- `drawInRect:` (page 192)
  Draws the border.

Instance Methods

dashPattern

Gets the dash pattern for the border as an array of NSNumber objects.

```
- (NSArray *)dashPattern
```

Discussion
Refer to the description for `NSBezierPath` for more information.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `setDashPattern:` (page 193)

Declared In
`PDFBorder.h`

drawInRect:

Draws the border.

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

Availability
Available in Mac OS X v10.4 and later.

Declared In
`PDFBorder.h`

horizontalCornerRadius

Gets the horizontal corner radius (in points) used for a rounded-rectangle border.

```
- (float)horizontalCornerRadius
```

Availability
Available in Mac OS X v10.4 and later.

See Also
- `setHorizontalCornerRadius:` (page 193)

Declared In
`PDFBorder.h`

lineWidth

Gets the line width for the border, in points.

```
- (float)lineWidth
```

Availability
Available in Mac OS X v10.4 and later.

See Also
- `setLineWidth:` (page 194)

Declared In
`PDFBorder.h`

setDashPattern:

Sets the dash pattern for the border.

```
- (void)setDashPattern:(NSArray *)pattern
```

Discussion
Provide *pattern* as an array of NSNumber objects. Refer to the description for `NSBezierPath` for more information.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `dashPattern` (page 192)

Declared In
`PDFBorder.h`

setHorizontalCornerRadius:

Sets the horizontal corner radius (in points) used for a rounded-rectangle border.

```
- (void)setHorizontalCornerRadius:(float)radius
```

Availability
Available in Mac OS X v10.4 and later.

See Also
- horizontalCornerRadius (page 192)

Declared In
PDFBorder.h

setLineWidth:

Sets the line width (in points) for the border.

```
- (void)setLineWidth:(float)width
```

Availability
Available in Mac OS X v10.4 and later.

See Also
- lineWidth (page 193)

Declared In
PDFBorder.h

setStyle:

Sets the border style.

```
- (void)setStyle:(PDFBorderStyle)style
```

Discussion
Refer to "Constants" (page 195) for the available border styles.

Availability
Available in Mac OS X v10.4 and later.

See Also
- style (page 195)

Declared In
PDFBorder.h

setVerticalCornerRadius:

Sets the vertical corner radius (in points) used for a rounded-rectangle border.

```
- (void)setVerticalCornerRadius:(float)radius
```

Availability
Available in Mac OS X v10.4 and later.

See Also
- verticalCornerRadius (page 195)

Declared In
PDFBorder.h

style

Gets the border style.

```
- (PDFBorderStyle)style
```

Discussion
See "Constants" (page 195) for possible return values.

Availability
Available in Mac OS X v10.4 and later.

See Also
- setStyle: (page 194)

Declared In
PDFBorder.h

verticalCornerRadius

Gets the vertical corner radius used for a rounded-rectangle border, in points.

```
- (float)verticalCornerRadius
```

Availability
Available in Mac OS X v10.4 and later.

See Also
- setVerticalCornerRadius: (page 194)

Declared In
PDFBorder.h

Constants

PDF Kit annotation borders may have the following styles:

Constant	Description
kPDFBorderStyleSolid	Solid border. Available in Mac OS X v10.4 and later. Declared in PDFBorder.h.
kPDFBorderStyleDashed	Dashed border. Available in Mac OS X v10.4 and later. Declared in PDFBorder.h.

Constant	Description
`kPDFBorderStyleBeveled`	Beveled border. Available in Mac OS X v10.4 and later. Declared in `PDFBorder.h`.
`kPDFBorderStyleInset`	Inset border. Available in Mac OS X v10.4 and later. Declared in `PDFBorder.h`.
`kPDFBorderStyleUnderline`	Underline border. Available in Mac OS X v10.4 and later. Declared in `PDFBorder.h`.

PDFDestination Class Reference

Inherits from	NSObject
Conforms to	NSCopying NSObject (NSObject)
Framework	Library/Frameworks/Quartz.framework/Frameworks/PDFKit.framework
Availability	Available in Mac OS X v10.4 and later.
Declared in	PDFKit/PDFDestination.h

Overview

A `PDFDestination` object describes a point on a PDF page.

In typical usage, you do not initialize `PDFDestination` objects but rather get them as either attributes of `PDFAnnotationLink` or `PDFOutline` objects, or in response to the `PDFView` method `currentDestination` (page 279).

Tasks

Initializing a Destination

- `initWithPage:atPoint:` (page 198)
  Initializes the destination.

Getting Pages and Points

- `page` (page 199)
  Returns the page that the destination refers to.
- `point` (page 199)
  Returns the point, in page space, that the destination refers to.

Getting a Relative Location

- `compare:` (page 198)
  Returns a comparison result that indicates the location of the destination in the document, relative to the current position.

Instance Methods

compare:

Returns a comparison result that indicates the location of the destination in the document, relative to the current position.

```
- (NSComparisonResult)compare:(PDFDestination *)destination
```

Parameters

destination
The destination in the document to be located.

Return Value
A comparison result, indicating the position of the passed-in destination relative to the current position.

Discussion
If *destination* is between the receiver's position and the end of the document, `compare` returns `NSOrderedAscending`; if it is between the receiver's position and the beginning of the document, `compare` returns `NSOrderedDescending`. Otherwise, if `destination` matches the receiver's position, `compare` returns `NSOrderedSame`.

This method ignores the horizontal component of the destination point (the x value). If the destination's vertical component (or y value) is `kPDFDestinationUnspecifiedValue`, `compare` treats the destination as if its y value is the top point on the destination page.

An exception is raised if *destination* does not have a page associated with it or if its page is associated with a document other than the receiver's document.

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

Declared In
`PDFDestination.h`

initWithPage:atPoint:

Initializes the destination.

```
- (id)initWithPage:(PDFPage *)page atPoint:(NSPoint)point
```

Parameters

page
The page of the destination.

point
The point of the destination, in page space.

Return Value
An initialized `PDFDestination` instance, or `NULL` if the object could not be initialized.

Discussion
Specify *point* in page space. Typically, there's no need to initialize destinations. Instead, you get them from `PDFAnnotationLink`, `PDFOutline`, or `PDFView` objects.

Page space is a 72-dpi coordinate system with the origin at the lower-left corner of the current page.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`PDFDestination.h`

page

Returns the page that the destination refers to.

```
- (PDFPage *)page
```

Return Value
The page referred to by the destination.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `point` (page 199)

Declared In
`PDFDestination.h`

point

Returns the point, in page space, that the destination refers to.

```
- (NSPoint)point
```

Return Value
The point, in page space, referred to by the destination.

Discussion
Page space is a 72 dpi coordinate system with the origin at the lower-left corner of the current page.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `page` (page 199)

Declared In
`PDFDestination.h`

Constants

Destination Undefined

Value used for unspecified destination.

```
#define kPDFDestinationUnspecifiedValue          FLT_MAX
```

Constants
`kPDFDestinationUnspecifiedValue`
Unspecified value used when a destination's actual x or y value is unimportant.

Available in Mac OS X v10.5 and later.

Declared in `PDFDestination.h`.

PDFDocument Class Reference

Inherits from	NSObject
Conforms to	NSObject (NSObject)
Framework	Library/Frameworks/Quartz.framework/Frameworks/PDFKit.framework
Availability	Available in Mac OS X v10.4 and later.
Declared in	PDFKit/PDFDocument.h

Overview

A `PDFDocument` object represents PDF data or a PDF file and defines methods for writing, searching, and selecting PDF data.

The other utility classes are either instantiated from methods in `PDFDocument`, as are `PDFPage` and `PDFOutline`; or support it, as do `PDFSelection` and `PDFDestination`.

You initialize a `PDFDocument` object with PDF data or with a URL to a PDF file. You can then ask for the page count, add or delete pages, perform a find, or parse selected content into an `NSString` object.

Tasks

Initializing Documents

- `initWithData:` (page 210)
  Initializes a `PDFDocument` object with the passed-in data.
- `initWithURL:` (page 210)
  Initializes a `PDFDocument` object with the contents at the specified URL (if the URL is invalid, this method returns `NULL`).

Accessing Document Information

- `documentURL` (page 207)
  Returns the URL for the document.
- `majorVersion` (page 212)
  Returns the major version of the document.

- `minorVersion` (page 212)
  Returns the minor version of the document.
- `string` (page 217)
  Returns a string representing the textual content for the entire document.
- `outlineItemForSelection:` (page 213)
  Returns the most likely parent PDF outline object for the selection.
- `outlineRoot` (page 213)
  Returns the root PDF outline object for the document.
- `documentAttributes` (page 207)
  Returns a dictionary of document metadata.
- `setDocumentAttributes:` (page 217)
  Sets the document attributes.
- `setOutlineRoot:` (page 217)
  Sets the document's root outline to a PDF outline object.

Managing Document Security

- `isEncrypted` (page 211)
  Returns a Boolean value specifying whether the document is encrypted.
- `isLocked` (page 212)
  Returns a Boolean value indicating whether the document is locked.
- `unlockWithPassword:` (page 218)
  Attempts to unlock an encrypted document.
- `allowsCopying` (page 204)
  Returns a Boolean value indicating whether the document allows copying of content to the Pasteboard.
- `allowsPrinting` (page 205)
  Returns a Boolean value indicating whether the document allows printing.

Writing Out the PDF Data

- `dataRepresentation` (page 206)
  Returns a representation of the document as an `NSData` object.
- `writeToFile:` (page 218)
  Writes the document to a file at the specified path.
- `writeToFile:withOptions:` (page 219)
  Writes the document to a file at the specified path with the specified options.
- `writeToURL:` (page 219)
  Writes the document to a location specified by the passed-in URL.
- `writeToURL:withOptions:` (page 220)
  Writes the document to the specified URL with the specified options.

Working with Pages

- `pageCount` (page 214)
  Returns the number of pages in the document.
- `pageAtIndex:` (page 214)
  Returns the page at the specified index number.
- `indexForPage:` (page 209)
  Gets the index number for the specified page.
- `insertPage:atIndex:` (page 210)
  Inserts a page at the specified index point.
- `removePageAtIndex:` (page 215)
  Removes the page at the specified index point.
- `exchangePageAtIndex:withPageAtIndex:` (page 208)
  Swaps one page with another.

Managing Find Operations

- `findString:withOptions:` (page 209)
  Synchronously finds all instances of the specified string in the document.
- `beginFindString:withOptions:` (page 205)
  Asynchronously finds all instances of the specified string in the document.
- `beginFindStrings:withOptions:` (page 206)
  Asynchronously finds all instances of the specified array of strings in the document.
- `findString:fromSelection:withOptions:` (page 208)
  Synchronously finds the next occurance of a string after the specified selection (or before the selection if you specified `NSBackwardsSearch` as a search option.
- `isFinding` (page 211)
  Returns a Boolean value indicating whether an asynchronous find operation is in progress.
- `cancelFindString` (page 206)
  Cancels a search initiated with `beginFindString:withOptions:` (page 205).

Working with Selections

- `selectionFromPage:atCharacterIndex:toPage:atCharacterIndex:` (page 215)
  Returns the specified selection based on starting and ending character indexes.
- `selectionFromPage:atPoint:toPage:atPoint:` (page 216)
  Returns the specified selection based on starting and ending points.
- `selectionForEntireDocument` (page 215)
  Returns a selection representing the textual content of the entire document.

Setting the Delegate

- `setDelegate:` (page 216)
  Establishes the specified object as the delegate for the `PDFDocument` object.
- `delegate` (page 207)
  Returns the object acting as the delegate for the `PDFDocument` object.

Searching Documents

- `didMatchString:` (page 220) *delegate method*
  Called for every match found during a find operation.
- `documentDidBeginDocumentFind:` (page 220) *delegate method*
  Called when the `PDFDocumentDidBeginFindNotification` notification is posted.
- `documentDidBeginPageFind:` (page 221) *delegate method*
  Called when the `PDFDocumentDidBeginPageFindNotification` notification is posted.
- `documentDidEndDocumentFind:` (page 221) *delegate method*
  Called when the `PDFDocumentDidEndFindNotification` notification is posted.
- `documentDidEndPageFind:` (page 221) *delegate method*
  Called when the `PDFDocumentDidEndPageFindNotification` notification is posted.
- `documentDidFindMatch:` (page 222) *delegate method*
  Called when the `PDFDocumentDidFindMatchNotification` notification is posted.

Unlocking Documents

- `documentDidUnlock:` (page 222) *delegate method*
  Called when the `PDFDocumentDidUnlockNotification` notification is posted.

Determining the Page Class

- `pageClass` (page 214)
  Returns the class that is allocated and initialized when page objects are created for the document.

Instance Methods

allowsCopying

Returns a Boolean value indicating whether the document allows copying of content to the Pasteboard.

```
- (BOOL)allowsCopying
```

Discussion
The ability to copy content from a PDF document is an attribute unrelated to whether the document is locked or unlocked. It depends on the PDF permissions set by the document's author.

This method only determines the desired permissions setting in the PDF document; it is up to the application to enforce (or ignore) the permissions.

This method always returns `YES` if the document is not encrypted. Note that in many cases an encrypted document may still be readable by all users due to the standard empty string password. For more details about user and owner passwords, see the Adobe PDF specification.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`PDFDocument.h`

allowsPrinting

Returns a Boolean value indicating whether the document allows printing.

```
- (BOOL)allowsPrinting
```

Discussion
The ability to print a PDF document is an attribute unrelated to whether the document is locked or unlocked. It depends on the PDF permissions set by the document's author.

This method only determines the desired permissions setting in the PDF document; it is up to the application to enforce (or ignore) the permissions.

This method always returns `YES` if the document is not encrypted. Note that in many cases an encrypted document may still be readable by all users due to the standard empty string password. For more details about user and owner passwords, see the Adobe PDF specification.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`PDFDocument.h`

beginFindString:withOptions:

Asynchronously finds all instances of the specified string in the document.

```
- (void)beginFindString:(NSString *)string withOptions:(int)options
```

Discussion
This method returns immediately. It causes notifications to be issued when searching begins and ends, on each search hit, and when the search proceeds to a new page. For options, refer to Searching and Comparing Strings.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `findString:withOptions:` (page 209)
- `isFinding` (page 211)

- cancelFindString (page 206)

Declared In
PDFDocument.h

beginFindStrings:withOptions:

Asynchronously finds all instances of the specified array of strings in the document.

```
- (void)beginFindStrings:(NSArray *)strings withOptions:(int)options;
```

Discussion
This method returns immediately. It causes notifications to be issued when searching begins and ends, on each search hit, and when the search proceeds to a new page. For options, refer to Searching and Comparing Strings.

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

See Also
- beginFindString:withOptions:
- findString:withOptions: (page 209)
- isFinding (page 211)
- cancelFindString (page 206)

Declared In
PDFDocument.h

cancelFindString

Cancels a search initiated with beginFindString:withOptions: (page 205).

```
- (void)cancelFindString
```

Availability
Available in Mac OS X v10.4 and later.

See Also
- findString:withOptions: (page 209)
- beginFindString:withOptions: (page 205)
- isFinding (page 211)

Declared In
PDFDocument.h

dataRepresentation

Returns a representation of the document as an NSData object.

```
- (NSData *)dataRepresentation
```

Availability
Available in Mac OS X v10.4 and later.

See Also
- `writeToFile:` (page 218)
- `writeToURL:` (page 219)

Declared In
`PDFDocument.h`

delegate

Returns the object acting as the delegate for the `PDFDocument` object.

```
- (id)delegate
```

Availability
Available in Mac OS X v10.4 and later.

See Also
- `setDelegate:` (page 216)

Declared In
`PDFDocument.h`

documentAttributes

Returns a dictionary of document metadata.

```
- (NSDictionary *)documentAttributes
```

Return Value
The dictionary of document metadata. The dictionary may be empty, or only some of the keys may have associated values. See "Constants" (page 222) for a list of possible key words.

Discussion
Metadata is optional for PDF documents.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `setDocumentAttributes:` (page 217)

Declared In
`PDFDocument.h`

documentURL

Returns the URL for the document.

```
- (NSURL *)documentURL
```

Return Value
The URL for the document; may return `NULL` if the document was created from an `NSData` object.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`PDFDocument.h`

exchangePageAtIndex:withPageAtIndex:

Swaps one page with another.

```
- (void)exchangePageAtIndex:(NSUInteger)indexA withPageAtIndex:(NSUInteger)indexB
```

Discussion
This method raises an exception if either *index* value is out of bounds.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `pageCount` (page 214)
- `pageAtIndex:` (page 214)
- `indexForPage:` (page 209)
- `insertPage:atIndex:` (page 210)
- `removePageAtIndex:` (page 215)

Declared In
`PDFDocument.h`

findString:fromSelection:withOptions:

Synchronously finds the next occurance of a string after the specified selection (or before the selection if you specified `NSBackwardsSearch` as a search option.

```
- (PDFSelection *)findString:(NSString *)string fromSelection:(PDFSelection
    *)selection withOptions:(int)options
```

Discussion
Matches are returned as a `PDFSelection` object. If the search reaches the end (or beginning) of the document without any hits, this method returns `NULL`.

If you pass `NULL` for the selection, this method begins searching from the beginning of the document (or the end, if you specified `NSBackwardsSearch`).

You can use this method to implement “Find Again” behavior. For options, refer to Searching and Comparing Strings.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `isFinding` (page 211)
- `findString:withOptions:` (page 209)

Declared In
`PDFDocument.h`

findString:withOptions:

Synchronously finds all instances of the specified string in the document.

```
- (NSArray *)findString:(NSString *)string withOptions:(int)options
```

Discussion
Each hit gets added to an `NSArray` object as a `PDFSelection` object. If there are no hits, this method returns an empty array.

Use this method when the complete search process will be brief and when you don't need the flexibility or control offered by `beginFindString:withOptions:` (page 205). For options, refer to Searching and Comparing Strings.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `isFinding` (page 211)
- `findString:fromSelection:withOptions:` (page 208)

Declared In
`PDFDocument.h`

indexForPage:

Gets the index number for the specified page.

```
- (NSUInteger)indexForPage:(PDFPage *)page
```

Discussion
Indexes are zero-based. This method raises an exception and returns `NSNotFound` if *page* is not found.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `pageCount` (page 214)
- `pageAtIndex:` (page 214)
- `insertPage:atIndex:` (page 210)
- `removePageAtIndex:` (page 215)
- `exchangePageAtIndex:withPageAtIndex:` (page 208)

Declared In
`PDFDocument.h`

initWithData:

Initializes a `PDFDocument` object with the passed-in data.

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

Return Value
A `PDFDocument` instance initialized with the passed-in PDF data, or `NULL` if the object could not be initialized.

Discussion
The data must be PDF data encapsulated in an `NSData` object; otherwise this method returns `NULL`.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- initWithURL:` (page 210)

Declared In
`PDFDocument.h`

initWithURL:

Initializes a `PDFDocument` object with the contents at the specified URL (if the URL is invalid, this method returns `NULL`).

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

Return Value
A `PDFDocument` instance initialized with the data at the passed-in URL or `NULL` if the object could not be initialized or if the URL is invalid.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- initWithData:` (page 210)

Declared In
`PDFDocument.h`

insertPage:atIndex:

Inserts a page at the specified index point.

```
- (void)insertPage:(PDFPage *)page atIndex:(NSUInteger)index
```

Discussion
This method raises an exception if *index* is out of bounds.

Be aware that a PDF viewing application might use the size of the first page in the document as representative of all page sizes when reporting the size of a document. If you need to get the actual size of an individual page, you can use `boundsForBox:` (page 241) (note that the size is returned in points, which are typically converted to inches or centimeters by PDF viewing applications).

Availability
Available in Mac OS X v10.4 and later.

See Also
- `pageCount` (page 214)
- `pageAtIndex:` (page 214)
- `indexForPage:` (page 209)
- `removePageAtIndex:` (page 215)
- `exchangePageAtIndex:withPageAtIndex:` (page 208)

Declared In
`PDFDocument.h`

isEncrypted

Returns a Boolean value specifying whether the document is encrypted.

```
- (BOOL)isEncrypted
```

Return Value
`YES` if the document is encrypted, whether it is locked or unlocked; `NO` otherwise.

Discussion
If encrypted, reading the document requires a password.

Encrypted documents whose password is the empty string are unlocked automatically upon opening, because PDF Kit tries the empty string as a password if none is supplied. Use the `unlockWithPassword:` (page 218) method to unlock a document using a password.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`PDFDocument.h`

isFinding

Returns a Boolean value indicating whether an asynchronous find operation is in progress.

```
- (BOOL)isFinding
```

Availability
Available in Mac OS X v10.4 and later.

See Also
- `beginFindString:withOptions:` (page 205)
- `cancelFindString` (page 206)

Declared In
`PDFDocument.h`

isLocked

Returns a Boolean value indicating whether the document is locked.

```
- (BOOL)isLocked
```

Return Value
`YES` if the document is locked; `NO` otherwise.

Discussion
Only encrypted documents can be locked. Encrypted documents whose password is the empty string are unlocked automatically upon opening, because PDF Kit tries the empty string as a password if none is supplied. Use the `unlockWithPassword:` (page 218) method to unlock a document using a password.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`PDFDocument.h`

majorVersion

Returns the major version of the document.

```
- (int)majorVersion
```

Return Value
The major version of the document.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- minorVersion` (page 212)

Declared In
`PDFDocument.h`

minorVersion

Returns the minor version of the document.

```
- (int)minorVersion
```

Return Value
The minor version of the document.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- majorVersion` (page 212)

Declared In
PDFDocument.h

outlineItemForSelection:

Returns the most likely parent PDF outline object for the selection.

```
- (PDFOutline *)outlineItemForSelection:(PDFSelection *)selection
```

Parameters

selection

The area of the document currently selected by the user. A selection can span multiple outline items, but only the point representing the first character is considered.

Return Value
The PDF outline object that is the most likely parent of the specified selection. Note that only the point representing the first character of the selection is considered in this method.

Discussion
Typically, outlines represent structural items such as chapters. You can use this method to identify the chapter that a selection falls within.

Availability
Available in Mac OS X v10.4 and later.

See Also
- outlineRoot (page 213)

Declared In
PDFDocument.h

outlineRoot

Returns the root PDF outline object for the document.

```
- (PDFOutline *)outlineRoot
```

Return Value
The root outline object or NULL if there is no root outline object. The root outline is the nonvisible top-level container for outline items.

Availability
Available in Mac OS X v10.4 and later.

See Also
- outlineItemForSelection: (page 213)
- setOutlineRoot:

Declared In
PDFDocument.h

pageAtIndex:

Returns the page at the specified index number.

```
- (PDFPage *)pageAtIndex:(NSUInteger)index
```

Discussion
Indexes are zero based. This method raises an exception if *index* is out of bounds.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `pageCount` (page 214)
- `indexForPage:` (page 209)
- `insertPage:atIndex:` (page 210)
- `removePageAtIndex:` (page 215)
- `exchangePageAtIndex:withPageAtIndex:` (page 208)

Declared In
`PDFDocument.h`

pageClass

Returns the class that is allocated and initialized when page objects are created for the document.

```
- (Class)pageClass
```

Discussion
If you want to supply a custom page class, subclass `PDFDocument` and implement this method to return your custom class. Note that your custom class must be a subclass of `PDFPage`; otherwise, the behavior is undefined.

The default implementation of `pageClass` returns `[PDFPage class]`.

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

Declared In
`PDFDocument.h`

pageCount

Returns the number of pages in the document.

```
- (NSUInteger)pageCount
```

Availability
Available in Mac OS X v10.4 and later.

See Also
- `pageAtIndex:` (page 214)
- `indexForPage:` (page 209)

- insertPage:atIndex: (page 210)
- removePageAtIndex: (page 215)
- exchangePageAtIndex:withPageAtIndex: (page 208)

Declared In
PDFDocument.h

removePageAtIndex:

Removes the page at the specified index point.

```
- (void)removePageAtIndex:(NSUInteger)index
```

Discussion
This method raises an exception if *index* is out of bounds.

Availability
Available in Mac OS X v10.4 and later.

See Also
- pageCount (page 214)
- pageAtIndex: (page 214)
- indexForPage: (page 209)
- insertPage:atIndex: (page 210)
- exchangePageAtIndex:withPageAtIndex: (page 208)

Declared In
PDFDocument.h

selectionForEntireDocument

Returns a selection representing the textual content of the entire document.

```
- (PDFSelection *)selectionForEntireDocument
```

Availability
Available in Mac OS X v10.4 and later.

Declared In
PDFDocument.h

selectionFromPage:atCharacterIndex:toPage:atCharacterIndex:

Returns the specified selection based on starting and ending character indexes.

```
- (PDFSelection *)selectionFromPage:(PDFPage *)startPage
    atCharacterIndex:(NSUInteger)startChar toPage:(PDFPage *)endPage
    atCharacterIndex:(NSUInteger)endChar
```

Discussion
The selection begins at *startChar* on *startPage* and ends at *endChar* on *endPage*. The starting and ending index values must be in the range of the number of characters (as returned by `numberOfCharacters` (page 244)) within the respective `PDFPage` objects.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `selectionFromPage:atPoint:toPage:atPoint:` (page 216)

Declared In
`PDFDocument.h`

selectionFromPage:atPoint:toPage:atPoint:

Returns the specified selection based on starting and ending points.

```
- (PDFSelection *)selectionFromPage:(PDFPage *)startPage atPoint:(NSPoint)startPt
    toPage:(PDFPage *)endPage atPoint:(NSPoint)endPt
```

Discussion
The selection begins at *startPt* on *startPage* and ends at *endPt* on *endPage*. The starting and ending points should be specified in page space, relative to their respective pages.

The starting and ending points can be on the same page. In this case, invoking this method is equivalent to sending the `selectionFromPoint:toPoint:` message to a `PDFPage` object.

Page space is a 72 dpi coordinate system with the origin at the lower-left corner of the current page.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `selectionFromPage:atCharacterIndex:toPage:atCharacterIndex:` (page 215)
- `selectionForRange:` (page 246)

Declared In
`PDFDocument.h`

setDelegate:

Establishes the specified object as the delegate for the `PDFDocument` object.

```
- (void)setDelegate:(id)anObject
```

Availability
Available in Mac OS X v10.4 and later.

See Also
- `delegate` (page 207)
- `didMatchString:` (page 220)

Declared In
PDFDocument.h

setDocumentAttributes:

Sets the document attributes.

```
- (void)setDocumentAttributes:(NSDictionary *)attributes
```

Parameters

attributes

A dictionary containing document attributes as key-value pairs. See "Constants" (page 222) for a list of possible key words.

Availability
Available in Mac OS X v10.4 and later.

See Also
- documentAttributes (page 207).

Declared In
PDFDocument.h

setOutlineRoot:

Sets the document's root outline to a PDF outline object.

```
- (void)setOutlineRoot:(PDFOutline *)outline
```

Parameters

outline

The outline to be used as the document's root outline. Pass NULL to strip the outline from a document.

Discussion
When a PDF document is saved, the outline tree structure is written out to the destination PDF file.

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

See Also
- outlineRoot

Declared In
PDFDocument.h

string

Returns a string representing the textual content for the entire document.

```
- (NSString *)string
```

Return Value
A string that represents the textual content of the entire document.

Discussion
Pages are delimited with linefeed characters.

This is a convenience method, equivalent to creating a selection object for the entire document and then invoking the `PDFSelection` class's `string` (page 257) method.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`PDFDocument.h`

unlockWithPassword:

Attempts to unlock an encrypted document.

```
- (BOOL)unlockWithPassword:(NSString *)password
```

Parameters

password

The password to unlock an encrypted document (you cannot lock an unlocked PDF document by using an incorrect password).

Return Value
`YES` if the specified password unlocks the document, `NO` otherwise.

Discussion
If the password is correct, this method returns `YES`, and a `PDFDocumentDidUnlockNotification` notification is sent. Once unlocked, you cannot use this function to relock the document.

If you attempt to unlock an already unlocked document, one of the following occurs:

- If the document is unlocked with full owner permissions, `unlockWithPassword` does nothing and returns `YES`. The password string is ignored.
- If the document is unlocked with only user permissions, `unlockWithPassword` attempts to obtain full owner permissions with the password string. If the string fails, the document maintains its user permissions. In either case, this method returns `YES`.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`PDFDocument.h`

writeToFile:

Writes the document to a file at the specified path.

```
- (BOOL)writeToFile:(NSString *)path
```

Availability
Available in Mac OS X v10.4 and later.

See Also
- dataRepresentation (page 206)
- writeToURL: (page 219)
- writeToURL:withOptions: (page 220)
- writeToFile:withOptions: (page 219)

Declared In
PDFDocument.h

writeToFile:withOptions:

Writes the document to a file at the specified path with the specified options.

```
- (BOOL)writeToFile:(NSString *)path withOptions:(NSDictionary *)options
```

Discussion
The most commonly-used options are kCGPDFContextOwnerPassword, kCGPDFContextUserPassword, kCGPDFContextAllowsCopying and kCGPDFContextAllowsPrinting. For more details about these options, see the "Auxiliary Dictionary Keys" in *CGPDFContext Reference*, part of the Quartz 2D Reference.

Availability
Available in Mac OS X v10.4 and later.

See Also
- dataRepresentation (page 206)
- writeToURL: (page 219)
- writeToURL:withOptions: (page 220)
- writeToFile: (page 218)

Declared In
PDFDocument.h

writeToURL:

Writes the document to a location specified by the passed-in URL.

```
- (BOOL)writeToURL:(NSURL *)url
```

Availability
Available in Mac OS X v10.4 and later.

See Also
- dataRepresentation (page 206)
- writeToFile: (page 218)
- writeToFile:withOptions: (page 219)
- writeToURL:withOptions: (page 220)

Declared In
PDFDocument.h

writeToURL:withOptions:

Writes the document to the specified URL with the specified options.

```
- (BOOL)writeToURL:(NSURL *)url withOptions:(NSDictionary *)options
```

Availability
Available in Mac OS X v10.4 and later.

See Also
- `dataRepresentation` (page 206)
- `writeToURL:` (page 219)
- `writeToFile:` (page 218)
- `writeToFile:withOptions:` (page 219)

Declared In
`PDFDocument.h`

Delegate Methods

didMatchString:

Called for every match found during a find operation.

```
- (void)didMatchString:(PDFSelection *)instance
```

Availability
Available in Mac OS X v10.4 and later.

See Also
- `findString:withOptions:` (page 209)
- `setDelegate:` (page 216)

Declared In
`PDFDocument.h`

documentDidBeginDocumentFind:

Called when the `PDFDocumentDidBeginFindNotification` notification is posted.

```
- (void)documentDidBeginDocumentFind:(NSNotification *)notification
```

Availability
Available in Mac OS X v10.4 and later.

See Also
`PDFDocumentDidBeginFindNotification`
- `setDelegate:` (page 216)

Declared In
PDFDocument.h

documentDidBeginPageFind:

Called when the PDFDocumentDidBeginPageFindNotification notification is posted.

```
- (void)documentDidBeginPageFind:(NSNotification *)notification
```

Availability
Available in Mac OS X v10.4 and later.

See Also
PDFDocumentDidBeginPageFindNotification
- setDelegate: (page 216)

Declared In
PDFDocument.h

documentDidEndDocumentFind:

Called when the PDFDocumentDidEndFindNotification notification is posted.

```
- (void)documentDidEndDocumentFind:(NSNotification *)notification
```

Availability
Available in Mac OS X v10.4 and later.

See Also
PDFDocumentDidEndFindNotification
- setDelegate: (page 216)

Declared In
PDFDocument.h

documentDidEndPageFind:

Called when the PDFDocumentDidEndPageFindNotification notification is posted.

```
- (void)documentDidEndPageFind:(NSNotification *)notification
```

Availability
Available in Mac OS X v10.4 and later.

See Also
PDFDocumentDidEndPageFindNotification
- setDelegate: (page 216)

Declared In
PDFDocument.h

documentDidFindMatch:

Called when the `PDFDocumentDidFindMatchNotification` notification is posted.

```
- (void)documentDidFindMatch:(NSNotification *)notification
```

Availability
Available in Mac OS X v10.4 and later.

See Also
`PDFDocumentDidFindMatchNotification`
`- setDelegate:` (page 216)

Declared In
`PDFDocument.h`

documentDidUnlock:

Called when the `PDFDocumentDidUnlockNotification` notification is posted.

```
- (void)documentDidUnlock:(NSNotification *)notification
```

Availability
Available in Mac OS X v10.4 and later.

See Also
`PDFDocumentDidUnlockNotification`
`- setDelegate:` (page 216)

Declared In
`PDFDocument.h`

Constants

PDFPrintScalingMode

The type of scaling to be used when printing a page (see "`PDF Page Scaling Modes for Printing`").

```
typedef NSInteger PDFPrintScalingMode;
```

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

Declared In
`PDFDocument.h`

Document Attribute Keys

Keys for the document attributes dictionary. See `documentAttributes` (page 207) and `setDocumentAttributes:` (page 217).

```
extern NSString *PDFDocumentTitleAttribute;
extern NSString *PDFDocumentAuthorAttribute;
extern NSString *PDFDocumentSubjectAttribute;
extern NSString *PDFDocumentCreatorAttribute;
extern NSString *PDFDocumentProducerAttribute;
extern NSString *PDFDocumentCreationDateAttribute;
extern NSString *PDFDocumentModificationDateAttribute;
extern NSString *PDFDocumentKeywordsAttribute;
```

Constants

`PDFDocumentTitleAttribute`
An optional text string (an `NSString`) containing the title of the document.

Available in Mac OS X v10.4 and later.

Declared in `PDFDocument.h`.

`PDFDocumentAuthorAttribute`
An optional text string (an `NSString`) containing the name of the author of the document.

Available in Mac OS X v10.4 and later.

Declared in `PDFDocument.h`.

`PDFDocumentSubjectAttribute`
An optional text string (an `NSString`) containing a description of the subject of the document.

Available in Mac OS X v10.4 and later.

Declared in `PDFDocument.h`.

`PDFDocumentCreatorAttribute`
An optional text string (an `NSString`) containing the name of the application that created the document content.

Available in Mac OS X v10.4 and later.

Declared in `PDFDocument.h`.

`PDFDocumentProducerAttribute`
An optional text string (an `NSString`) containing the name of the application that produced the PDF data for the document.

Available in Mac OS X v10.4 and later.

Declared in `PDFDocument.h`.

`PDFDocumentCreationDateAttribute`
An optional text string (an `NSDate`) containing the document's creation date.

Available in Mac OS X v10.4 and later.

Declared in `PDFDocument.h`.

`PDFDocumentModificationDateAttribute`
An optional text string (an `NSDate`) containing the document's last-modified date.

Available in Mac OS X v10.4 and later.

Declared in `PDFDocument.h`.

`PDFDocumentKeywordsAttribute`
An optional array of text strings (an `NSArray` of `NSString` objects) containing keywords for the document.

Available in Mac OS X v10.4 and later.

Declared in `PDFDocument.h`.

Declared In
PDFDocument.h

PDF Page Scaling Modes for Printing

Modes that specify how the page should be scaled when printing. See the PDFView method printWithInfo:autoRotate:pageScaling: (page 289).

```
enum { kPDFPrintPageScaleNone = 0,      kPDFPrintPageScaleToFit = 1,
kPDFPrintPageScaleDownToFit = 2 };
```

Constants

kPDFPrintPageScaleNone
Do not apply scaling to the page when printing.
Available in Mac OS X v10.5 and later.
Declared in PDFDocument.h.

kPDFPrintPageScaleToFit
Scale each page up or down to best fit the paper size.
Available in Mac OS X v10.5 and later.
Declared in PDFDocument.h.

kPDFPrintPageScaleDownToFit
Scale large pages down to fit the paper size (smaller pages do not get scaled up).
Available in Mac OS X v10.5 and later.
Declared in PDFDocument.h.

Declared In
PDFDocument.h

Notifications

PDFDocument declares and posts the following notifications:

PDFDocumentDidUnlockNotification

Posted when a document unlocks after a unlockWithPassword: (page 218) message.

The notification object is the PDFDocument object itself.

Availability
Available in Mac OS X v10.4 and later.

Declared In
PDFDocument.h

PDFDocumentDidBeginFindNotification

Posted when the beginFindString:withOptions: (page 205) or findString:withOptions: (page 209) method begins finding.

The notification object is the `PDFDocument` object itself.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`PDFDocument.h`

PDFDocumentDidEndFindNotification

Posted when the `beginFindString:withOptions:` (page 205) or `findString:withOptions:` (page 209) method returns.

The `beginFindString:withOptions:` (page 205) method returns immediately, so this notification is posted when the "find" operation is finished.

You can use this notification to know when to close or hide a progress bar.

The notification object is the `PDFDocument` object itself.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`PDFDocument.h`

PDFDocumentDidBeginPageFindNotification

Posted each time a find operation begins working on a new page of a document.

You can use this notification to update a progress bar.

The notification object is the `PDFDocument` object itself. To determine the page, use the `@"PDFDocumentPageIndex"` key to obtain userinfo of type `NSNumber`.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`PDFDocument.h`

PDFDocumentDidEndPageFindNotification

Posted each time a find operation finishes working on a page in a document.

You can use this notification to update a progress bar.

The notification object is the `PDFDocument` object itself. To determine the page, use the `@"PDFDocumentPageIndex"` key to obtain userinfo of type `NSNumber`.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`PDFDocument.h`

PDFDocumentDidFindMatchNotification

Posted each time a string match is found in a document.

The notification object is the `PDFDocument` object itself. To determine the string selection found, use the `@"PDFDocumentFoundSelection"` key to obtain userinfo of type `PDFSelection *`

Availability
Available in Mac OS X v10.4 and later.

Declared In
`PDFDocument.h`

PDFDocumentDidBeginWriteNotification

Posted each time a write operation begins working on a document.

The notification object is the `PDFDocument` object itself.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`PDFDocument.h`

PDFDocumentDidEndWriteNotification

Posted each time a write operation finishes working on a document.

The notification object is the `PDFDocument` object itself.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`PDFDocument.h`

PDFDocumentDidBeginPageWriteNotification

Posted each time a write operation begins working on a page in a document.

The notification object is the `PDFDocument` object itself. To determine the page, use the `@"PDFDocumentPageIndex"` key to obtain userinfo of type `NSNumber`.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`PDFDocument.h`

PDFDocumentDidEndPageWriteNotification

Posted each time a write operation finishes working on a page in a document.

The notification object is the `PDFDocument` object itself. To determine the page, use the `@"PDFDocumentPageIndex"` key to obtain userinfo of type `NSNumber`.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`PDFDocument.h`

PDFOutline Class Reference

Inherits from	NSObject
Conforms to	NSObject (NSObject)
Framework	Library/Frameworks/Quartz.framework/Frameworks/PDFKit.framework
Declared in	PDFKit/PDFOutline.h
Availability	Available in Mac OS X v10.4 and later.

Overview

A `PDFOutline` object is an element in a tree-structured hierarchy that can represent the structure of a PDF document.

An outline is an optional component of a PDF document, useful for viewing the structure of the document and for navigating within it.

Outlines are created by the document's author. If you represent a PDF document outline using outline objects, the root of the hierarchy is obtained from the PDF document itself. This root outline is not visible and serves merely as a container for the visible outlines.

Tasks

Initializing an Outline

- `init` (page 233)
  Initializes a `PDFOutline` object.
- `initWithDocument:` (page 233)
  Initializes an outline with the specified PDF document. (Deprecated. Use the `PDFDocument outlineRoot` (page 213) method instead.)

Getting Information About an Outline

- `document` (page 232)
  Returns the document with which the outline is associated.

- `numberOfChildren` (page 234)
  Returns the number of child outline objects in the outline.
- `parent` (page 235)
  Returns the parent outline object of the outline (returns `NULL` if called on the root outline object).
- `childAtIndex:` (page 231)
  Returns the child outline object at the specified index.
- `index` (page 232)
  Returns the index of the outline.

Managing Outline Labels

- `label` (page 234)
  Returns the label for the outline.
- `setLabel:` (page 236)
  Sets the label for the outline (has no effect on the root outline object).

Managing Actions and Destinations

- `destination` (page 232)
  Returns the destination associated with the outline.
- `action` (page 231)
  Returns the action performed when users click the outline.
- `setAction:` (page 235)
  Sets the action associated with the outline.
- `setDestination:` (page 236)
  Sets the destination associated with the outline.

Changing an Outline Hierarchy

- `insertChild:atIndex:` (page 233)
  Inserts the specified outline object at the specified index.
- `removeFromParent` (page 235)
  Removes the outline object from its parent (does nothing if outline object is the root outline object).

Managing the Disclosure of an Outline Object

- `isOpen` (page 234)
  Returns a Boolean value that indicates whether the outline object is initially disclosed.
- `setIsOpen:` (page 236)
  Sets the initial disclosure state of the outline object.

Instance Methods

action

Returns the action performed when users click the outline.

```
- (PDFAction *)action
```

Discussion
The root outline serves only as a container for the outlines it owns; it does not have an action. Note that a `PDFOutline` object can have either an action or a destination, not both.

If the `PDFOutline` object has a destination, instead of an action, `action` returns a `PDFActionGoTo` object (this is equivalent to calling `destination` (page 232) on the `PDFOutline` object). For other action types, `action` returns the appropriate PDF Kit action type object, such as `PDFActionURL`.

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

See Also
`- setAction:` (page 235)

Declared In
`PDFOutline.h`

childAtIndex:

Returns the child outline object at the specified index.

```
- (PDFOutline *)childAtIndex:(NSUInteger)index
```

Discussion
The index is zero-based. This method throws an exception if *index* is out of range.

Important: In Mac OS X v10.5 and later, a `PDFOutline` object retains all its children, so `childAtIndex:` returns the same retained child outline object every time it's called. This means that you do not need to retain the object returned by `childAtIndex:`. This differs from the behavior of `PDFOutline` in Mac OS X v10.4. In Tiger, `childAtIndex:` returns an auto-released, one-off child outline object, when meant that you had to include code to retain the child.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- index` (page 232)

Declared In
`PDFOutline.h`

destination

Returns the destination associated with the outline.

```
- (PDFDestination *)destination
```

Discussion
The root outline serves only as a container for the outlines it owns; it does not have a destination. Note that a `PDFOutline` object can have either a destination or an action, not both.

This method may return `NULL` if the outline has an associated action instead of a destination. Note that if the associated action is a `PDFActionGoTo`, this method returns the destination from the `PDFActionGoTo` object. However, it is better to use the `action` (page 231) method for this purpose.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- setDestination:` (page 236)

Declared In
`PDFOutline.h`

document

Returns the document with which the outline is associated.

```
- (PDFDocument *)document
```

Availability
Available in Mac OS X v10.4 and later.

Declared In
`PDFOutline.h`

index

Returns the index of the outline.

```
- (NSUInteger)index
```

Discussion
The index of the outline object is relative to its siblings and from the perspective of the parent of the outline object. The root outline object, and any outline object without a parent, has an index value of `0`.

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

Declared In
`PDFOutline.h`

init

Initializes a `PDFOutline` object.

```
- (id)init
```

Discussion
If you want the `PDFOutline` object returned by this method to be the outline root, you must add additional `PDFOutline` objects to create the outline hierarchy you desire. Then, you must add the root outline object to your PDF document by passing it to the `PDFDocument` `setOutlineRoot:` (page 217) method.

If you want the `PDFOutline` object returned by this method to be a child of an existing outline, you must use `setLabel:` (page 236) to give it a label and give it either a destination or action using `setDestination:` (page 236) or `setAction:` (page 235), respectively. In addition, you must add this outline object to the existing `PDFOutline` object as a new child, using `insertChild:atIndex:` (page 233)

Availability
Available in Mac OS X v10.4 and later.

Declared In
`PDFOutline.h`

initWithDocument:

Initializes an outline with the specified PDF document. (**Deprecated.** Use the `PDFDocument` `outlineRoot` (page 213) method instead.)

```
- (id)initWithDocument:(PDFDocument *)document
```

Discussion
Returns `NULL` if the document does not contain an outline. Invoking this method is equivalent to sending the `outlineRoot` message to a `PDFDocument` object.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`PDFOutline.h`

insertChild:atIndex:

Inserts the specified outline object at the specified index.

```
- (void)insertChild:(PDFOutline *)child atIndex:(NSUInteger)index
```

Discussion
To build a PDF outline hierarchy, use this method to add child outline objects. Before you call this method on a `PDFOutline` object that already has a parent, you should retain the object and call `removeFromParent` (page 235) on it first.

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

See Also
- `childAtIndex:` (page 231)

Declared In
`PDFOutline.h`

isOpen

Returns a Boolean value that indicates whether the outline object is initially disclosed.

```
- (BOOL)isOpen
```

Discussion
Calling `isOpen` on an outline object that has no children always returns `NO`. Calling `isOpen` on the root outline object always returns `YES`.

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

See Also
- `setIsOpen:` (page 236)

Declared In
`PDFOutline.h`

label

Returns the label for the outline.

```
- (NSString *)label
```

Discussion
The root outline serves only as a container for the outlines it owns; it does not have a label.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `setLabel:` (page 236)

Declared In
`PDFOutline.h`

numberOfChildren

Returns the number of child outline objects in the outline.

```
- (NSUInteger)numberOfChildren
```

Availability
Available in Mac OS X v10.4 and later.

See Also
- childAtIndex: (page 231)

Declared In
PDFOutline.h

parent

Returns the parent outline object of the outline (returns NULL if called on the root outline object).

```
- (PDFOutline *)parent
```

Availability
Available in Mac OS X v10.5 and later

Declared In
PDFOutline.h

removeFromParent

Removes the outline object from its parent (does nothing if outline object is the root outline object).

```
- (void)removeFromParent
```

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

See Also
- parent (page 235)

Declared In
PDFOutline.h

setAction:

Sets the action associated with the outline.

```
- (void)setAction:(PDFAction *)action
```

Discussion
Calling setAction on the root outline object has no effect, because the root outline does not have an action or a destination..

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

See Also
- action (page 231)

Declared In
PDFOutline.h

setDestination:

Sets the destination associated with the outline.

```
- (void)setDestination:(PDFDestination *)destination
```

Discussion
Calling `setDestination` on the root outline object has no effect, because the root outline does not have an action or a destination.

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

See Also
- `destination` (page 232)

Declared In
`PDFOutline.h`

setIsOpen:

Sets the initial disclosure state of the outline object.

```
- (void)setIsOpen:(BOOL)open
```

Discussion
Calling `setIsOpen` on an outline object with no children or on the root outline object has no effect.

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

See Also
- `isOpen` (page 234)

Declared In
`PDFOutline.h`

setLabel:

Sets the label for the outline (has no effect on the root outline object).

```
- (void)setLabel:(NSString *)label
```

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

See Also
- `label` (page 234)

Declared In
`PDFOutline.h`

PDFPage Class Reference

Inherits from	NSObject
Conforms to	NSCopying NSObject (NSObject)
Framework	Library/Frameworks/Quartz.framework/Frameworks/PDFKit.framework
Declared in	PDFKit/PDFPage.h
Availability	Available in Mac OS X v10.4 and later.

Overview

`PDFPage`, a subclass of `NSObject`, defines methods used to render PDF pages and work with annotations, text, and selections.

`PDFPage` objects are flexible and powerful. With them you can render PDF content onscreen or to a printer, add annotations, count characters, define selections, and get the textual content of a page as an `NSString` object.

Your application instantiates a `PDFPage` object by asking for one from a `PDFDocument` object.

For simple display and navigation of PDF documents within your application, you don't need to use `PDFPage`. You need only use `PDFView`.

Tasks

Initializing a Page

`- initWithDocument:` (page 243)
Initializer for subclasses of PDFPage. (Deprecated. Use `[PDFPage init]` or `initWithImage:` instead.)

`- initWithImage:` (page 244)
Creates a new `PDFPage` object and initializes it with the specified `NSImage` object.

Getting Information About a Page

`- document` (page 243)
Returns the `PDFDocument` object with which the page is associated.

- `label` (page 244)
  Returns the label for the page.
- `boundsForBox:` (page 241)
  Returns the bounds for the specified PDF display box.
- `setBounds:forBox:` (page 248)
  Sets the bounds for the specified box.
- `rotation` (page 245)
  Returns the page rotation angle in degrees.
- `setRotation:` (page 248)
  Sets the rotation angle for the page in degrees.

Working with Annotations

- `annotations` (page 240)
  Returns an array containing the page's annotations.
- `displaysAnnotations` (page 242)
  Returns a Boolean value indicating whether annotations are displayed for the page.
- `setDisplaysAnnotations:` (page 248)
  Specifies whether or not to display annotations for the page.
- `addAnnotation:` (page 239)
  Adds the specified annotation object to the page.
- `removeAnnotation:` (page 245)
  Removes the specified annotation from the page.
- `annotationAtPoint:` (page 240)
  Returns the annotation, if there is one, at the specified point.

Rendering Pages

- `drawWithBox:` (page 243)
  Draws the page within the specified box.
- `transformContextForBox:` (page 249)
  Transforms the current context, given the specified box.

Working with Textual Content

- `numberOfCharacters` (page 244)
  Returns the number of characters on the page, including whitespace characters.
- `string` (page 249)
  Returns an `NSString` object representing the text on the page.
- `attributedString` (page 240)
  Returns an `NSAttributedString` object representing the text on the page.
- `characterBoundsAtIndex:` (page 241)
  Returns the bounds, in page space, of the character at the specified index.

- `characterIndexAtPoint:` (page 242)
  Returns the character index value for the specified point in page space.

Working with Selections

- `selectionForRect:` (page 246)
  Returns the text enclosed within the specified rectangle, expressed in page (user) coordinates.
- `selectionForWordAtPoint:` (page 247)
  Returns the whole word that includes the specified point.
- `selectionForLineAtPoint:` (page 245)
  Returns the whole line of text that includes the specified point.
- `selectionFromPoint:toPoint:` (page 247)
  Returns the text between the two specified points in page space.
- `selectionForRange:` (page 246)
  Returns the text contained within the specified range.

Miscellaneous

- `dataRepresentation` (page 242)
  Returns the PDF data (that is, a PDF document) representing this page. This method does not preserve external page links.

Instance Methods

addAnnotation:

Adds the specified annotation object to the page.

```
- (void)addAnnotation:(PDFAnnotation *)annotation
```

Availability
Available in Mac OS X v10.4 and later.

See Also
- `annotations` (page 240)
- `displaysAnnotations` (page 242)
- `setDisplaysAnnotations:` (page 248)
- `removeAnnotation:` (page 245)
- `annotationAtPoint:` (page 240)

Declared In
`PDFPage.h`

annotationAtPoint:

Returns the annotation, if there is one, at the specified point.

```
- (PDFAnnotation *)annotationAtPoint:(NSPoint)point
```

Discussion
Use this method for hit-testing based on the current cursor position. If more than one annotation shares the specified point, the frontmost (or topmost) one is returned (the annotations are searched in reverse order of their appearance in the PDF data file). Returns `NULL` if there is no annotation at *point*.

Specify the point in page space. Page space is a 72 dpi coordinate system with the origin at the lower-left corner of the current page.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `annotations` (page 240)
- `displaysAnnotations` (page 242)
- `setDisplaysAnnotations:` (page 248)
- `addAnnotation:` (page 239)
- `removeAnnotation:` (page 245)

Declared In
`PDFPage.h`

annotations

Returns an array containing the page's annotations.

```
- (NSArray *)annotations
```

Discussion
The elements of the array will most likely be typed to subclasses of the `PDFAnnotation` class.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `displaysAnnotations` (page 242)
- `setDisplaysAnnotations:` (page 248)
- `addAnnotation:` (page 239)
- `removeAnnotation:` (page 245)
- `annotationAtPoint:` (page 240)

Declared In
`PDFPage.h`

attributedString

Returns an `NSAttributedString` object representing the text on the page.

```
- (NSAttributedString *)attributedString
```

Availability
Available in Mac OS X v10.4 and later.

See Also
- `numberOfCharacters` (page 244)
- `string` (page 249)

Declared In
`PDFPage.h`

boundsForBox:

Returns the bounds for the specified PDF display box.

```
- (NSRect)boundsForBox:(PDFDisplayBox)box
```

Discussion
The `PDFDisplayBox` enumeration defines the various box types (see "Constants" (page 250) for additional information about box types).

Note that only the media box is required for a PDF. If you request the bounds for the crop box, but the PDF does not include a crop box, the bounds for the media box are returned instead. If you request the bounds for other box types, and the PDF does not includes these types, the bounds for the crop box are returned instead.

The coordinates for the box are in page space, so you might need to transform the points if the page has a rotation on it. Also, note that the bounds `boundsForBox` returns are intersected with the page's media box.

`boundsForBox` throws a range exception if *box* is not in range.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `setBounds:forBox:`

Declared In
`PDFPage.h`

characterBoundsAtIndex:

Returns the bounds, in page space, of the character at the specified index.

```
- (NSRect)characterBoundsAtIndex:(NSInteger)index
```

Discussion
In the unlikely event that there is more than one character at the specified index point, only the bounds of the first character is returned.

Page space is a 72 dpi coordinate system with the origin at the lower-left corner of the current page. Note that the bounds returned are not guaranteed to have integer coordinates.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `characterIndexAtPoint:` (page 242)

Declared In
`PDFPage.h`

characterIndexAtPoint:

Returns the character index value for the specified point in page space.

```
- (NSInteger)characterIndexAtPoint:(NSPoint)point
```

Discussion
If there is no character at the specified point, the method returns `-1`.

Page space is a 72 dpi coordinate system with the origin at the lower-left corner of the current page.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `characterBoundsAtIndex:` (page 241)

Declared In
`PDFPage.h`

dataRepresentation

Returns the PDF data (that is, a PDF document) representing this page. This method does not preserve external page links.

```
- (NSData *)dataRepresentation
```

Availability
Available in Mac OS X v10.4 and later.

Declared In
`PDFPage.h`

displaysAnnotations

Returns a Boolean value indicating whether annotations are displayed for the page.

```
- (BOOL)displaysAnnotations
```

Discussion
If `YES`, the page will draw annotations when a drawing method is called.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `annotations` (page 240)
- `setDisplaysAnnotations:` (page 248)
- `addAnnotation:` (page 239)
- `removeAnnotation:` (page 245)
- `annotationAtPoint:` (page 240)
- `drawWithBox:` (page 243)

Declared In
`PDFPage.h`

document

Returns the `PDFDocument` object with which the page is associated.

```
- (PDFDocument *)document
```

Availability
Available in Mac OS X v10.4 and later.

Declared In
`PDFPage.h`

drawWithBox:

Draws the page within the specified box.

```
- (void)drawWithBox:(PDFDisplayBox)box
```

Discussion
This method takes into account the page rotation and draws clipped to the specified box. If the page is set to display annotations, this method also draws them. This method does not clear the background. To clear the background before drawing, use `NSRectFill` with `NSColor` set (typically) to white.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `displaysAnnotations` (page 242)

Declared In
`PDFPage.h`

initWithDocument:

Initializer for subclasses of PDFPage. (**Deprecated.** Use `[PDFPage init]` or `initWithImage:` instead.)

```
- (id)initWithDocument:(PDFDocument *)document
```

Discussion
Subclasses of `PDFPage` must handle several methods that are transparently handled when using the `PDFPage` class directly, including `boundsForBox` and `drawInRect:withBox:`.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`PDFPage.h`

initWithImage:

Creates a new `PDFPage` object and initializes it with the specified `NSImage` object.

```
- (id)initWithImage:(NSImage *)image
```

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

Declared In
`PDFPage.h`

label

Returns the label for the page.

```
- (NSString *)label
```

Discussion
Typically, the label is “1” for the first page, “2” for the second page, and so on, but nonnumerical labels are also possible (such as “xxi”, “4-1” and so on).

Availability
Available in Mac OS X v10.4 and later.

See Also
- `document` (page 243)

Declared In
`PDFPage.h`

numberOfCharacters

Returns the number of characters on the page, including whitespace characters.

```
- (NSUInteger)numberOfCharacters
```

Availability
Available in Mac OS X v10.4 and later.

See Also
- `string` (page 249)

Declared In
PDFPage.h

removeAnnotation:

Removes the specified annotation from the page.

```
- (void)removeAnnotation:(PDFAnnotation *)annotation
```

Availability
Available in Mac OS X v10.4 and later.

See Also
- annotations (page 240)
- displaysAnnotations (page 242)
- setDisplaysAnnotations: (page 248)
- addAnnotation: (page 239)
- annotationAtPoint: (page 240)

Declared In
PDFPage.h

rotation

Returns the page rotation angle in degrees.

```
- (int)rotation
```

Discussion
The rotation is a positive multiple of 90: 0, 90, 180, or 270. The rotation of pages with negative rotation is converted to a corresponding positive rotation.

If you are subclassing PDFView and displaying pages yourself, don't assume a rotation of 0. Pages with an inherent rotation display rotated when opened unless you set their rotation to zero. Regardless of the inherent rotation angle, it is up to the author of a page whether zero rotation corresponds to upright text when displayed on a monitor.

Availability
Available in Mac OS X v10.4 and later.

See Also
- setRotation: (page 248)

Declared In
PDFPage.h

selectionForLineAtPoint:

Returns the whole line of text that includes the specified point.

```
- (PDFSelection *)selectionForLineAtPoint:(NSPoint)point
```

Discussion
Returns `NULL` if no line of text contains *`point`*.

Use this method to respond to a triple-click.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `- selectionForRect:` (page 246)
- `- selectionForWordAtPoint:` (page 247)
- `- selectionFromPoint:toPoint:` (page 247)
- `- selectionForRange:` (page 246)

Declared In
`PDFPage.h`

selectionForRange:

Returns the text contained within the specified range.

```
- (PDFSelection *)selectionForRange:(NSRange)range
```

Discussion
This method raises an exception if the range length is `0` or if either end of the range is outside the range of characters on the page.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `- selectionForRect:` (page 246)
- `- selectionForWordAtPoint:` (page 247)
- `- selectionForLineAtPoint:` (page 245)
- `- selectionFromPoint:toPoint:` (page 247)

Declared In
`PDFPage.h`

selectionForRect:

Returns the text enclosed within the specified rectangle, expressed in page (user) coordinates.

```
- (PDFSelection *)selectionForRect:(NSRect)rect
```

Availability
Available in Mac OS X v10.4 and later.

See Also
- `- selectionForWordAtPoint:` (page 247)
- `- selectionForLineAtPoint:` (page 245)
- `- selectionFromPoint:toPoint:` (page 247)

- selectionForRange: (page 246)

Declared In
PDFPage.h

selectionForWordAtPoint:

Returns the whole word that includes the specified point.

- (PDFSelection *)selectionForWordAtPoint:(NSPoint)*point*

Discussion
Returns NULL if no word contains *point*.

Use this method to respond to a double-click.

Availability
Available in Mac OS X v10.4 and later.

See Also
- selectionForRect: (page 246)
- selectionForLineAtPoint: (page 245)
- selectionFromPoint:toPoint: (page 247)
- selectionForRange: (page 246)

Declared In
PDFPage.h

selectionFromPoint:toPoint:

Returns the text between the two specified points in page space.

- (PDFSelection *)selectionFromPoint:(NSPoint)*startPoint* toPoint:(NSPoint)*endPoint*

Discussion
Either point may be the one closer to the start of the page. In determining the selection, the points are sorted first top to bottom and then left to right.

Page space is a 72 dpi coordinate system with the origin at the lower-left corner of the current page.

To visualize the selection, picture the rectangle defined by *startPoint* and *endPoint*. The selection begins at the first character fully within the defined rectangle and closest to its upper-left corner. The selection ends at the last character fully within the defined rectangle and closest to its lower-right corner.

Availability
Available in Mac OS X v10.4 and later.

See Also
- selectionForRect: (page 246)
- selectionForWordAtPoint: (page 247)
- selectionForLineAtPoint: (page 245)
- selectionForRange: (page 246)

Declared In
PDFPage.h

setBounds:forBox:

Sets the bounds for the specified box.

```
- (void)setBounds:(NSRect)bounds forBox:(PDFDisplayBox)box
```

Discussion
If the box does not exist, this method creates it for you.

To remove a box, pass `NSZeroRect` for the bounds (note that you cannot remove the media box). If the box bounds are not in range, this method throws a range exception.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `boundsForBox:` (page 241)

Declared In
PDFPage.h

setDisplaysAnnotations:

Specifies whether or not to display annotations for the page.

```
- (void)setDisplaysAnnotations:(BOOL)display
```

Discussion
If *display* is `YES`, the page will draw annotations when a drawing method is called.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `annotations` (page 240)
- `displaysAnnotations` (page 242)
- `addAnnotation:` (page 239)
- `removeAnnotation:` (page 245)
- `annotationAtPoint:` (page 240)

Declared In
PDFPage.h

setRotation:

Sets the rotation angle for the page in degrees.

```
- (void)setRotation:(int)angle
```

Discussion
The rotation must be a positive or negative multiple of 90 (negative angles are converted to their positive equivalents; for example, -90 is changed to 270); otherwise this method throws an exception.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `rotation` (page 245)

Declared In
`PDFPage.h`

string

Returns an `NSString` object representing the text on the page.

```
- (NSString *)string
```

Availability
Available in Mac OS X v10.4 and later.

See Also
- `numberOfCharacters` (page 244)
- `attributedString` (page 240)

Declared In
`PDFPage.h`

transformContextForBox:

Transforms the current context, given the specified box.

```
- (void)transformContextForBox:(PDFDisplayBox)box
```

Discussion
When transforming the current context, this method takes into account the rotation of the page, as well as the origin of the box with respect to the page's base coordinate system. This is a convenient method to call within the `PDFView` `drawPage:` (page 283) method or from within a draw method of a `PDFAnnotation` subclass.

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

See Also
- `drawWithBox:`

Declared In
`PDFPage.h`

Constants

The following box types may be used with `PDFPage` drawing and bounds-setting methods. See the Adobe PDF Specification for more information on box types, units, and coordinate systems.

Constant	Description
`kPDFDisplayBox-MediaBox`	A rectangle defining the boundaries of the physical medium for display or printing, expressed in default user-space units. Available in Mac OS X v10.4 and later. Declared in `PDFPage.h`.
`kPDFDisplayBoxCropBox`	A rectangle defining the boundaries of the visible region , expressed in default user-space units. Default value equal to `kPDFDisplayBoxMediaBox`. Available in Mac OS X v10.4 and later. Declared in `PDFPage.h`.
`kPDFDisplayBox-BleedBox`	A rectangle defining the boundaries of the clip region for the page contents in a production environment. Default value equal to `kPDFDisplay-BoxCropBox`. Available in Mac OS X v10.4 and later. Declared in `PDFPage.h`.
`kPDFDisplayBoxTrimBox`	A rectangle defining the intended boundaries of the finished page. Default value equal to `kPDFDisplayBoxCropBox`. Available in Mac OS X v10.4 and later. Declared in `PDFPage.h`.
`kPDFDisplayBoxArtBox`	A rectangle defining the boundaries of the page's meaningful content including surrounding white space intended for display. Default value equal to `kPDFDisplayBoxCropBox`. Available in Mac OS X v10.4 and later. Declared in `PDFPage.h`.

PDFSelection Class Reference

Inherits from	NSObject
Conforms to	NSCopying NSObject (NSObject)
Framework	Library/Frameworks/Quartz.framework/Frameworks/PDFKit.framework
Declared in	PDFKit/PDFSelection.h
Availability	Available in Mac OS X v10.4 and later.

Overview

A `PDFSelection` object identifies a contiguous or noncontiguous selection of text in a PDF document.

Tasks

Initializing a Selection

- `initWithDocument:` (page 255)
  Returns an empty `PDFSelection` object.

Getting Information About a Selection

- `pages` (page 256)
  Returns the array of pages contained in the selection.
- `string` (page 257)
  Returns an `NSString` object representing the text contained in the selection (may contain linefeed characters).
- `attributedString` (page 253)
  Returns an `NSAttributedString` object representing the text contained in the selection (may contain linefeed characters).
- `boundsForPage:` (page 253)
  Returns the bounds of the selection on the specified page.
- `selectionsByLine` (page 256)
  Returns an array of selections, one for each line of text covered by the receiver.

- `color` (page 254)
  Returns the color used to draw the selection.

Modifying a Selection

- `addSelection:` (page 252)
  Adds the specified selection to the receiving selection.
- `addSelections:` (page 253)
  Adds the specified array of selections to the receiving selection.
- `extendSelectionAtEnd:` (page 255)
  Extends the selection from its end toward the end of the document.
- `extendSelectionAtStart:` (page 255)
  Extends the selection from its start toward the beginning of the document.

Managing Selection Drawing

- `drawForPage:active:` (page 254)
  Calls `drawForPage:withBox:active:` (page 254) with a default value for box parameter.
- `drawForPage:withBox:active:` (page 254)
  Draws the selection relative to the origin of the specified box in page space.
- `setColor:` (page 256)
  Sets the color used for the drawing of a selection in both active and inactive states.

Instance Methods

addSelection:

Adds the specified selection to the receiving selection.

```
- (void)addSelection:(PDFSelection *)selection
```

Discussion
Selections do not have to be contiguous. If the selection to be added overlaps with the receiving selection, the overlap is removed in a process called normalization.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `extendSelectionAtEnd:` (page 255)
- `extendSelectionAtStart:` (page 255)

Declared In
`PDFSelection.h`

addSelections:

Adds the specified array of selections to the receiving selection.

```
- (void)addSelections:(NSArray *)selections
```

Discussion
This method provides better performance than multiple calls to `addSelection` if you need to add several selections to an existing selection. This is because the normalization of the selection (the removal of any overlaps between selections) occurs only once, after all selections have been added.

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

See Also
`- addSelection:`

Declared In
`PDFSelection.h`

attributedString

Returns an `NSAttributedString` object representing the text contained in the selection (may contain linefeed characters).

```
- (NSAttributedString *)attributedString
```

Availability
Available in Mac OS X v10.4 and later.

See Also
`- string` (page 257)

Declared In
`PDFSelection.h`

boundsForPage:

Returns the bounds of the selection on the specified page.

```
- (NSRect)boundsForPage:(PDFPage *)page
```

Discussion
The selection rectangle is given in page space.

Page space is a 72 dpi coordinate system with the origin at the lower-left corner of the current page.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`PDFSelection.h`

color

Returns the color used to draw the selection.

```
- (NSColor *)color
```

Discussion
Note that when no color has been specified for the `PDFSelection` objects in a document, the selections are drawn using `[NSColor selectedTextBackgroundColor]` for the active state and `[NSColor secondarySelectedControlColor]` for the inactive state.

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

See Also
`- setColor:`

Declared In
`PDFSelection.h`

drawForPage:active:

Calls `drawForPage:withBox:active:` (page 254) with a default value for box parameter.

```
- (void)drawForPage:(PDFPage *)page active:(BOOL)active
```

Discussion
The default value is `kPDFDisplayBoxCropBox`. If active is `YES`, drawing uses `selectedTextBackgroundColor`. If `NO`, it uses `secondarySelectedControlColor`.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- drawForPage:withBox:active:` (page 254)

Declared In
`PDFSelection.h`

drawForPage:withBox:active:

Draws the selection relative to the origin of the specified box in page space.

```
- (void)drawForPage:(PDFPage *)page withBox:(PDFDisplayBox)box active:(BOOL)active
```

Discussion
The selection is drawn using the current highlight color. If active is `YES`, drawing uses `selectedTextBackgroundColor`. If `NO`, it uses `secondarySelectedControlColor`. Refer to the `PDFPage` class for the list of available box types.

Page space is a 72 dpi coordinate system with the origin at the lower-left corner of the current page.

Availability
Available in Mac OS X v10.4 and later.

See Also
- drawForPage:active: (page 254)

Declared In
PDFSelection.h

extendSelectionAtEnd:

Extends the selection from its end toward the end of the document.

```
- (void)extendSelectionAtEnd:(NSInteger)chars
```

Discussion
The selection may be extended by any amount, up to and including the end of the document.

Availability
Available in Mac OS X v10.4 and later.

See Also
- extendSelectionAtStart: (page 255)
- addSelection: (page 252)

Declared In
PDFSelection.h

extendSelectionAtStart:

Extends the selection from its start toward the beginning of the document.

```
- (void)extendSelectionAtStart:(NSInteger)chars
```

Discussion
The selection may be extended by any amount, up to and including the beginning of the document.

Availability
Available in Mac OS X v10.4 and later.

See Also
- extendSelectionAtEnd: (page 255)
- addSelection: (page 252)

Declared In
PDFSelection.h

initWithDocument:

Returns an empty `PDFSelection` object.

```
- (id)initWithDocument:(PDFDocument *)document
```

Discussion
Typically, you don't need to create a `PDFSelection` object, but you can use an empty `PDFSelection` object as a container into which you can place selections, using `addSelection:` and addSelections.

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

Declared In
`PDFSelection.h`

pages

Returns the array of pages contained in the selection.

```
- (NSArray *)pages
```

Discussion
Pages are sorted by index number.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`PDFSelection.h`

selectionsByLine

Returns an array of selections, one for each line of text covered by the receiver.

```
- (NSArray *)selectionsByLine
```

Discussion
If you call this method on a `PDFSelection` object that represents a paragraph, for example, `selectionsByLine` returns an array that contains one `PDFSelection` object for each line of text in the paragraph.

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

Declared In
`PDFSelection.h`

setColor:

Sets the color used for the drawing of a selection in both active and inactive states.

```
- (void)setColor:(NSColor *)color
```

Discussion
When no color has been specified for the `PDFSelection` objects in a document, the selections are drawn using `[NSColor selectedTextBackgroundColor]` for the active state and `[NSColor secondarySelectedControlColor]` for the inactive state. Use the `setColor` method to supply a color you want to be used for the drawing of both active and inactive selections.

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

See Also
`- color`

Declared In
`PDFSelection.h`

string

Returns an `NSString` object representing the text contained in the selection (may contain linefeed characters).

```
- (NSString *)string
```

Availability
Available in Mac OS X v10.4 and later.

See Also
`- attributedString`

Declared In
`PDFSelection.h`

PDFThumbnailView Class Reference

Inherits from	NSView : NSResponder : NSObject
Conforms to	NSCoding NSAnimatablePropertyContainer (NSView) NSCoding (NSResponder) NSObject (NSObject)
Framework	Library/Frameworks/Quartz.framework/Frameworks/PDFKit.framework
Declared in	PDFKit/PDFThumbnailView.h
Availability	Available in Mac OS X v10.5 and later.

Overview

A `PDFThumbnailView` object contains a set of thumbnails, each of which represents a page in a PDF document.

Tasks

Accessing the Associated PDF View

- `PDFView` (page 262)
  Returns the `PDFView` object associated with the thumbnail view.
- `setPDFView:` (page 265)
  Associates the specified `PDFView` object with the thumbnail view.

Managing the Size of a Thumbnail View

- `thumbnailSize` (page 266)
  Returns the maximum width and height of the thumbnails in the thumbnail view.
- `setThumbnailSize:` (page 265)
  Sets the maximum width and height of the thumbnails in the thumbnail view.

Working with Thumbnail View Display Characteristics

- `maximumNumberOfColumns` (page 262)
  Returns the maximum number of columns of thumbnails the thumbnail view can display.
- `setMaximumNumberOfColumns:` (page 265)
  Sets the maximum number of columns of thumbnails the thumbnail view can display.
- `labelFont` (page 262)
  Returns the font used to label the thumbnails.
- `setLabelFont:` (page 264)
  Sets the font used to label the thumbnails.
- `backgroundColor` (page 261)
  Returns the color used in the background of the thumbnail view.
- `setBackgroundColor:` (page 264)
  Sets the color used in the background of the thumbnail view.

Managing the Behavior of a Thumbnail View

- `allowsDragging` (page 260)
  Returns a Boolean value indicating whether users can drag thumbnails (that is, re-order pages in the document) within the thumbnail view.
- `setAllowsDragging:` (page 263)
  Sets whether users can drag thumbnails within the thumbnail view; that is, re-order pages in the document.
- `allowsMultipleSelection` (page 261)
  Returns a Boolean value indicating whether users can select multiple thumbnails in the thumbnail view at one time.
- `setAllowsMultipleSelection:` (page 263)
  Sets whether the thumbnail view allows users to select more than one thumbnail at a time.
- `selectedPages` (page 263)
  Returns an array of PDF pages that correspond to the selected thumbnails in the thumbnail view.

Instance Methods

allowsDragging

Returns a Boolean value indicating whether users can drag thumbnails (that is, re-order pages in the document) within the thumbnail view.

```
- (BOOL)allowsDragging
```

Return Value
`YES` if users can re-order pages by dragging thumbnails, `NO` otherwise.

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

See Also
- `setAllowsDragging:` (page 263)

Declared In
`PDFThumbnailView.h`

allowsMultipleSelection

Returns a Boolean value indicating whether users can select multiple thumbnails in the thumbnail view at one time.

```
- (BOOL)allowsMultipleSelection
```

Return Value
`YES` if users can select multiple thumbnails simultaneously, `NO` otherwise.

Discussion
By default, `PDFThumbnailView` allows only a single thumbnail to be selected at one time. When this is the case, you can get the PDF page that corresponds to the selected thumbnail using the `PDFView` method `currentPage` (page 280).

When multiple selections are enabled, however, you must use `selectedPages` (page 263) to get the pages that correspond to the set of selected thumbnails.

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

See Also
- `setAllowsMultipleSelection:` (page 263)
- `selectedPages` (page 263)

Declared In
`PDFThumbnailView.h`

backgroundColor

Returns the color used in the background of the thumbnail view.

```
- (NSColor *)backgroundColor
```

Return Value
The color of the background in the thumbnail view.

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

See Also
- `setBackgroundColor:` (page 264)

Declared In
`PDFThumbnailView.h`

labelFont

Returns the font used to label the thumbnails.

```
- (NSFont *)labelFont
```

Return Value
The font used in the thumbnail labels.

Discussion
Typically, the label of a thumbnail is the page number of the page it represents.

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

See Also
- `setLabelFont:` (page 264)

Declared In
`PDFThumbnailView.h`

maximumNumberOfColumns

Returns the maximum number of columns of thumbnails the thumbnail view can display.

```
- (NSUInteger)maximumNumberOfColumns
```

Return Value
The maximum number of columns of thumbnails the thumbnail view can display. If `0`, the thumbnail displays as many columns of thumbnails as fit in its size.

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

See Also
- `setThumbnailSize:` (page 265)

Declared In
`PDFThumbnailView.h`

PDFView

Returns the `PDFView` object associated with the thumbnail view.

```
- (PDFView *)PDFView
```

Return Value
The PDF view object associated with the thumbnail view.

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

See Also
- `setPDFView:` (page 265)

Declared In
PDFThumbnailView.h

selectedPages

Returns an array of PDF pages that correspond to the selected thumbnails in the thumbnail view.

```
- (NSArray *)selectedPages
```

Return Value
An array of PDF pages that correspond to the thumbnails selected in the thumbnail view.

Discussion
If the thumbnail view allows multiple selections (if `allowsMultipleSelection` (page 261) returns `YES`), you can use this method to get the PDF pages that correspond to the selected thumbnails.

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

See Also
- `allowsMultipleSelection` (page 261)
- `setAllowsDragging:` (page 263)

Declared In
PDFThumbnailView.h

setAllowsDragging:

Sets whether users can drag thumbnails within the thumbnail view; that is, re-order pages in the document.

```
- (void)setAllowsDragging:(BOOL)allow
```

Parameters
allow
Pass `YES` to allow users to drag thumbnails in the thumbnail view (this allows them to re-order pages in the document), or `NO` to disallow.

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

See Also
- `allowsDragging` (page 260)

Declared In
PDFThumbnailView.h

setAllowsMultipleSelection:

Sets whether the thumbnail view allows users to select more than one thumbnail at a time.

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

Parameters

flag

Pass YES to allows users to select multiple thumbnails at one time, or NO to disallow.

Discussion

By default, PDFThumbnailView allows only a single thumbnail to be selected at one time. When this is the case, you can get the PDF page that corresponds to the selected thumbnail using the PDFView method currentPage (page 280).

If you use setAllowsMultipleSelection to enable multiple selections, however, you must use selectedPages (page 263) to get the pages that correspond to the set of selected thumbnails.

Availability

Available in Mac OS X v10.5 and later.

See Also

- allowsMultipleSelection (page 261)
- selectedPages (page 263)

Declared In

PDFThumbnailView.h

setBackgroundColor:

Sets the color used in the background of the thumbnail view.

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

Parameters

color

The color to be used in the background of the thumbnail view.

Availability

Available in Mac OS X v10.5 and later.

See Also

- backgroundColor (page 261)

Declared In

PDFThumbnailView.h

setLabelFont:

Sets the font used to label the thumbnails.

```
- (void)setLabelFont:(NSFont *)font
```

Parameters

font

The font to be used in the thumbnail labels.

Discussion

Typically, the label of a thumbnail is the page number of the page it represents.

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

See Also
- `labelFont` (page 262)

Declared In
`PDFThumbnailView.h`

setMaximumNumberOfColumns:

Sets the maximum number of columns of thumbnails the thumbnail view can display.

```
- (void)setMaximumNumberOfColumns:(NSUInteger)maxColumns
```

Parameters

maxColumns

The maximum number of columns of thumbnails the thumbnail view can display. Pass `0` to make the thumbnail view display as many columns as fit in its size.

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

See Also
- `thumbnailSize` (page 266)

Declared In
`PDFThumbnailView.h`

setPDFView:

Associates the specified `PDFView` object with the thumbnail view.

```
- (void)setPDFView:(PDFView *)view
```

Parameters

view

The PDF view object to associate with the thumbnail view.

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

See Also
- `PDFView` (page 262)

Declared In
`PDFThumbnailView.h`

setThumbnailSize:

Sets the maximum width and height of the thumbnails in the thumbnail view.

```
- (void)setThumbnailSize:(NSSize)size
```

Parameters

size

The maximum width and height the thumbnails in the thumbnail view should be.

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

See Also
- thumbnailSize (page 266)

Declared In
PDFThumbnailView.h

thumbnailSize

Returns the maximum width and height of the thumbnails in the thumbnail view.

```
- (NSSize)thumbnailSize
```

Return Value
The maximum width and height of the thumbnails in the thumbnail view.

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

See Also
- setThumbnailSize: (page 265)

Declared In
PDFThumbnailView.h

PDFView Class Reference

Inherits from	NSView : NSResponder : NSObject
Conforms to	NSAnimatablePropertyContainer (NSView) NSCoding (NSResponder) NSObject (NSObject)
Framework	Library/Frameworks/Quartz.framework/Frameworks/PDFKit.framework
Declared in	PDFKit/PDFView.h
Availability	Available in Mac OS X v10.4 and later.

Overview

A `PDFView` object encapsulates the functionality of PDF Kit into a single widget that you can add to your application using Interface Builder.

`PDFView` may be the only class you need to deal with for adding PDF functionality to your application. It lets you display PDF data and allows users to select content, navigate through a document, set zoom level, and copy textual content to the Pasteboard. `PDFView` also keeps track of page history.

You can subclass `PDFView` to create a custom PDF viewer.

You can also create a custom PDF viewer by using the PDF Kit utility classes directly and not using `PDFView` at all.

Tasks

Associating a Document with a View

`- document` (page 282)
Returns the document associated with a `PDFView` object.

`- setDocument:` (page 295)
Associates a document with a `PDFView` object.

Navigating Within a Document

- `canGoBack` (page 274)
  Returns a Boolean value indicating whether the user can navigate to the previous page in the page history.
- `canGoForward` (page 275)
  Returns a Boolean value indicating whether the user can navigate to the next page in the page history.
- `canGoToFirstPage` (page 275)
  Returns a Boolean value indicating whether the user can navigate to the first page of the document.
- `canGoToLastPage` (page 275)
  Returns a Boolean value indicating whether the user can navigate to the last page of the document.
- `canGoToNextPage` (page 276)
  Returns a Boolean value indicating whether the user can navigate to the next page of the document.
- `canGoToPreviousPage` (page 276)
  Returns a Boolean value indicating whether the user can navigate to the previous page of the document.
- `currentPage` (page 280)
  Returns the current page.
- `currentDestination` (page 279)
  Returns a `PDFDestination` object representing the current page and the current point in the view specified in page space.
- `goBack:` (page 284)
  Navigates back one step in the page history.
- `goForward:` (page 284)
  Navigates forward one step in the page history.
- `goToFirstPage:` (page 285)
  Navigates to the first page of the document.
- `goToLastPage:` (page 285)
  Navigates to the last page of the document.
- `goToNextPage:` (page 285)
  Navigates to the next page of the document.
- `goToPreviousPage:` (page 286)
  Navigates to the previous page of the document.
- `goToPage:` (page 286)
  Scrolls to the specified page.
- `goToDestination:` (page 284)
  Navigates to the specified destination.
- `goToSelection:` (page 287)
  Scrolls to the first character of the specified selection.
- `goToRect:onPage:` (page 286)
  Navigates to the specified rectangle on the specified page.

Working with Display Modes and Characteristics

- `setDisplayMode:` (page 294)
  Sets the display mode for the view.
- `displayMode` (page 281)
  Returns the current display mode.
- `setDisplaysPageBreaks:` (page 294)
  Toggles the display of page breaks.
- `displaysPageBreaks` (page 282)
  Returns a Boolean value indicating whether the view is displaying page breaks.
- `setDisplayBox:` (page 293)
  Specifies the box to display and to clip to.
- `displayBox` (page 281)
  Returns the current style of display box.
- `displaysAsBook` (page 282)
  Returns a Boolean value indicating whether the view will display the first page as a book cover (meaningful only when the document is in two-up or two-up continuous display mode).
- `setDisplaysAsBook:` (page 294)
  Specifies whether the view should treat the document's first page as a book cover.
- `setShouldAntiAlias:` (page 296)
  Specifies whether to use anti-aliasing in the view.
- `shouldAntiAlias` (page 296)
  Returns a Boolean value indicating whether the view is anti-aliased.
- `setGreekingThreshold:` (page 295)
  Sets the greeking threshold to use for displaying text.
- `greekingThreshold` (page 287)
  Returns the current greeking threshold for the view.
- `takeBackgroundColorFrom:` (page 297)
  Sets the view's background color to the specified color.
- `setBackgroundColor:` (page 291)
  Sets the view's background color.
- `backgroundColor` (page 274)
  Returns the view's background color.

Setting the Delegate

- `setDelegate:` (page 293)
  Sets a delegate for the view.
- `delegate` (page 281)
  Returns the view's delegate.

Scaling the View

- `setScaleFactor:` (page 296)
  Sets the scale factor for the view.
- `scaleFactor` (page 290)
  Returns the current scale factor for the view.
- `zoomIn:` (page 298)
  Zooms in by increasing the scaling factor.
- `canZoomIn` (page 276)
  Returns a Boolean value indicating whether the user can magnify the view—that is, zoom in.
- `zoomOut:` (page 298)
  Zooms out by decreasing the scaling factor.
- `canZoomOut` (page 277)
  Returns a Boolean value indicating whether the user can view an expanded area—that is, zoom out.
- `setAutoScales:` (page 291)
  Toggles whether the scaling factor applied to a view automatically responds to resizing.
- `autoScales` (page 274)
  Returns a Boolean value indicating whether autoscaling is set.

Working with Mouse Position and Events

- `areaOfInterestForMouse:` (page 273)
  Returns the type of area the mouse cursor is over.
- `setCursorForAreaOfInterest:` (page 293)
  Sets the type of mouse cursor according to the type of area the mouse cursor is over.
- `performAction:` (page 288)
  Performs the specified action.

Handling Selections

- `currentSelection` (page 280)
  Returns the current selection.
- `setCurrentSelection:` (page 292)
  Sets the selection.
- `selectAll:` (page 290)
  Selects all text in the document.
- `clearSelection` (page 277)
  Clears the selection.
- `copy:` (page 279)
  Copies the text in the selection, if any, to the Pasteboard.
- `scrollSelectionToVisible:` (page 290)
  Scrolls the view until the selection is visible.

- `setCurrentSelection:animate:` (page 292)
  Sets the selection, in an animated way, if desired.
- `setHighlightedSelections:` (page 295)
  Highlights the specified array of selections.
- `highlightedSelections` (page 287)
  Returns the array of selections that are highlighted using `setHighlightedSelections`.

Setting the Password

- `takePasswordFrom:` (page 297)
  A convenience method that calls `- [[self document] setpassword:]` with the password from the specified sender.

Rendering the View and Printing

- `drawPage:` (page 283)
  For use by subclasses of `PDFView` for custom rendering of pages.
- `drawPagePost:` (page 283)
  For use by subclasses of `PDFView` for post-page rendering.
- `printWithInfo:autoRotate:` (page 289)
  Prints the document with the specified printer information.
- `printWithInfo:autoRotate:pageScaling:` (page 289)
  Prints the document with the specified printer and page-scaling information.

Conversion Methods for Subclasses

- `pageForPoint:nearest:` (page 288)
  Returns the page containing a point specified in view coordinates.
- `convertPoint:toPage:` (page 278)
  Converts a point from view space to page space.
- `convertRect:toPage:` (page 279)
  Converts a rectangle from view space to page space.
- `convertPoint:fromPage:` (page 278)
  Converts a point from page space to view space.
- `convertRect:fromPage:` (page 278)
  Converts a rectangle from page space to view space.

Miscellaneous Methods

- `documentView` (page 283)
  Returns the innermost view used by `PDFView` or by your `PDFView` subclass.
- `rowSizeForPage:` (page 289)
  Returns the size needed to display a row of the current document page.

- `layoutDocumentView` (page 288)
  Performs layout of the inner views.
- `allowsDragging` (page 273)
  Determines whether the view can accept new PDF documents dragged into it by the user.
- `setAllowsDragging:` (page 291)
  Specifies whether the view can accept drags.
- `visiblePages` (page 297)
  Returns an array of `PDFPage` objects that represent the currently visible pages.
- `annotationsChangedOnPage:` (page 273)
  Tells the PDF view that an annotation on the specified page has changed.

Managing Scale Factor

- `PDFViewWillChangeScaleFactor:toScale:` (page 300) *delegate method*
  Delegate method for overriding changes to scale factor.

Handling URL Links

- `PDFViewWillClickOnLink:withURL:` (page 301) *delegate method*
  Delegate method for handling clicks on URL links in a view.

Responding to Annotation Actions

- `PDFViewPerformFind:` (page 299) *delegate method*
  Delegate method that performs a find operation.
- `PDFViewPerformGoToPage:` (page 299) *delegate method*
  Delegate method that performs a go-to operation.
- `PDFViewPerformPrint:` (page 300) *delegate method*
  Delegate method that prints the current document.
- `PDFViewOpenPDF:forRemoteGoToAction:` (page 298) *delegate method*
  Delegate method that opens a specified page.

Changing the Print Job Title

- `PDFViewPrintJobTitle:` (page 300) *delegate method*
  Delegate method that overrides the job title used when the `PDFView` is printed.

Instance Methods

allowsDragging

Determines whether the view can accept new PDF documents dragged into it by the user.

```
- (BOOL)allowsDragging
```

Availability
Available in Mac OS X v10.4 and later.

See Also
`- setAllowsDragging:` (page 291)

Declared In
`PDFView.h`

annotationsChangedOnPage:

Tells the PDF view that an annotation on the specified page has changed.

```
- (void)annotationsChangedOnPage:(PDFPage *)page
```

Discussion
When the `PDFView` object receives this message, it rescans for tool tips and pop-ups and informs the `PDFThumbailView` objects so the thumbnail images can be redrawn.

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

Declared In
`PDFView.h`

areaOfInterestForMouse:

Returns the type of area the mouse cursor is over.

```
- (PDFAreaOfInterest)areaOfInterestForMouse:(NSEvent *)theEvent
```

Discussion
The `PDFAreaOfInterest` enumeration defines the various area types. This method is for custom subclasses of the `PDFView` class. Use it if you override the `NSResponder` class's `mouseMoved:` method or related methods.

Refer to "Constants" (page 301) for the various values of the area-of-interest constants. Each of these constants contributes to the value of the `PDFAreaOfInterest` bit field.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `setCursorForAreaOfInterest:` (page 293)

Declared In
`PDFView.h`

autoScales

Returns a Boolean value indicating whether autoscaling is set.

```
- (BOOL)autoScales
```

Availability
Available in Mac OS X v10.4 and later.

See Also
- `setAutoScales:` (page 291)

Declared In
`PDFView.h`

backgroundColor

Returns the view's background color.

```
- (NSColor *)backgroundColor
```

Discussion
A view's background is the area displayed to either side of a PDF document's pages. The background also appears between pages when page breaks are enabled. The default color is a 50% gray.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `takeBackgroundColorFrom:` (page 297)
- `setBackgroundColor:` (page 291)

Declared In
`PDFView.h`

canGoBack

Returns a Boolean value indicating whether the user can navigate to the previous page in the page history.

```
- (BOOL)canGoBack
```

Discussion
The page history gets built as your application calls navigation methods such as `goToDestination:` (page 284) and `goToLastPage:` (page 285).

Availability
Available in Mac OS X v10.4 and later.

See Also
- `goBack:` (page 284)

Declared In
`PDFView.h`

canGoForward

Returns a Boolean value indicating whether the user can navigate to the next page in the page history.

```
- (BOOL)canGoForward
```

Discussion
The page history gets built as your application calls navigation methods such as `goToDestination:` (page 284) and `goToLastPage:` (page 285).

Availability
Available in Mac OS X v10.4 and later.

See Also
- `goForward:` (page 284)

Declared In
`PDFView.h`

canGoToFirstPage

Returns a Boolean value indicating whether the user can navigate to the first page of the document.

```
- (BOOL)canGoToFirstPage
```

Discussion
The return value will be `YES` unless the view is already displaying the first page.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `goToFirstPage:` (page 285)

Declared In
`PDFView.h`

canGoToLastPage

Returns a Boolean value indicating whether the user can navigate to the last page of the document.

```
- (BOOL)canGoToLastPage
```

Discussion
The return value will be `YES` unless the view is already displaying the last page.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- goToLastPage:` (page 285)

Declared In
`PDFView.h`

canGoToNextPage

Returns a Boolean value indicating whether the user can navigate to the next page of the document.

```
- (BOOL)canGoToNextPage
```

Discussion
The return value will be `YES` unless the view is displaying the last page.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- goToNextPage:` (page 285)

Declared In
`PDFView.h`

canGoToPreviousPage

Returns a Boolean value indicating whether the user can navigate to the previous page of the document.

```
- (BOOL)canGoToPreviousPage
```

Discussion
The return value will be `YES` unless the view is displaying the first page.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- canGoToPreviousPage` (page 276)

Declared In
`PDFView.h`

canZoomIn

Returns a Boolean value indicating whether the user can magnify the view—that is, zoom in.

```
- (BOOL)canZoomIn
```

Availability
Available in Mac OS X v10.4 and later.

See Also
- `zoomIn:` (page 298)
- `zoomOut:` (page 298)
- `canZoomOut` (page 277)

Declared In
`PDFView.h`

canZoomOut

Returns a Boolean value indicating whether the user can view an expanded area—that is, zoom out.

```
- (BOOL)canZoomOut
```

Availability
Available in Mac OS X v10.4 and later.

See Also
- `zoomIn:` (page 298)
- `canZoomIn` (page 276)
- `zoomOut:` (page 298)

Declared In
`PDFView.h`

clearSelection

Clears the selection.

```
- (void)clearSelection
```

Discussion
The view redraws as necessary but does not scroll. This call is equivalent to calling `[PDFView setCurrentSelection:NULL]`.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `currentSelection` (page 280),
- `setCurrentSelection:` (page 292)

Declared In
`PDFView.h`

convertPoint:fromPage:

Converts a point from page space to view space.

```
- (NSPoint)convertPoint:(NSPoint)point fromPage:(PDFPage *)page
```

Discussion
Page space is a 72 dpi coordinate system with the origin at the lower-left corner of the current page. View space is a coordinate system with the origin at the lower-left corner of the current PDF view.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `convertPoint:toPage:` (page 278)
- `convertRect:toPage:` (page 279)
- `convertRect:fromPage:` (page 278)
- `pageForPoint:nearest:` (page 288)

Declared In
`PDFView.h`

convertPoint:toPage:

Converts a point from view space to page space.

```
- (NSPoint)convertPoint:(NSPoint)point toPage:(PDFPage *)page
```

Discussion
Page space is a 72 dpi coordinate system with the origin at the lower-left corner of the current page. View space is a coordinate system with the origin at the lower-left corner of the current PDF view.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `convertRect:toPage:` (page 279)
- `convertPoint:fromPage:` (page 278)
- `convertRect:fromPage:` (page 278)

Declared In
`PDFView.h`

convertRect:fromPage:

Converts a rectangle from page space to view space.

```
- (NSRect)convertRect:(NSRect)rect fromPage:(PDFPage *)page
```

Discussion
Page space is a 72 dpi coordinate system with the origin at the lower-left corner of the current page. View space is a coordinate system with the origin at the lower-left corner of the current PDF view.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `convertPoint:toPage:` (page 278)
- `convertRect:toPage:` (page 279)
- `convertPoint:fromPage:` (page 278)

Declared In
`PDFView.h`

convertRect:toPage:

Converts a rectangle from view space to page space.

```
- (NSRect)convertRect:(NSRect)rect toPage:(PDFPage *)page
```

Discussion
Page space is a 72 dpi coordinate system with the origin at the lower-left corner of the current page. View space is a coordinate system with the origin at the lower-left corner of the current PDF view.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `convertPoint:toPage:` (page 278)
- `convertPoint:fromPage:` (page 278)
- `convertRect:fromPage:` (page 278)

Declared In
`PDFView.h`

copy:

Copies the text in the selection, if any, to the Pasteboard.

```
- (void)copy:(id)sender
```

Availability
Available in Mac OS X v10.4 and later.

See Also
- `currentSelection` (page 280)

Declared In
`PDFView.h`

currentDestination

Returns a `PDFDestination` object representing the current page and the current point in the view specified in page space.

```
- (PDFDestination *)currentDestination
```

Discussion
Page space is a 72 dpi coordinate system with the origin at the lower-left corner of the current page.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `currentPage` (page 280)
- `goToDestination:` (page 284) (PDFDestination)

Declared In
`PDFView.h`

currentPage

Returns the current page.

```
- (PDFPage *)currentPage
```

Discussion
When there are two pages in the view in a two-up mode, "current page" is the left page. For continuous modes, returns the page crossing a horizontal line halfway between the view's top and bottom bounds.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `currentDestination` (page 279)
- `goToDestination:` (page 284)

Declared In
`PDFView.h`

currentSelection

Returns the current selection.

```
- (PDFSelection *)currentSelection
```

Discussion
Returns `NULL` if no selection exists.

Note that this method returns the actual instance of the current `PDFSelection` object. Therefore, if you want to modify it, you should make a copy of the returned selection and modify that, instead.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `setCurrentSelection:` (page 292)
- `clearSelection` (page 277)

Declared In
PDFView.h

delegate

Returns the view's delegate.

```
- (id)delegate
```

Availability
Available in Mac OS X v10.4 and later.

See Also
- setDelegate: (page 293)

Declared In
PDFView.h

displayBox

Returns the current style of display box.

```
- (PDFDisplayBox)displayBox
```

Discussion
The available values for display boxes are defined in the Constants section in the PDFPage class.

Availability
Available in Mac OS X v10.4 and later.

See Also
- setDisplayBox: (page 293)

Declared In
PDFView.h

displayMode

Returns the current display mode.

```
- (PDFDisplayMode)displayMode
```

Discussion
See "Constants" (page 301) for possible values.

Availability
Available in Mac OS X v10.4 and later.

See Also
- setDisplayMode: (page 294)

Declared In
PDFView.h

displaysAsBook

Returns a Boolean value indicating whether the view will display the first page as a book cover (meaningful only when the document is in two-up or two-up continuous display mode).

```
- (BOOL)displaysAsBook
```

Availability
Available in Mac OS X v10.4 and later.

See Also
- setDdisplaysAsBook: (page 294)

Declared In
PDFView.h

displaysPageBreaks

Returns a Boolean value indicating whether the view is displaying page breaks.

```
- (BOOL)displaysPageBreaks
```

Availability
Available in Mac OS X v10.4 and later.

See Also
- setDisplaysPageBreaks: (page 294)

Declared In
PDFView.h

document

Returns the document associated with a PDFView object.

```
- (PDFDocument *)document
```

Availability
Available in Mac OS X v10.4 and later.

See Also
- setDocument: (page 295)

Declared In
PDFView.h

documentView

Returns the innermost view used by `PDFView` or by your `PDFView` subclass.

```
- (NSView *)documentView
```

Discussion
The innermost view is the one displaying the visible document pages. This method is useful when converting coordinates from one view to another.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- layoutDocumentView` (page 288)

Declared In
`PDFView.h`

drawPage:

For use by subclasses of `PDFView` for custom rendering of pages.

```
- (void)drawPage:(PDFPage *)page
```

Discussion
Do not invoke this method, except by invoking it on `super` from a subclass.

The `PDFView` class calls `drawPage:` (page 283) as necessary for each visible page that requires rendering. In the `PDFView` class, this method erases *page* to white, calls `[page drawInRect: pageRect withBox: [self displayBox]]`, and then draws the selection, if any.

You can override this method to draw on top of a PDF page or to control how pages are drawn. In these cases, invoke this method on `super` and then perform custom drawing on top of the PDF page.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`PDFView.h`

drawPagePost:

For use by subclasses of `PDFView` for post-page rendering.

```
- (void)drawPagePost:(PDFPage *)page
```

Discussion
The default implementation of this method draws the text highlighting (if any) for the page. This method does not apply scaling or rotating to the current context to map to page space; instead, the context is in view-space coordinates (in which the origin is at the lower-left corner of the current PDF view).

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

Declared In
PDFView.h

goBack:

Navigates back one step in the page history.

```
- (IBAction)goBack:(id)sender
```

Discussion
The page history gets built as your application calls navigation methods such as `goToDestination:` (page 284) and `goToLastPage:` (page 285).

Availability
Available in Mac OS X v10.4 and later.

See Also
- `canGoBack` (page 274)

Declared In
PDFView.h

goForward:

Navigates forward one step in the page history.

```
- (IBAction)goForward:(id)sender
```

Discussion
The page history gets built as your application calls navigation methods such as `goToDestination:` (page 284) and `goToLastPage:` (page 285).

Availability
Available in Mac OS X v10.4 and later.

See Also
- `canGoForward` (page 275)

Declared In
PDFView.h

goToDestination:

Navigates to the specified destination.

```
- (void)goToDestination:(PDFDestination *)destination
```

Discussion
Destinations include a page and a point on the page specified in page space.

Page space is a 72 dpi coordinate system with the origin at the lower-left corner of the current page.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `currentDestination` (page 279) (PDFDestination)
- `currentPage` (page 280)

Declared In
`PDFView.h`

goToFirstPage:

Navigates to the first page of the document.

```
- (IBAction)goToFirstPage:(id)sender
```

Discussion
PDF Kit records the move in its page history.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `canGoToFirstPage` (page 275)

Declared In
`PDFView.h`

goToLastPage:

Navigates to the last page of the document.

```
- (IBAction)goToLastPage:(id)sender
```

Discussion
PDF Kit records the move in its page history.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `canGoToLastPage` (page 275)

Declared In
`PDFView.h`

goToNextPage:

Navigates to the next page of the document.

```
- (IBAction)goToNextPage:(id)sender
```

Discussion
PDF Kit records the move in its page history.

Availability
Available in Mac OS X v10.4 and later.

See Also
- canGoToNextPage (page 276)

Declared In
PDFView.h

goToPage:

Scrolls to the specified page.

```
- (void)goToPage:(PDFPage *)page
```

Discussion
PDF Kit records the move in its page history.

Availability
Available in Mac OS X v10.4 and later.

Declared In
PDFView.h

goToPreviousPage:

Navigates to the previous page of the document.

```
- (IBAction)goToPreviousPage:(id)sender
```

Discussion
PDF Kit records the move in its page history.

Availability
Available in Mac OS X v10.4 and later.

See Also
- canGoToPreviousPage (page 276)

Declared In
PDFView.h

goToRect:onPage:

Navigates to the specified rectangle on the specified page.

```
- (void)goToRect:(NSRect)rect onPage:(PDFPage *)page
```

Discussion
If the specified rectangle is already visible, this method does nothing. This allows you to scroll the `PDFView` object to a specific `PDFAnnotation` or `PDFSelection` object, because both of these objects have bounds methods that return an annotation or selection position in page space.

Note that *`rect`* is specified in page-space coordinates. Page space is a 72 dpi coordinate system with the origin at the lower-left corner of the current page.

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

Declared In
`PDFView.h`

goToSelection:

Scrolls to the first character of the specified selection.

```
- (void)goToSelection:(PDFSelection *)selection
```

Discussion
PDF Kit records the move in its page history.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`PDFView.h`

greekingThreshold

Returns the current greeking threshold for the view.

```
- (float)greekingThreshold
```

Availability
Available in Mac OS X v10.4 and later.

See Also
`- setGreekingThreshold:` (page 295)

Declared In
`PDFView.h`

highlightedSelections

Returns the array of selections that are highlighted using `setHighlightedSelections`.

```
- (NSArray *)highlightedSelections
```

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

See Also
- `setHighlightedSelections:` (page 295)

Declared In
`PDFView.h`

layoutDocumentView

Performs layout of the inner views.

```
- (void)layoutDocumentView
```

Discussion
The `PDFView` actually contains several subviews, such as the document view (where the PDF is actually drawn) and a "matte view" (which may appear as a gray area around the PDF content, depending on the scaling). Changes to the PDF content may require changes to these inner views, so you must call this method explicitly if you use PDF Kit utility classes to add or remove a page, rotate a page, or perform other operations affecting visible layout.

This method is called automatically from `PDFView` methods that affect the visible layout (such as `setDocument:` (page 295), `setDisplayBox:` (page 293) or `zoomIn:` (page 298)).

Availability
Available in Mac OS X v10.4 and later.

See Also
- `documentView` (page 283)

Declared In
`PDFView.h`

pageForPoint:nearest:

Returns the page containing a point specified in view coordinates.

```
- (PDFPage *)pageForPoint:(NSPoint)point nearest:(BOOL)nearest
```

Discussion
Returns `NULL` if there's no page at the specified point and *`nearest`* is set to `NO`.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`PDFView.h`

performAction:

Performs the specified action.

```
- (void)performAction:(PDFAction *)action
```

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

Declared In
PDFView.h

printWithInfo:autoRotate:

Prints the document with the specified printer information.

```
- (void)printWithInfo:(NSPrintInfo *)printInfo autoRotate:(BOOL)doRotate
```

Discussion
If `autoRotate` is set to `YES`, then ths method ignores the orientation attribute in the `NSPrintInfo` object and instead chooses the orientation that best fits the page to the paper size. This orientation occurs on a page-by-page basis.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- printWithInfo:autoRotate:pageScaling:` (page 289)

Declared In
PDFView.h

printWithInfo:autoRotate:pageScaling:

Prints the document with the specified printer and page-scaling information.

```
- (void)printWithInfo:(NSPrintInfo *)printInfo autoRotate:(BOOL)doRotate
    pageScaling:(PDFPrintScalingMode)scale
```

Discussion
If *`pageScaling`* is set to `kPDFPrintPageScaleToFit`, each page is scaled up or down to best fit the paper size. If *`pageScaling`* is set to `kPDFPrintPageScaleDownToFit`, only large pages are scaled down to fit; small pages are not scaled up to fit. Specifying `kPDFPrintPageScaleNone` for *`pageScaling`* is equivalent to calling `printWithInfo:autoRotate:` (page 289). See PDFDocument for more information on page-scaling types.

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

See Also
`- printWithInfo:autoRotate:` (page 289)

Declared In
PDFView.h

rowSizeForPage:

Returns the size needed to display a row of the current document page.

```
- (NSSize)rowSizeForPage:(PDFPage *)page
```

Discussion
The size is dependent on the current scale factor and display attributes.

Availability
Available in Mac OS X v10.4 and later.

Declared In
PDFView.h

scaleFactor

Returns the current scale factor for the view.

```
- (float)scaleFactor
```

Availability
Available in Mac OS X v10.4 and later.

See Also
- setScaleFactor: (page 296)

Declared In
PDFView.h

scrollSelectionToVisible:

Scrolls the view until the selection is visible.

```
- (void)scrollSelectionToVisible:(id)sender
```

Availability
Available in Mac OS X v10.4 and later.

Declared In
PDFView.h

selectAll:

Selects all text in the document.

```
- (IBAction)selectAll:(id)sender
```

Availability
Available in Mac OS X v10.4 and later.

Declared In
PDFView.h

setAllowsDragging:

Specifies whether the view can accept drags.

```
- (void)setAllowsDragging:(BOOL)allow
```

Discussion
If set to `YES`, the user can drag a new PDF document into the view. The new document is then displayed in the view, and the old document is released.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `allowsDragging` (page 273)

Declared In
`PDFView.h`

setAutoScales:

Toggles whether the scaling factor applied to a view automatically responds to resizing.

```
- (void)setAutoScales:(BOOL)newAuto
```

Discussion
When set to autoscaling, the document scales to fill the `PDFView` object as the user resizes it.

For the single-page and two-up continuous modes, autoscaling fits the page to the width of the view. For single-page and two-up noncontinuous modes, autoscaling provides best fit, in which the viewed pages are as large as possible while displaying in their entirety within the view.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `autoScales` (page 274)

Declared In
`PDFView.h`

setBackgroundColor:

Sets the view's background color.

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

Discussion
A view's background is the area displayed to either side of a PDF document's pages. The background also appears between pages when page breaks are enabled. The default color is a 50% gray.

Availability
Available in Mac OS X v10.4 and later.

See Also
- takeBackgroundColorFrom: (page 297)
- backgroundColor (page 274)

Declared In
PDFView.h

setCurrentSelection:

Sets the selection.

```
- (void)setCurrentSelection:(PDFSelection *)selection
```

Discussion
The view redraws as necessary but does not scroll. If you need to scroll to the current selection, use scrollSelectionToVisible: (page 290). If you pass nil for the selection, this call is equivalent to calling clearSelection (page 277).

Availability
Available in Mac OS X v10.4 and later.

See Also
- setCurrentSelection:animate: (page 292)
- currentSelection (page 280)
- clearSelection (page 277)

Declared In
PDFView.h

setCurrentSelection:animate:

Sets the selection, in an animated way, if desired.

```
- (void)setCurrentSelection:(PDFSelection *)selection animate:(BOOL)animate
```

Discussion
This method behaves as setCurrentSelection: (page 292), but with the addition of animation, if *animate* is YES. The animation serves to draw the user's attention to the new selection, which can be useful when implementing search.

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

See Also
- setCurrentSelection: (page 292)
- clearSelection (page 277)

Declared In
PDFView.h

setCursorForAreaOfInterest:

Sets the type of mouse cursor according to the type of area the mouse cursor is over.

```
- (void)setCursorForAreaOfInterest:(PDFAreaOfInterest)area
```

Discussion
This method is especially useful for custom subclasses of the `PDFView` class.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- areaOfInterestForMouse:` (page 273)

Declared In
`PDFView.h`

setDelegate:

Sets a delegate for the view.

```
- (void)setDelegate:(id)anObject
```

Availability
Available in Mac OS X v10.4 and later.

See Also
`- delegate` (page 281)

Declared In
`PDFView.h`

setDisplayBox:

Specifies the box to display and to clip to.

```
- (void)setDisplayBox:(PDFDisplayBox)box
```

Discussion
The values for *box* are defined in the `PDFDisplayBox` enumeration. The default value for this method is `kPDFDisplayBoxCropBox`.

The available values for display boxes are defined in the Constants section in the `PDFPage` class.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- displayBox` (page 281)

Declared In
`PDFView.h`

setDisplayMode:

Sets the display mode for the view.

```
- (void)setDisplayMode:(PDFDisplayMode)mode
```

Discussion
Available display modes are single page, single-page continuous, two-up, and two-up continuous, as defined in "Constants" (page 301).

Availability
Available in Mac OS X v10.4 and later.

See Also
- displayMode (page 281)

Declared In
PDFView.h

setDisplaysAsBook:

Specifies whether the view should treat the document's first page as a book cover.

```
- (void)setDisplaysAsBook:(BOOL)asBook
```

Discussion
For two-up modes, a YES value for this method specifies that the first page should be displayed by itself.

Availability
Available in Mac OS X v10.4 and later.

See Also
- displaysAsBook (page 282)

Declared In
PDFView.h

setDisplaysPageBreaks:

Toggles the display of page breaks.

```
- (void)setDisplaysPageBreaks:(BOOL)breaks
```

Availability
Available in Mac OS X v10.4 and later.

See Also
- displaysPageBreaks (page 282)

Declared In
PDFView.h

setDocument:

Associates a document with a `PDFView` object.

```
- (void)setDocument:(PDFDocument *)document
```

Discussion
If a document was already associated with the view, it is released first and then *document* is associated with the view.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `document` (page 282)

Declared In
`PDFView.h`

setGreekingThreshold:

Sets the greeking threshold to use for displaying text.

```
- (void)setGreekingThreshold:(float)threshold
```

Discussion
The default threshold is 3.0.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `greekingThreshold` (page 287)

Declared In
`PDFView.h`

setHighlightedSelections:

Highlights the specified array of selections.

```
- (void)setHighlightedSelections:(NSArray *)selections
```

Discussion
Unlike the selections users set (using, for example, `setCurrentSelection:` (page 292)), the selections you specify in this method do not go away (that is, appear deselected) when users click elsewhere in the view or document. Instead, to deselect the selections, you must call `[setHighlightedSelections:NULL]` to remove them.

You might use this method to highlight the set of matches from a text search. To prevent the user from confusing their own selections with selections you set using this method, it is recommended that you use a highlight color that is different from the user's default text selection color.

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

See Also
- highlightedSelections (page 287)

Declared In
PDFView.h

setScaleFactor:

Sets the scale factor for the view.

```
- (void)setScaleFactor:(float)scale
```

Discussion
The default value is 1.0, corresponding to actual size.

Availability
Available in Mac OS X v10.4 and later.

See Also
- scaleFactor (page 290)

Declared In
PDFView.h

setShouldAntiAlias:

Specifies whether to use anti-aliasing in the view.

```
- (void)setShouldAntiAlias:(BOOL)aliasing
```

Discussion
The default value is YES.

Availability
Available in Mac OS X v10.4 and later.

See Also
- shouldAntiAlias (page 296)

Declared In
PDFView.h

shouldAntiAlias

Returns a Boolean value indicating whether the view is anti-aliased.

```
- (BOOL)shouldAntiAlias
```

Availability
Available in Mac OS X v10.4 and later.

See Also
- `setShouldAntiAlias:` (page 296)

Declared In
`PDFView.h`

takeBackgroundColorFrom:

Sets the view's background color to the specified color.

```
- (IBAction)takeBackgroundColorFrom:(id)sender
```

Discussion
A view's background is the area displayed to either side of a PDF document's pages. The background also appears between pages when page breaks are enabled. The default color is a 50% gray.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `setBackgroundColor:` (page 291)
- `backgroundColor` (page 274)

Declared In
`PDFView.h`

takePasswordFrom:

A convenience method that calls `- [[self document] setpassword:]` with the password from the specified sender.

```
- (void)takePasswordFrom:(id)sender
```

Availability
Available in Mac OS X v10.4 and later.

Declared In
`PDFView.h`

visiblePages

Returns an array of `PDFPage` objects that represent the currently visible pages.

```
- (NSArray *)visiblePages
```

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

Declared In
PDFView.h

zoomIn:

Zooms in by increasing the scaling factor.

```
- (IBAction)zoomIn:(id)sender
```

Discussion
Each invocation of `zoomIn` muliplies the scaling factor by the square root of 2.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `canZoomIn` (page 276)
- `zoomOut:` (page 298)
- `canZoomOut` (page 277)

Declared In
PDFView.h

zoomOut:

Zooms out by decreasing the scaling factor.

```
- (IBAction)zoomOut:(id)sender
```

Discussion
Each invocation of `zoomOut` divides the scaling factor by the square root of 2.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `zoomIn:` (page 298)
- `canZoomIn` (page 276)
- `canZoomOut` (page 277)

Declared In
PDFView.h

Delegate Methods

PDFViewOpenPDF:forRemoteGoToAction:

Delegate method that opens a specified page.

```
- (void)PDFViewOpenPDF:(PDFView *)sender forRemoteGoToAction:(PDFActionRemoteGoTo
    *)action
```

Discussion
A delegate responding to this method is called to handle clicks in an annotation that contains a `PDFActionRemoteGoTo` action. Such an action contains a URL, a page index, and a point. The delegate should open the PDF document specified by the URL and go to the specified page and point. An easy way to do this is to create a `PDFDocument` object with the specified URL and then create a `PDFDestination` object with the specified page and point. Then, you can call `goToDestination:` (page 284).

The default implementation of this method beeps.

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

Declared In
`PDFView.h`

PDFViewPerformFind:

Delegate method that performs a find operation.

```
- (void)PDFViewPerformFind:(PDFView *)sender
```

Discussion
Some `PDFAction` objects request a PDF viewer application to perform a find operation. A delegate responding to this method is called when users click an annotation with such an action.

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

Declared In
`PDFView.h`

PDFViewPerformGoToPage:

Delegate method that performs a go-to operation.

```
- (void)PDFViewPerformGoToPage:(PDFView *)sender
```

Discussion
Some `PDFAction` objects request a PDF viewer application to display a panel that allows users to enter a page number to go to. A delegate responding to this method is called when users click an annotation with such an action.

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

Declared In
`PDFView.h`

PDFViewPerformPrint:

Delegate method that prints the current document.

```
- (void)PDFViewPerformPrint:(PDFView *)sender
```

Discussion
Some `PDFAction` objects request a PDF viewer application to print the current document. A delegate responding to this method is called when users click an annotation with such an action.

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

Declared In
`PDFView.h`

PDFViewPrintJobTitle:

Delegate method that overrides the job title used when the `PDFView` is printed.

```
- (NSString *)PDFViewPrintJobTitle:(PDFView *)sender
```

Discussion
By default, this method uses the string, if any, associated with the "Title" key in the view's `PDFDocument` attribute dictionary. If there is no such string, this method uses the last path component if the document is URL-based.

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

Declared In
`PDFView.h`

PDFViewWillChangeScaleFactor:toScale:

Delegate method for overriding changes to scale factor.

```
- (float)PDFViewWillChangeScaleFactor:(PDFView *)sender toScale:(float)scale
```

Discussion
By default, the scale factor is restricted to a range between 0.1 and 10.0 inclusive.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- setScaleFactor:` (page 296)

Declared In
`PDFView.h`

PDFViewWillClickOnLink:withURL:

Delegate method for handling clicks on URL links in a view.

```
- (void)PDFViewWillClickOnLink:(PDFView *)sender withURL:(NSURL *)url
```

Discussion
By default, this method calls `[[NSWorkspace sharedWorkspace] openURL:url]`.

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

Declared In
`PDFView.h`

Constants

PDF views use the following display mode constants:

Constant	Description
`kPDFDisplaySinglePage`	The document displays one page at a time horizontally and vertically. Vertical and horizontal scrolling apply only to the current page. Available in Mac OS X v10.4 and later. Declared in `PDFView.h`.
`kPDFDisplaySingle-PageContinuous`	The document displays in continuous mode vertically, with single-page width horizontally. Vertical scrolling applies to the entire document. Available in Mac OS X v10.4 and later. Declared in `PDFView.h`.
`kPDFDisplayTwoUp`	The document displays two pages side-by-side. Vertical and horizontal scrolling apply only to the pair of displayed pages Available in Mac OS X v10.4 and later. Declared in `PDFView.h`.
`kPDFDisplayTwo-UpContinuous`	The document displays in continuous mode vertically and displays two pages side-by-side horizontally. Vertical scrolling applies to the entire document. Available in Mac OS X v10.4 and later. Declared in `PDFView.h`.

The following constants apply to mouse position over PDF view areas. These constants are components of a bit field and may be combined arbitrarily:

Constant	Description
`kPDFNoArea`	The mouse is over an undefined area. Available in Mac OS X v10.4 and later. Declared in `PDFView.h`.

Constant	Description
`kPDFPageArea`	The mouse is over a page. Available in Mac OS X v10.4 and later. Declared in `PDFView.h`.
`kPDFTextArea`	The mouse is over text. Available in Mac OS X v10.4 and later. Declared in `PDFView.h`.
`kPDFAnnotationArea`	The mouse is over an annotation. Available in Mac OS X v10.4 and later. Declared in `PDFView.h`.
`kPDFLinkArea`	The mouse is over a link. Available in Mac OS X v10.4 and later. Declared in `PDFView.h`.
`kPDFControlArea`	The mouse is over a control. Available in Mac OS X v10.4 and later. Declared in `PDFView.h`.
`kPDFTextFieldArea`	The mouse is over a text field. Available in Mac OS X v10.4 and later. Declared in `PDFView.h`.
`kPDFIconArea`	The mouse is over an icon. Available in Mac OS X v10.5 and later. Declared in `PDFView.h`.
`kPDFPopupArea`	The mouse is over a popup menu. Available in Mac OS X v10.4 and later. Declared in `PDFView.h`.

Notifications

A `PDFView` object posts the following notifications:

PDFViewChangedHistoryNotification

Posted when the page history changes.

The notification object is the `PDFView` object itself.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`PDFView.h`

PDFViewDocumentChangedNotification

Posted when a new document is associated with the view.

The notification object is the `PDFView` object itself.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`PDFView.h`

PDFViewPageChangedNotification

Posted when a new page becomes the current page.

The notification object is the `PDFView` object itself.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`PDFView.h`

PDFViewScaleChangedNotification

Posted when the scale factor changes.

The notification object is the `PDFView` object itself.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`PDFView.h`

PDFViewAnnotationHitNotification

Posted when the user clicks on an annotation.

The notification object is the `PDFView` object itself.

Use the `@"PDFAnnotationHit"` key to obtain userinfo of type `PDFAnnotation *`.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`PDFView.h`

PDFViewCopyPermissionNotification

Posted when the user attempts to copy to the pasteboard without the appropriate permissions.

The notification object is the `PDFView` object itself.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`PDFView.h`

PDFViewPrintPermissionNotification

Posted when the user attempts to print without the appropriate permissions.

The notification object is the `PDFView` object itself.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`PDFView.h`

PDFViewAnnotationWillHitNotification

Posted before the user clicks an annotation.

The notification object is the `PDFView` object itself.

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

Declared In
`PDFView.h`

PDFViewSelectionChangedNotification

Posted when the current selection has changed.

The notification object is the `PDFView` object itself.

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

Declared In
`PDFView.h`

PDFViewDisplayModeChangedNotification

Posted when the display mode has changed.

The notification object is the `PDFView` object itself.

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

Declared In
`PDFView.h`

PDFViewDisplayBoxChangedNotification

Posted when the display box has changed.

The notification object is the `PDFView` object itself.

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

Declared In
`PDFView.h`

QCComposition Class Reference

Inherits from	NSObject
Conforms to	NSCopying NSObject (NSObject)
Framework	/System/Library/Frameworks/Quartz.framework/Frameworks/QuartzComposer.framework
Availability	Available in Mac OS X v10.5 and later.
Declared in	QuartzComposer/QCComposition.h
Companion guide	Quartz Composer Programming Guide

Overview

The `QCComposition` class represents a Quartz Composer composition that either:

- comes from the system-wide composition repository (`/Library/Compositions` and `~/Library/Compositions`) where it can be accessed by any application through the methods of the `QCCompositionRepository` class
- is created from an arbitrary source (typically a file on disk) using one of its methods

This class cannot be subclassed.

A `QCComposition` object has the following information associated with it and that you can obtain by using the appropriate method of the `QCComposition` class:

- Attributes include the name and description of the composition, copyright information, and whether or not its provided by Mac OS X (built-in).
- The protocols that the composition conforms to. A **composition protocol** defines a set of required and optional input parameters and output results.

Many methods of the `QCRenderer`, `QCCompositionLayer`, and `QCView` classes take a `QCComposition` object as a parameter.

Tasks

Creating a Composition

`+ compositionWithFile:` (page 309)
Returns an autoreleased composition object initialized with a Quartz Composer composition file.

`+ compositionWithData:` (page 308)
Returns an autoreleased composition object initialized with the contents of a Quartz Composer composition file.

Getting Information About a Composition

`- attributes` (page 309)
Returns the attributes of the composition.

`- protocols` (page 310)
Returns the list of protocols to which the composition conforms.

`- identifier` (page 309)
Returns the unique and persistent identifier for the composition from the composition repository.

Getting Port Keys

`- inputKeys` (page 310)
Returns an array listing the keys that identify the input ports of the root patch of the composition.

`- outputKeys` (page 310)
Returns an array listing the keys that identify the output ports of the root patch of the composition.

Class Methods

compositionWithData:

Returns an autoreleased composition object initialized with the contents of a Quartz Composer composition file.

```
+ (QCComposition*) compositionWithData:(NSData*)data;
```

Parameters

data
The contents of a file created with the Quartz Composer developer tool.

Return Value
A Quartz Composer composition object or `nil` if there is an error.

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

Declared In
QCComposition.h

compositionWithFile:

Returns an autoreleased composition object initialized with a Quartz Composer composition file.

```
+ (QCComposition*) compositionWithFile:(NSString*)path;
```

Parameters

path

A path to a file created with the Quartz Composer developer tool (`.qtz` extension).

Return Value
A Quartz Composer composition object or `nil` if there is an error.

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

Declared In
QCComposition.h

Instance Methods

attributes

Returns the attributes of the composition.

```
- (NSDictionary*) attributes
```

Return Value
A dictionary of composition attributes. See "Attribute Keys" (page 311) for the attributes that can be returned.

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

Declared In
QCComposition.h

identifier

Returns the unique and persistent identifier for the composition from the composition repository.

```
- (NSString*) identifier
```

Return Value
The unique identifier for the composition if it comes from the composition repository; `nil` otherwise.

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

Declared In
QCCompositionRepository.h

inputKeys

Returns an array listing the keys that identify the input ports of the root patch of the composition.

```
- (NSArray*) inputKeys
```

Return Value
An array of input keys.

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

Declared In
QCComposition.h

outputKeys

Returns an array listing the keys that identify the output ports of the root patch of the composition.

```
- (NSArray*) outputKeys
```

Return Value
An array of output keys.

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

Declared In
QCComposition.h

protocols

Returns the list of protocols to which the composition conforms.

```
- (NSArray*) protocols
```

Return Value
A list of protocols. See "Standard Protocols" (page 315).

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

Declared In
QCComposition.h

Constants

Attribute Keys

Attributes of a composition.

```
extern NSString* const QCCompositionAttributeNameKey;
extern NSString* const QCCompositionAttributeDescriptionKey;
extern NSString* const QCCompositionAttributeCopyrightKey;
extern NSString* const QCCompositionAttributeBuiltInKey;
extern NSString* const QCCompositionAttributeTimeDependentKey;
extern NSString* const QCCompositionAttributeHasConsumersKey;
extern NSString* const QCCompositionAttributeCategoryKey;
```

Constants

`QCCompositionAttributeNameKey`
The key for the composition name. The associated value is an `NSString` object.

Available in Mac OS X v10.4 and later.

Declared in `QCComposition.h`.

`QCCompositionAttributeDescriptionKey`
The key for the composition description. The associated value is an `NSString` object.

Available in Mac OS X v10.4 and later.

Declared in `QCComposition.h`.

`QCCompositionAttributeCopyrightKey`
The key for composition copyright information. The associated value is an `NSString` object.

Available in Mac OS X v10.4 and later.

Declared in `QCComposition.h`.

`QCCompositionAttributeBuiltInKey`
The key for the composition origin. The associated value is an `NSNumber` object that contains a Boolean value. `YES` indicates the composition is built-in (provided by Mac OS X).

Available in Mac OS X v10.5 and later.

Declared in `QCComposition.h`.

`QCCompositionAttributeTimeDependentKey`
The key for the composition time dependency. The associated value is an `NSNumber` object that contains a Boolean value. `YES` indicates that the composition is time dependent.

`QCCompositionAttributeHasConsumersKey`
The key for a composition that has consumer patches. The associated value is an `NSNumber` object that contains a Boolean value. `YES` indicates that the composition has consumers.

Available in Mac OS X v10.5 and later.

Declared in `QCComposition.h`.

`QCCompositionAttributeCategoryKey`
The composition category. The associated value is a category constant. See "Composition Categories" (page 312).

Available in Mac OS X v10.5 and later.

Declared in `QCComposition.h`.

Declared In
QCComposition.h

Composition Categories

Categories for compositions.

```
extern NSString* const QCCompositionCategoryDistortion;
extern NSString* const QCCompositionCategoryStylize;
extern NSString* const QCCompositionCategoryUtility;
```

Constants

QCCompositionCategoryDistortion
: A composition that produces a distortion effect.

  Available in Mac OS X v10.5 and later.

  Declared in QCComposition.h.

QCCompositionCategoryStylize
: A composition that produces a stylize effect.

  Available in Mac OS X v10.5 and later.

  Declared in QCComposition.h.

QCCompositionCategoryUtility
: A utility composition.

  Available in Mac OS X v10.5 and later.

  Declared in QCComposition.h.

Declared In
QCComposition.h

Standard Protocol Input Keys

Input ports of a composition.

```
extern NSString* const QCCompositionInputImageKey;
extern NSString* const QCCompositionInputSourceImageKey;
extern NSString* const QCCompositionInputDestinationImageKey;
extern NSString* const QCCompositionInputRSSFeedURLKey;
extern NSString* const QCCompositionInputRSSArticleDurationKey;
extern NSString* const QCCompositionInputPreviewModeKey;
extern NSString* const QCCompositionInputXKey;
extern NSString* const QCCompositionInputYKey;
extern NSString* const QCCompositionInputScreenImageKey;
extern NSString* const QCCompositionInputAudioPeakKey;
extern NSString* const QCCompositionInputAudioSpectrumKey;
extern NSString* const QCCompositionInputTrackPositionKey;
extern NSString* const QCCompositionInputTrackInfoKey;
extern NSString* const QCCompositionInputTrackSignalKey;
extern NSString* const QCCompositionInputPrimaryColorKey;
extern NSString* const QCCompositionInputSecondaryColorKey;
extern NSString* const QCCompositionInputPaceKey;
```

Constants

`QCCompositionInputImageKey`
An image input port whose key is `inputImage`.

Available in Mac OS X v10.5 and later.

Declared in `QCComposition.h`.

`QCCompositionInputSourceImageKey`
An image input port whose key is `inputSourceImage`.

Available in Mac OS X v10.5 and later.

Declared in `QCComposition.h`.

`QCCompositionInputDestinationImageKey`
An image input port whose key is `inputDestinationImage`.

Available in Mac OS X v10.5 and later.

Declared in `QCComposition.h`.

`QCCompositionInputRSSFeedURLKey`
A string input port whose key is `inputRSSFeedURL`. This port must be passed an http or feed scheme URL.

Available in Mac OS X v10.5 and later.

Declared in `QCComposition.h`.

`QCCompositionInputRSSArticleDurationKey`
A number input port whose key is `inputRSSArticleDuration`. The value must be expressed in seconds.

Available in Mac OS X v10.5 and later.

Declared in `QCComposition.h`.

`QCCompositionInputPreviewModeKey`
A Boolean input port whose key is `inputPreviewMode`. When the value of this input port is set to `TRUE`, the composition that provides this port must be able to run in a low-quality mode that produces a preview of the composition.

Available in Mac OS X v10.5 and later.

Declared in `QCComposition.h`.

`QCCompositionInputXKey`
A number input port whose key is `inputX`. The value must be normalized to the image width with the origin on the left.

Available in Mac OS X v10.5 and later.

Declared in `QCComposition.h`.

`QCCompositionInputYKey`
A number input port whose key is `inputY`. The value must be normalized to the image height with the origin at the bottom.

Available in Mac OS X v10.5 and later.

Declared in `QCComposition.h`.

`QCCompositionInputScreenImageKey`
An image input port whose key is `inputScreenImage`.

Available in Mac OS X v10.5 and later.

Declared in `QCComposition.h`.

`QCCompositionInputAudioPeakKey`
A number input port whose key is `inputAudioPeak`. The value must be in the `[0,1]` range as a mono signal with no decay applied.

Available in Mac OS X v10.5 and later.

Declared in `QCComposition.h`.

`QCCompositionInputAudioSpectrumKey`
A structure input port whose key is `inputAudioSpectrum`. The structure must contain 16 values in the `[0,1]` range representing 16 spectrum bands of the mono signal from low to high frequencies with no decay applied.

Available in Mac OS X v10.5 and later.

Declared in `QCComposition.h`.

`QCCompositionInputTrackPositionKey`
A number input port whose key is `inputTrackPosition`. The value must be expressed in seconds.

Available in Mac OS X v10.5 and later.

Declared in `QCComposition.h`.

`QCCompositionInputTrackInfoKey`
A structure input port whose key is `inputTrackInfo`. The structure contains optional entries, such as “name”, “artist”, "album", "duration", "artwork", and so on.

Available in Mac OS X v10.5 and later.

Declared in `QCComposition.h`.

`QCCompositionInputTrackSignalKey`
A Boolean input port whose key is `inputTrackSignal`.

Available in Mac OS X v10.5 and later.

Declared in `QCComposition.h`.

`QCCompositionInputPrimaryColorKey`
A color input port whose key is `inputPrimaryColor`.

Available in Mac OS X v10.5 and later.

Declared in `QCComposition.h`.

`QCCompositionInputSecondaryColorKey`
A color input port whose key is `inputSecondaryColor`.

Available in Mac OS X v10.5 and later.

Declared in `QCComposition.h`.

`QCCompositionInputPaceKey`
A number input port whose key is `inputPace`. The value must be in the `[0,1]` range.

Available in Mac OS X v10.5 and later.

Declared in `QCComposition.h`.

Declared In
`QCComposition.h`

Standard Protocol Output Keys

Output ports of a composition.

```
extern NSString* const QCCompositionOutputImageKey;
extern NSString* const QCCompositionOutputWebPageURLKey;
```

Constants

`QCCompositionOutputImageKey`

An image output port whose key is `outputImage`.

Available in Mac OS X v10.5 and later.

Declared in `QCComposition.h`.

`QCCompositionOutputWebPageURLKey`

A string output port whose key is `outputWebPageURL`.

Available in Mac OS X v10.5 and later.

Declared in `QCComposition.h`.

Declared In

`QCComposition.h`

Standard Protocols

Protocols for a composition.

```
extern NSString* const QCCompositionProtocolGraphicAnimation;
extern NSString* const QCCompositionProtocolGraphicTransition;
extern NSString* const QCCompositionProtocolImageFilter;
extern NSString* const QCCompositionProtocolImageCompositor;
extern NSString* const QCCompositionProtocolImageTransition;
extern NSString* const QCCompositionProtocolScreenSaverRSS;
```

Constants

`QCCompositionProtocolGraphicAnimation`

A composition that renders a generic graphical animation. It has the option to use `QCCompositionInputPrimaryColorKey` (page 314) for the primary color of the animation, `QCCompositionInputSecondaryColorKey` (page 314) for the secondary color of the animation, `QCCompositionInputPaceKey` (page 314) for the global pace of the animation, and `QCCompositionInputPreviewModeKey` (page 313) to indicate if the animation should run in lower-quality for preview purposes.

Available in Mac OS X v10.5 and later.

Declared in `QCComposition.h`.

`QCCompositionProtocolGraphicTransition`

A composition that performs a transition between two images, using a transition time in range of `0` to `1`. A conforming composition must use the input keys `QCCompositionInputSourceImageKey` (page 313) for the starting image and `QCCompositionInputDestinationImageKey` (page 313) for the image to transition to. The composition can optionally use `QCCompositionInputPreviewModeKey` (page 313) to indicate if the animation should run in lower-quality for preview purposes.

Available in Mac OS X v10.5 and later.

Declared in `QCComposition.h`.

`QCCompositionProtocolImageFilter`

A composition that applies an effect to a source image. A conforming composition must use the input key `QCCompositionInputImageKey` (page 313) for the source image and `QCCompositionOutputImageKey` (page 315) for the output image. The composition can optionally use `QCCompositionInputXKey` (page 313) to specify the X position of the center point of the effect, `QCCompositionInputYKey` (page 313) to specify the Y position of the center point of the effect, and`QCCompositionInputPreviewModeKey` (page 313) to indicate if the animation should run in lower-quality for preview purposes.

Available in Mac OS X v10.5 and later.

Declared in `QCComposition.h`.

`QCCompositionProtocolScreenSaver`

A composition that can be used as a screen saver. The composition has the option to use `QCCompositionInputScreenImageKey` (page 313) for a screenshot image of the screen that the screen saver runs on, `QCCompositionInputPreviewModeKey` (page 313) to indicate if the animation should run in lower-quality for preview purposes, and `QCCompositionOutputWebPageURLKey` (page 315) for a URL to open in the default web browser when screen saver exits (only allowed if screen saver password is disabled).

Available in Mac OS X v10.5 and later.

Declared in `QCComposition.h`.

`QCCompositionProtocolImageTransition`

A composition that performs a transition between two images, using a parametric time value to drives the transition from start (at time `0`) to end (at time `1`). A conforming composition must use the input keys `QCCompositionInputImageKey` (page 313) for the starting image and `QCCompositionInputDestinationImageKey` (page 313) for the ending image. The composition can optionally use `QCCompositionInputPreviewModeKey` (page 313) to indicate if the animation should run in lower-quality for preview purposes.

`QCCompositionProtocolRSSVisualizer`

A composition that acts as a visualizer for an RSS feed. A conforming composition must use the input key `QCCompositionInputRSSFeedURLKey` (page 313) for the URL to use for the RSS feed. It can optionally use `QCCompositionInputRSSArticleDurationKey` (page 313) to specify the duration of each feed article.

Available in Mac OS X v10.5 and later.

Declared in `QCComposition.h`.

`QCCompositionProtocolMusicVisualizer`

A composition that acts as a visualizer for music. A conforming composition must use the input key `QCCompositionInputAudioPeakKey` (page 314) for the instantaneous audio peak and the `QCCompositionInputAudioSpectrumKey` (page 314) for the instantaneous audio spectrum. It can optionally use the `QCCompositionInputTrackInfoKey` (page 314) to indicate it receives information about the current track and the `QCCompositionInputTrackSignalKey` (page 314) to indicate the start of a new track.

Available in Mac OS X v10.5 and later.

Declared in `QCComposition.h`.

Declared In

`QCComposition.h`

QCCompositionLayer Class Reference

Inherits from	CAOpenGLLayer : CALayer : NSObject
Conforms to	QCCompositionRenderer NSCoding (CALayer) CAMediaTiming (CALayer) NSObject (NSObject)
Framework	/System/Library/Frameworks/Quartz.framework/Frameworks/QuartzComposer.framework
Availability	Available in Mac OS X v10.5 and later.
Declared in	QuartzComposer/QCCompositionLayer.h
Companion guides	Core Animation Programming Guide Quartz Composer Programming Guide
Related sample code	CALayerEssentials

Overview

The `QCCompositionLayer` class loads, plays, and controls Quartz Composer compositions in a Core Animation layer hierarchy. The composition tracks the Core Animation layer time and is rendered directly at the current dimensions of the `QCCompositionLayer` object.

An archived `QCCompositionLayer` object saves the composition that's loaded at the time the layer is archived. It detects layer usage and pauses or resumes the composition appropriately. A `QCCompositionLayer` object starts rendering the composition automatically when the layer is placed in a visible layer hierarchy. The layer stops rendering when it is hidden or removed from the visible layer hierarchy.

You can pass data to the input ports, or retrieve data from the output ports, of the root patch of a composition by accessing the `patch` attribute of the `QCCompositionLayer` instance using methods provided by the `QCCompositionRenderer` protocol.

Note: You must not modify the `asynchronous` property of the superclass `CAOpenGLLayer`.

Tasks

Creating the Layer

`+ compositionLayerWithFile:` (page 319)
Creates and returns an instance of a composition layer using the Quartz Composer composition in the specified file.

`+ compositionLayerWithComposition:` (page 318)
Creates and returns an instance of a composition layer using the provided Quartz Composer composition.

`- initWithFile:` (page 320)
Initializes and returns a composition layer using the Quartz Composer composition in the specified file.

`- initWithComposition:` (page 319)
Initializes and returns a composition layer using the provided Quartz Composer composition.

Getting the Composition

`- composition` (page 319)
Returns the composition associated with the layer.

Class Methods

compositionLayerWithComposition:

Creates and returns an instance of a composition layer using the provided Quartz Composer composition.

```
+ (QCCompositionLayer*)compositionLayerWithComposition:(QCComposition*)composition
```

Parameters

composition
The Quartz Composer composition to use as content.

Return Value
An autoreleased, initialized `QCCompositionLayer` object or `nil` if initialization is not successful.

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

See Also
`+ compositionLayerWithFile:` (page 319)

Declared In
QCCompositionLayer.h

compositionLayerWithFile:

Creates and returns an instance of a composition layer using the Quartz Composer composition in the specified file.

```
+ (QCCompositionLayer*)compositionLayerWithFile:(NSString*)path
```

Parameters

path
A string that specifies the location of a Quartz Composer composition.

Return Value
An autoreleased, initialized `QCCompositionLayer` object or `nil` if initialization is not successful.

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

See Also
`+ compositionLayerWithComposition:` (page 318)

Related Sample Code
CALayerEssentials

Declared In
QCCompositionLayer.h

Instance Methods

composition

Returns the composition associated with the layer.

```
- (QCComposition*) composition
```

Return Value
The composition object associated with the layer or `nil` if there is none.

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

Declared In
QCCompositionLayer.h

initWithComposition:

Initializes and returns a composition layer using the provided Quartz Composer composition.

```
- (id)initWithComposition:(QCComposition*)composition
```

Parameters

composition

The Quartz Composer composition to use as content.

Return Value

The initialized `QCCompositionLayer` object or `nil` if initialization is not successful.

Availability

Available in Mac OS X v10.5 and later.

See Also

`- initWithFile:` (page 320)

Declared In

`QCCompositionLayer.h`

initWithFile:

Initializes and returns a composition layer using the Quartz Composer composition in the specified file.

```
- (id)initWithFile:(NSString*)path
```

Parameters

path

A string that specifies the location of a Quartz Composer composition.

Return Value

The initialized `QCCompositionLayer` object or `nil` if initialization is not successful.

Availability

Available in Mac OS X v10.5 and later.

See Also

`- initWithComposition:` (page 319)

Declared In

`QCCompositionLayer.h`

QCCompositionParameterView Class Reference

Inherits from	NSView : NSResponder : NSObject
Conforms to	NSAnimatablePropertyContainer (NSView) NSCoding (NSResponder) NSObject (NSObject)
Framework	/System/Library/Frameworks/Quartz.framework/Frameworks/QuartzComposer.framework
Availability	Available in Mac OS X v10.5 and later.
Declared in	QuartzComposer/QCCompositionParameterView.h
Companion guide	Quartz Composer Programming Guide

Overview

The `QCCompositionParameterView` class allows allows users to edit, in real time, the input parameters of a composition. The composition can be rendering in any of the following objects: `QCRenderer`, `QCView`, or `QCCompositionLayer`.

Tasks

Getting and Setting the Renderer

- `setCompositionRenderer:` (page 324)
  Sets the composition parameter view for editing the input parameters of the provided renderer object.
- `compositionRenderer` (page 322)
  Returns the renderer object associated with the composition parameter view.

Checking for Input Parameters

- `hasParameters` (page 323)
  Checks whether the composition that is currently edited by the composition parameter view has any input parameters.

Setting and Retrieving the Delegate

- `setDelegate:` (page 324)
  Sets the composition parameter view delegate.
- `delegate` (page 323)
  Returns the composition parameter view delegate.

Managing Background Drawing

- `setDrawsBackground:` (page 325)
  Sets whether the composition parameter view draws its background.
- `drawsBackground` (page 323)
  Returns whether the composition parameter view draws its background.

Setting and Getting the Background Color

- `setBackgroundColor:` (page 324)
  Sets the background color of the composition parameter view.
- `backgroundColor` (page 322)
  Retrieves the background color of the composition parameter view.

Instance Methods

backgroundColor

Retrieves the background color of the composition parameter view.

```
- (NSColor*) backgroundColor;
```

Return Value
The color of the background.

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

Declared In
`QCCompositionParameterView.h`

compositionRenderer

Returns the renderer object associated with the composition parameter view.

```
- (id<QCCompositionRenderer>) compositionRenderer
```

Return Value
A renderer object or `nil`, if the composition parameter view is not set to a renderer object.

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

See Also
`- setCompositionRenderer:` (page 324)

Declared In
`QCCompositionParameterView.h`

delegate

Returns the composition parameter view delegate.

```
- (id) delegate;
```

Return Value
The composition parameter view delegate.

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

Declared In
`QCCompositionParameterView.h`

drawsBackground

Returns whether the composition parameter view draws its background.

```
- (BOOL) drawsBackground;
```

Return Value
`YES` if the view draws its background; otherwise `NO`.

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

Declared In
`QCCompositionParameterView.h`

hasParameters

Checks whether the composition that is currently edited by the composition parameter view has any input parameters.

```
- (BOOL) hasParameters
```

Return Value
`YES` if the composition has any input parameters.

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

Declared In
`QCCompositionParameterView.h`

setBackgroundColor:

Sets the background color of the composition parameter view.

```
- (void) setBackgroundColor:(NSColor*)color;
```

Parameters

color

The color to set.

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

Declared In
`QCCompositionParameterView.h`

setCompositionRenderer:

Sets the composition parameter view for editing the input parameters of the provided renderer object.

```
- (void) setCompositionRenderer:(id<QCCompositionRenderer>)renderer
```

Parameters

renderer

A `QCCompositionRenderer` object, either `QCView`, `QCRenderer`, or `QCCompositionLayer`. Pass `nil` to unset this renderer.

Discussion
If the renderer is a `QCView` object, the view track the composition.

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

See Also
`- compositionRenderer` (page 322)

Declared In
`QCCompositionParameterView.h`

setDelegate:

Sets the composition parameter view delegate.

```
- (void) setDelegate:(id)delegate;
```

Parameters

delegate

The delegate for the composition parameter view.

Availability

Available in Mac OS X v10.5 and later.

Declared In

QCCompositionParameterView.h

setDrawsBackground:

Sets whether the composition parameter view draws its background.

```
- (void) setDrawsBackground:(BOOL)flag;
```

Parameters

flag

YES for the view to draw its background; otherwise NO.

Availability

Available in Mac OS X v10.5 and later.

Declared In

QCCompositionParameterView.h

QCCompositionPickerPanel Class Reference

Inherits from	NSPanel : NSWindow : NSResponder : NSObject
Conforms to	NSUserInterfaceValidations (NSWindow) NSAnimatablePropertyContainer (NSWindow) NSCoding (NSResponder) NSObject (NSObject)
Framework	/System/Library/Frameworks/Quartz.framework/Frameworks/QuartzComposer.framework
Availability	Available in Mac OS X v10.5 and later.
Declared in	QuartzComposer/QCCompositionPickerPanel.h
Companion guide	Quartz Composer Programming Guide

Overview

The `QCCompositionPickerPanel` class represents a utility window that allows users to browse compositions that are in the Quartz Composer composition repository and, if supported, preview the composition. The `QCCompositionPickerPanel` class cannot be subclassed.

Tasks

Creating the Utility Window for Browsing Compositions

`+ sharedCompositionPickerPanel` (page 328)
Returns the shared instance of the composition picker panel.

Getting the Picker Panel View

`- compositionPickerView` (page 328)
Returns the composition picker view used by the panel so that it can be configured.

Class Methods

sharedCompositionPickerPanel

Returns the shared instance of the composition picker panel.

```
+ (QCCompositionPickerPanel*) sharedCompositionPickerPanel
```

Return Value
The shared `QCCompositionPickerPanel` object.

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

Declared In
`QCCompositionPickerPanel.h`

Instance Methods

compositionPickerView

Returns the composition picker view used by the panel so that it can be configured.

```
- (QCCompositionPickerView*) compositionPickerView;
```

Return Value
The `QCCompositionPickerView` used by the composition picker panel.

Discussion
After you retrieve the view, you can configure it.

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

Declared In
`QCCompositionPickerPanel.h`

Notifications

QCCompositionPickerPanelDidSelectCompositionNotification

Posted when the user chooses a composition.

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

Declared In
`QCCompositionPickerPanel.h`

QCCompositionPickerView Class Reference

Inherits from	NSView : NSResponder : NSObject
Conforms to	NSAnimatablePropertyContainer (NSView) NSCoding (NSResponder) NSObject (NSObject)
Framework	/System/Library/Frameworks/Quartz.framework/Frameworks/QuartzComposer.framework
Availability	Available in Mac OS X v10.5 and later.
Declared in	QuartzComposer/QCCompositionPickerView.h
Companion guide	Quartz Composer Programming Guide

Overview

The `QCCompositionPickerView` class allows users to browse compositions that are in the Quartz Composer composition repository, and to preview them. You can set the default input parameters for a composition preview by using the method `setDefaultValue:forInputKey:`.

Note that the composition picker view does not automatically refresh its content when the composition repository is updated. It's your responsibility to perform any necessary updating.

Tasks

Setting and Getting the Background Color

- `setBackgroundColor:` (page 336)
  Sets the background color for the composition picker view.
- `backgroundColor` (page 332)
  Returns the background color of the composition picker view.

Managing Background Drawing

- `setDrawsBackground:` (page 338)
  Sets whether the composition picker view draws its background.

- `drawsBackground` (page 333)
  Returns whether the composition picker view draws its background.

Setting Composition Input Parameters

- `setDefaultValue:forInputKey:` (page 337)
  Sets the default value to use for a composition input parameter.
- `resetDefaultInputValues` (page 335)
  Clears all previously set default values for composition input parameters.

Managing Animation

- `startAnimation:` (page 340)
  Starts animating the composition in the composition picker view.
- `stopAnimation:` (page 340)
  Stops animating the composition that is currently animating in the composition picker view.
- `isAnimating` (page 333)
  Returns whether or not the composition picker view is currently animating its composition.
- `setMaxAnimationFrameRate:` (page 338)
  Sets the maximum frame rate for animating compositions.
- `maxAnimationFrameRate` (page 334)
  Retrieves the maximum frame rate for animating compositions.

Controlling Display of Composition Names

- `setShowsCompositionNames:` (page 339)
  Enables the display of composition names in the composition picker view.
- `showsCompositionNames` (page 340)
  Retrieves whether composition names can be shown in the composition picker view.

Setting and Retrieving the View Delegate

- `setDelegate:` (page 337)
  Sets the composition picker view delegate.
- `delegate` (page 333)
  Retrieves the composition picker view delegate.

Managing the Composition Picker View

- `setCompositionsFromRepositoryWithProtocol:andAttributes:` (page 336)
  Sets the compositions in the composition picker view to those that match the specified criteria.

- `compositions` (page 332)
  Returns the list of compositions that are currently in the composition picker view.
- `setAllowsEmptySelection:` (page 335)
  Sets whether to allow an empty selection in the composition picker view.
- `allowsEmptySelection` (page 331)
  Retrieves the empty-selection state of the composition picker view.
- `setCompositionAspectRatio:` (page 336)
  Sets the aspect ratio used to display compositions in the composition picker view.
- `compositionAspectRatio` (page 332)
  Retrieves the aspect ratio used to display compositions in the composition picker view.
- `setSelectedComposition:` (page 339)
  Sets a composition as selected in the composition picker view.
- `selectedComposition` (page 335)
  Returns the composition that is currently selected in the composition picker view.

Working with Columns and Rows

- `setNumberOfColumns:` (page 338)
  Sets the number of columns in the composition picker view.
- `numberOfColumns` (page 334)
  Retrieves the number of columns in the composition picker view.
- `setNumberOfRows:` (page 339)
  Sets the number of rows in the composition picker view.
- `numberOfRows` (page 334)
  Retrieves the number of rows in the composition picker view.

Instance Methods

allowsEmptySelection

Retrieves the empty-selection state of the composition picker view.

```
- (BOOL) allowsEmptySelection
```

Return Value
`YES` if an empty selection is allowed `NO` otherwise.

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

See Also
- `setAllowsEmptySelection:` (page 335)

Declared In
`QCCompositionPickerView.h`

backgroundColor

Returns the background color of the composition picker view.

```
- (NSColor*) backgroundColor;
```

Return Value
The background color.

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

See Also
- `setBackgroundColor:` (page 336)

Declared In
`QCCompositionPickerView.h`

compositionAspectRatio

Retrieves the aspect ratio used to display compositions in the composition picker view.

```
- (NSSize) compositionAspectRatio
```

Return Value
The aspect ratio.

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

See Also
- `setCompositionAspectRatio:` (page 336)

Declared In
`QCCompositionPickerView.h`

compositions

Returns the list of compositions that are currently in the composition picker view.

```
- (NSArray*) compositions
```

Return Value
An array of `QCComposition` objects.

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

See Also
- `setCompositionsFromRepositoryWithProtocol:andAttributes:` (page 336)

Declared In
`QCCompositionPickerView.h`

delegate

Retrieves the composition picker view delegate.

```
- (id) delegate
```

Return Value
The delegate.

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

See Also
- `setDelegate:` (page 337)

Declared In
`QCCompositionPickerView.h`

drawsBackground

Returns whether the composition picker view draws its background.

```
- (BOOL) drawsBackground;
```

Return Value
`YES` if the composition picker view draws its background; otherwise `NO`.

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

See Also
- `setDrawsBackground:` (page 338)

Declared In
`QCCompositionPickerView.h`

isAnimating

Returns whether or not the composition picker view is currently animating its composition.

```
- (BOOL) isAnimating
```

Return Value
`YES` if a composition is animating in the composition picker view; `NO` otherwise.

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

See Also
- `startAnimation:` (page 340)
- `stopAnimation:` (page 340)

Declared In
`QCCompositionPickerView.h`

maxAnimationFrameRate

Retrieves the maximum frame rate for animating compositions.

```
- (float) maxAnimationFrameRate
```

Return Value
The maximum frame rate.

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

See Also
- setMaxAnimationFrameRate: (page 338)

Declared In
QCCompositionPickerView.h

numberOfColumns

Retrieves the number of columns in the composition picker view.

```
- (NSUInteger) numberOfColumns;
```

Return Value
The number of columns.

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

See Also
- setNumberOfColumns: (page 338)

Declared In
QCCompositionPickerView.h

numberOfRows

Retrieves the number of rows in the composition picker view.

```
- (NSUInteger) numberOfRows;
```

Return Value
The number of columns.

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

See Also
- setNumberOfRows: (page 339)

Declared In
QCCompositionPickerView.h

resetDefaultInputValues

Clears all previously set default values for composition input parameters.

```
- (void) resetDefaultInputValues
```

Discussion
This method resets the defaults that were set with the method `setDefaultValue:forInputKey:` (page 337).

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

Declared In
`QCCompositionPickerView.h`

selectedComposition

Returns the composition that is currently selected in the composition picker view.

```
- (QCComposition*) selectedComposition
```

Return Value
A `QCComposition` object, or `nil` if a composition is not selected.

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

See Also
`- setSelectedComposition:` (page 339)

Declared In
`QCCompositionPickerView.h`

setAllowsEmptySelection:

Sets whether to allow an empty selection in the composition picker view.

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

Parameters
flag
`YES` to allow an empty selection. The default value is `NO`.

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

See Also
`- allowsEmptySelection` (page 331)

Declared In
`QCCompositionPickerView.h`

setBackgroundColor:

Sets the background color for the composition picker view.

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

Parameters

aColor

The color for the background.

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

See Also
- backgroundColor (page 332)

Declared In
QCCompositionPickerView.h

setCompositionAspectRatio:

Sets the aspect ratio used to display compositions in the composition picker view.

```
- (void) setCompositionAspectRatio:(NSSize)ratio
```

Parameters

ratio

An aspect ratio.

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

See Also
- compositionAspectRatio (page 332)

Declared In
QCCompositionPickerView.h

setCompositionsFromRepositoryWithProtocol:andAttributes:

Sets the compositions in the composition picker view to those that match the specified criteria.

```
- (void) setCompositionsFromRepositoryWithProtocol:(NSString*)protocol
    andAttributes:(NSDictionary*)attributes
```

Parameters

protocol

The protocols that you want compositions shown in the picker view to conform to. You can pass any of these protocols: QCCompositionProtocolAnimation, QCCompositionProtocolImageProducer, QCCompositionProtocolImageFilter, QCCompositionProtocolImageCompositor, QCCompositionProtocolImageTransition, and QCCompositionProtocolScreenSaverRSS.

attributes

A dictionary that contains the attributes, and their associated values, that you want compositions in the picker view to match. For example, you can pass: `QCCompositionAttributeNameKey`, `QCCompositionAttributeDescriptionKey`, `QCCompositionAttributeCopyrightKey`, `QCCompositionAttributeBuiltInKey`, and `QCCompositionAttributeTimeDependentKey`. Pass `nil` if you don't want to filter based on the attributes.

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

See Also
- `compositions` (page 332)

Declared In
`QCCompositionPickerView.h`

setDefaultValue:forInputKey:

Sets the default value to use for a composition input parameter.

```
- (void) setDefaultValue:(id)value forInputKey:(NSString*)key
```

Parameters

value

This default value overrides any initial value existing for composition input parameters with this key. Pass `nil` to clear the default value.

key

The input parameter key whose default value you want to set.

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

See Also
- `resetDefaultInputValues` (page 335)

Declared In
`QCCompositionPickerView.h`

setDelegate:

Sets the composition picker view delegate.

```
- (void) setDelegate:(id)delegate
```

Parameters

delegate

The delegate to set.

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

See Also
- `delegate` (page 333)

Declared In
QCCompositionPickerView.h

setDrawsBackground:

Sets whether the composition picker view draws its background.

```
- (void) setDrawsBackground:(BOOL)flag;
```

Parameters

flag

The background drawing state. Pass YES if the composition picker view draws its background.

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

See Also
- drawsBackground (page 333)

Declared In
QCCompositionPickerView.h

setMaxAnimationFrameRate:

Sets the maximum frame rate for animating compositions.

```
- (void) setMaxAnimationFrameRate:(float)maxFPS
```

Parameters

maxFPS

A frame rate in frames per second. Pass 0.0 to specify no limit to the maximum value.

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

See Also
- maxAnimationFrameRate (page 334)

Declared In
QCCompositionPickerView.h

setNumberOfColumns:

Sets the number of columns in the composition picker view.

```
- (void) setNumberOfColumns:(NSUInteger)columns;
```

Parameters

columns

The number of columns.

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

See Also
- `numberOfColumns` (page 334)

Declared In
`QCCompositionPickerView.h`

setNumberOfRows:

Sets the number of rows in the composition picker view.

```
- (void) setNumberOfRows:(NSUInteger)rows;
```

Parameters

columns

The number of rows.

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

See Also
- `numberOfRows` (page 334)

Declared In
`QCCompositionPickerView.h`

setSelectedComposition:

Sets a composition as selected in the composition picker view.

```
- (void) setSelectedComposition:(QCComposition*)composition
```

Parameters

composition

The composition to select. Pass `nil` if you don't want to select a composition. The behavior is undefined if you pass a composition that is not in the list of compositions that are currently in the composition picker view.

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

See Also
- `selectedComposition` (page 335)

Declared In
`QCCompositionPickerView.h`

setShowsCompositionNames:

Enables the display of composition names in the composition picker view.

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

Parameters

flag

YES specifies to show compositions name. The default value is NO.

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

Declared In
QCCompositionPickerView.h

showsCompositionNames

Retrieves whether composition names can be shown in the composition picker view.

```
- (BOOL) showsCompositionNames
```

Return Value
YES if the display of names is enabled; otherwise NO.

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

Declared In
QCCompositionPickerView.h

startAnimation:

Starts animating the composition in the composition picker view.

```
- (void) startAnimation:(id)sender
```

Parameters

sender

The object initiating the animation.

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

See Also
- stopAnimation: (page 340)
- isAnimating (page 333)

Declared In
QCCompositionPickerView.h

stopAnimation:

Stops animating the composition that is currently animating in the composition picker view.

```
- (void) stopAnimation:(id)sender
```

Parameters

sender

The object stopping the animation.

Availability

Available in Mac OS X v10.5 and later.

See Also

- `startAnimation:` (page 340)
- `isAnimating` (page 333)

Declared In

`QCCompositionPickerView.h`

Notifications

QCCompositionPickerViewDidSelectCompositionNotification

Posted when the user selects a composition in the picker view.

Availability

Available in Mac OS X v10.5 and later.

Declared In

`QCCompositionPickerView.h`

QCCompositionRepository Class Reference

Inherits from	NSObject
Conforms to	NSObject (NSObject)
Framework	/System/Library/Frameworks/Quartz.framework/Frameworks/QuartzComposer.framework
Availability	Available in Mac OS X v10.5 and later.
Declared in	QuartzComposer/QCCompositionRepository.h
Companion guide	Quartz Composer Programming Guide

Overview

The `QCCompositionRepository` class represents a system-wide centralized repository of built-in and installed Quartz Composer compositions (`/Library/Compositions` and `~/Library/Compositions`). The `QCCompositionRepository` class cannot be subclassed.

Compositions in the repository are represented by the `QCComposition` class. You can use the methods of the `QCCompositionRepository` class to fetch all compositions or only those that meet specific criteria.

Tasks

Getting the Composition Repository

`+ sharedCompositionRepository` (page 344)
Returns the shared instance of the composition repository.

Fetching Compositions

`- compositionWithIdentifier:` (page 345)
Returns the composition that corresponds to the identifier.

`- compositionsWithProtocols:andAttributes:` (page 344)
Returns an array of compositions that match a set of criteria.

`- allCompositions` (page 344)
Returns an array that contains all compositions currently in the composition repository.

Class Methods

sharedCompositionRepository

Returns the shared instance of the composition repository.

```
+ (QCCompositionRepository*) sharedCompositionRepository
```

Return Value
The shared instance of `QCCompositionRepository`.

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

Declared In
`QCCompositionRepository.h`

Instance Methods

allCompositions

Returns an array that contains all compositions currently in the composition repository.

```
- (NSArray*) allCompositions
```

Return Value
An array of `QCComposition` objects.

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

See Also
`- compositionWithIdentifier:` (page 345)
`- compositionsWithProtocols:andAttributes:` (page 344)

Declared In
`QCCompositionRepository.h`

compositionsWithProtocols:andAttributes:

Returns an array of compositions that match a set of criteria.

```
- (NSArray*) compositionsWithProtocols:(NSArray*)protocols
    andAttributes:(NSDictionary*)attributes
```

Parameters

protocols

The protocols that you want compositions to conform to. Pass `nil` if you don't want to filter based on the protocol. You can pass any of these protocols: `QCCompositionProtocolAnimation`, `QCCompositionProtocolImageProducer`, `QCCompositionProtocolImageFilter`, `QCCompositionProtocolImageCompositor`, `QCCompositionProtocolImageTransition`, and `QCCompositionProtocolScreenSaverRSS`.

attributes

A dictionary that contains the attributes, and their associated values, that you want compositions to match. Pass `nil` if you don't want to filter based on the attributes. For example, you can pass any of these attributes: `QCCompositionAttributeNameKey`, `QCCompositionAttributeDescriptionKey`, `QCCompositionAttributeCopyrightKey`, `QCCompositionAttributeBuiltInKey`, and `QCCompositionAttributeTimeDependentKey`.

Return Value

An array of `QCComposition` objects that meet the supplied criteria.

Availability

Available in Mac OS X v10.5 and later.

See Also

- `compositionWithIdentifier:` (page 345)
- `allCompositions` (page 344)

Declared In

`QCCompositionRepository.h`

compositionWithIdentifier:

Returns the composition that corresponds to the identifier.

```
- (QCComposition*) compositionWithIdentifier:(NSString*)identifier
```

Parameters

identifier

A string that uniquely identifies the composition to retrieve.

Return Value

The composition identified by the provided string, or `nil` if there is no composition with that identifier in the composition repository.

Availability

Available in Mac OS X v10.5 and later.

See Also

- `compositionsWithProtocols:andAttributes:` (page 344)
- `allCompositions` (page 344)

Declared In

`QCCompositionRepository.h`

Notifications

QCCompositionRepositoryDidUpdateNotification

Posted whenever the list of compositions in the composition repository is updated.

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

Declared In
`QCCompositionRepository.h`

QCPlugIn Class Reference

Inherits from	NSObject
Conforms to	NSObject (NSObject)
Framework	/System/Library/Frameworks/Quartz.framework/Frameworks/QuartzComposer.framework
Availability	Available in Mac OS X v10.5 and later.
Declared in	QuartzComposer/QCPlugIn.h
Companion guides	Quartz Composer Custom Patch Programming Guide Quartz Composer Programming Guide

Overview

The `QCPlugIn` class provides the base class to subclass for writing custom Quartz Composer patches. You implement a custom patch by subclassing `QCPlugIn`, overriding the appropriate methods, packaging the code as an `NSBundle` object, and installing the bundle in the appropriate location. A bundle can contain more than one subclass of `QCPlugIn`, allowing you to provide a suite of custom patches in one bundle. *Quartz Composer Custom Patch Programming Guide* provides detailed instructions on how to create and package a custom patch. *QCPlugIn Class Reference* supplements the information in the programming guide.

The methods related to the executing the custom patch (called when the Quartz Composer engine is rendering) are passed an opaque object that conforms to the `QCPlugInContext Protocol` protocol. This object represents the execution context of the `QCPlugIn` object. You should not retain the execution context or use it outside of the scope of the execution method that it is passed to.

Tasks

Defining the Characteristics of a Custom Patch

`+ executionMode` (page 351)
Returns the execution mode of the custom patch.

`+ timeMode` (page 353)
Returns the time mode for the custom patch.

Executing a Custom Patch

- `execute:atTime:withArguments:` (page 357)
  Performs the processing or rendering tasks appropriate for the custom patch.

Performing Custom Tasks During Execution

- `startExecution:` (page 360)
  Allows you to perform custom setup tasks before the Quartz Composer engine starts rendering.
- `enableExecution:` (page 357)
  Allows you to perform custom tasks when the execution of the `QCPlugIn` object is resumed.
- `disableExecution:` (page 356)
  Allows you to perform custom tasks when the execution of the `QCPlugIn` object is paused.
- `stopExecution:` (page 361)
  Allows you to perform custom tasks when the `QCPlugIn` object stops executing.

Defining Patch and Property Port Attributes

+ `attributes` (page 349)
  Returns a dictionary that contains strings for the user interface that describe the custom patch.
+ `attributesForPropertyPortWithKey:` (page 350)
  Returns a dictionary that contains strings for the user interface that describe the optional attributes for ports created from properties.

Defining Internal Settings

- `createViewController` (page 355)
  Creates and returns a view controller for the Settings pane of a custom patch.
+ `plugInKeys` (page 352)
  Returns the keys for the internal settings of a custom patch.

Supporting Saving and Retrieving Internal Settings

- `serializedValueForKey:` (page 359)
  Provides custom serialization for patch internal settings that do not comply to the `NSCoding` protocol.
- `setSerializedValue:forKey:` (page 359)
  Provides custom deserialization for patch internal settings that were previously serialized using the method `serializedValueForKey:` (page 359).

Adding Ports Dynamically

- `addInputPortWithType:forKey:withAttributes:` (page 354)
  Adds an input port of the specified type and associates a key and attributes with the port.

- `removeInputPortForKey:` (page 358)
  Removes the input port for a given key.
- `addOutputPortWithType:forKey:withAttributes:` (page 354)
  Adds an output port of the specified type and associates a key and attributes with the port.
- `removeOutputPortForKey:` (page 358)
  Removes the output port for a given key.

Getting and Setting Port Values

- `didValueForInputKeyChange:` (page 356)
  Returns whether the input port value changed since the last execution of the custom patch.
- `valueForInputKey:` (page 361)
  Returns the current value for an input port.
- `setValue:forOutputKey:` (page 360)
  Sets the value of an output port.

Loading Bundle and Custom Patches Manually

+ `loadPlugInAtPath:` (page 351)
  Loads a Quartz Composer plug-in bundle from the specified path.
+ `registerPlugInClass:` (page 352)
  Registers a `QCPlugIn` subclass.

Ordering Property Ports

+ `sortedPropertyPortKeys` (page 353)
  Returns and array of property port keys in the order you want them to appear in the user interface.

Class Methods

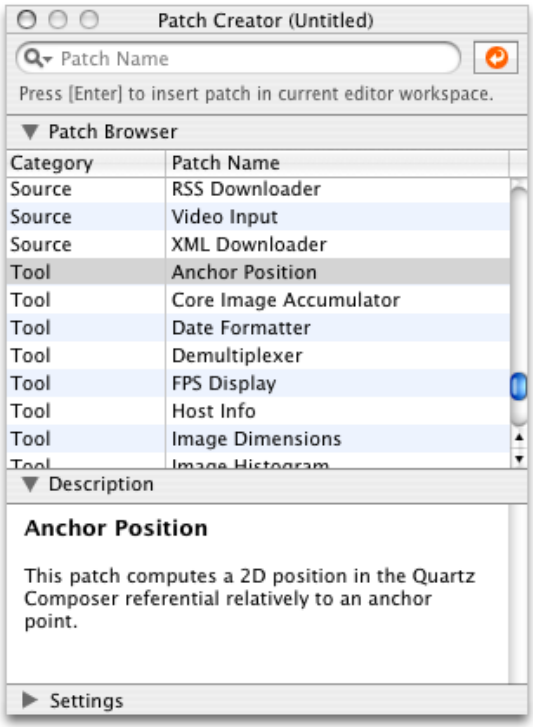
attributes

Returns a dictionary that contains strings for the user interface that describe the custom patch.

```
+ (NSDictionary*) attributes
```

Return Value
The dictionary can contain one or more of these keys along with the appropriate string: `QCPlugInAttributeNameKey` (page 362), `QCPlugInAttributeDescriptionKey` (page 362), and `QQCPlugInAttributeCopyrightKey` (page 362).

Discussion
It's recommended that you implement this method to enhance the experience of those who use your custom patch. The attribute name string that you provide is displayed in the Quartz Composer editor window when the custom patch name is selected in the Patch Creator (see figure). The attribute description key is displayed in the Information pane of the inspector for the custom patch.



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

See Also
`+ attributesForPropertyPortWithKey:` (page 350)

Declared In
`QCPlugIn.h`

attributesForPropertyPortWithKey:

Returns a dictionary that contains strings for the user interface that describe the optional attributes for ports created from properties.

```
+ (NSDictionary*) attributesForPropertyPortWithKey:(NSString*)key
```

Parameters

key

The name of the property.

Return Value
A dictionary that contains key-value pairs for the port's attributes. The keys must be one or more of the constants defined in "Input and Output Port Attributes" (page 362).

Discussion
It's recommended that you implement this method to enhance the experience of those who use your custom patch. The attributes appear in a help tag when the user hovers a pointer over the property port on your custom patch. At a minimum, you should provide a user-readable name for the port. It might also be helpful to provide default, minimum, and maximum values for the port.

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

See Also
`+ attributes` (page 349)

Declared In
`QCPlugIn.h`

executionMode

Returns the execution mode of the custom patch.

```
+ (QCPlugInExecutionMode) executionMode
```

Return Value
The execution mode of the custom patch. See "`Execution Modes`" (page 366) for the constants you can return.

Discussion
You must implement this method to define whether your custom patch is a provider, a processor, or a consumer.

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

Declared In
`QCPlugIn.h`

loadPlugInAtPath:

Loads a Quartz Composer plug-in bundle from the specified path.

```
+ (BOOL) loadPlugInAtPath:(NSString*)path
```

Parameters

path
The location of the bundle.

Return Value
`YES` if successful.

Discussion
Call this method only if you need to load a plug-in bundle from a nonstandard location. Typically you don't need to call this method because Quartz Composer automatically loads bundles that you install in one of the following locations:

- `/Library/Graphics/Quartz Composer Plug-Ins`

- `~/Library/Graphics/Quartz Composer Plug-Ins`

This method does nothing if the bundle is already loaded. (This method does not load in all environments. Web Kit, for example, cannot load custom patches.)

The bundle can contain more than one `QCPlugIn` subclass. After the bundle is loaded, each `QCPlugIn` subclass appears as a patch in the Quartz Composer patch library.

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

Declared In
`QCPlugIn.h`

plugInKeys

Returns the keys for the internal settings of a custom patch.

```
+ (NSArray*) plugInKeys
```

Return Value
An array of keys used for key-value coding (KVC) of the internal settings.

Discussion
You must override this method if your patch provides a Settings pane. This keys are used for automatic serialization of the internal settings and are also used by the `QCPlugInViewController` instance for the Settings pane. The implementation is straightforward; the keys are strings that represent the instance variables used for the Settings pane. For example, the `plugInKeys` method for these instance variables:

```
@property(ivar, byref) NSColor * systemColor;
@property(ivar, byref) NSConfiguration * systemConfiguration;
```

are:

```
+ (NSArray*) plugInKeys
{
    return [NSArray arrayWithObjects: @"systemColor",
                                      @"systemConfiguration",
                                      nil];
}
```

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

See Also
`- createViewController` (page 355)

Declared In
`QCPlugIn.h`

registerPlugInClass:

Registers a `QCPlugIn` subclass.

```
+ (void) registerPlugInClass:(Class)aClass
```

Parameters

aClass

The `QCPlugIn` subclass.

Discussion

You call this method only if the code for your custom patch is mixed with your application code, and you plan only to use the custom patch from within your application.

Availability

Available in Mac OS X v10.5 and later.

Declared In

`QCPlugIn.h`

sortedPropertyPortKeys

Returns and array of property port keys in the order you want them to appear in the user interface.

```
+ (NSArray*) sortedPropertyPortKeys;
```

Return Value

The property port keys in the order you want them to appear in the user interface.

Discussion

Override this method to specify an optional ordering for property based ports in the user interface.

Availability

Available in Mac OS X v10.5 and later.

Declared In

`QCPlugIn.h`

timeMode

Returns the time mode for the custom patch.

```
+ (QCPlugInTimeMode) timeMode
```

Return Value

The time mode of the custom patch. See "`Time Modes`" (page 367) for the constants you can return.

Discussion

You must implement this method to define whether you custom patch depends on time, doesn't depend on time, or needs time to idle.

Availability

Available in Mac OS X v10.5 and later.

Declared In

`QCPlugIn.h`

Instance Methods

addInputPortWithType:forKey:withAttributes:

Adds an input port of the specified type and associates a key and attributes with the port.

```
- (void) addInputPortWithType:(NSString*)type forKey:(NSString*)key
    withAttributes:(NSDictionary*)attributes
```

Parameters

type

The port type. See "Port Input and Output Types" (page 363).

key

The key to associate with the port.

attributes

A dictionary of attributes for the port. See "Input and Output Port Attributes" (page 362). Although the dictionary is optional, it's recommended that provide attributes to enhance the experience of those who use your custom patch. The attributes appear in a help tag when the user hovers a pointer over the property port on your custom patch. (See `attributesForPropertyPortWithKey:` (page 350).) Pass `nil` if you do not want to provide attributes.

Discussion

This method throws an exception if called from within the `execute:atTime:withArguments:` (page 357) method or if there's already an input or output port with that key.

Availability

Available in Mac OS X v10.5 and later.

See Also

`- removeInputPortForKey:` (page 358)

Declared In

`QCPlugIn.h`

addOutputPortWithType:forKey:withAttributes:

Adds an output port of the specified type and associates a key and attributes with the port.

```
- (void) addOutputPortWithType:(NSString*)type forKey:(NSString*)key
    withAttributes:(NSDictionary*)attributes
```

Parameters

type

The port type. See "Port Input and Output Types" (page 363).

key

The key to associate with the port.

attributes

A dictionary of attributes for the port. See "Input and Output Port Attributes" (page 362). Although the dictionary is optional, it's recommended that provide attributes to enhance the experience of those who use your custom patch. The attributes appear in a help tag when the user hovers a pointer over the property port on your custom patch. (See `attributesForPropertyPortWithKey:` (page 350).) Pass `nil` if you do not want to provide attributes.

Discussion
This method throws an exception if called from within the `execute:atTime:withArguments:` (page 357) method or if there is already an output port with that key.

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

See Also
`- removeOutputPortForKey:` (page 358)

Declared In
`QCPlugIn.h`

createViewController

Creates and returns a view controller for the Settings pane of a custom patch.

```
- (QCPlugInViewController*) createViewController
```

Return Value
A view controller for the custom patch. Quartz Composer releases the controller when it is no longer needed. If necessary, you can return a subclass of `QCPlugInViewController`, but this it not typically done.

Discussion
This extension to the `QCPlugInViewController` class provides user-interface support for the Settings pane of the inspector for a custom patch. You must override this method if your custom patch provides a Settings pane. The `QCPlugInViewController` object acts as a controller for Cocoa bindings between the custom patch instance (the model) and the `NSView` that contains the controls. It loads the nib file from the bundle.

The implementation is straightforward. You allocate a `QCPlugInViewController` object, initialize it, and provide the name of the nib file that contains the user interface for the Settings pane.

Note that this method follows the Core Foundation "create" rule. See the ownership policy in *Memory Management Programming Guide for Core Foundation*.

For example, if the nib file name that contains the settings pane is `MySettingsPane.nib`, the implementation is:

```
- (QCPlugInViewController *) createViewController
{
    return [[QCPlugInViewController alloc] initWithPlugIn:self
                                              viewNibName:@"MySettingsPane"];
}
```

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

See Also
`+ plugInKeys` (page 352)

Declared In
`QCPlugInViewController.h`

didValueForInputKeyChange:

Returns whether the input port value changed since the last execution of the custom patch.

```
- (BOOL) didValueForInputKeyChange:(NSString*)key
```

Parameters

key

The key for the input port whose value you want to check.

Return Value
`YES` if the value on the input port changed since the last time the `execute:atTime:withArguments:` (page 357) method was called; always returns `NO` if called outside of the `execute:atTime:withArguments:` method.

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

See Also
`- valueForInputKey:` (page 361)

Declared In
`QCPlugIn.h`

disableExecution:

Allows you to perform custom tasks when the execution of the `QCPlugIn` object is paused.

```
- (void) disableExecution:(id<QCPlugInContext>)context
```

Parameters

context

An opaque object , conforming to the `QCPlugInContext Protocol` protocol, that represents the execution context of the `QCPlugIn` object. Do not retain this object or use it outside of the scope of this method.

Discussion
The Quartz Composer engine calls this method when results are no longer being pulled from the custom patch. You can optionally override this execution method to perform custom tasks at that time.

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

See Also
`- enableExecution:` (page 357)

Declared In
`QCPlugIn.h`

enableExecution:

Allows you to perform custom tasks when the execution of the `QCPlugIn` object is resumed.

```
- (void) enableExecution:(id<QCPlugInContext>)context
```

Parameters

context

An opaque object , conforming to the `QCPlugInContext Protocol` protocol, that represents the execution context of the `QCPlugIn` object. Do not retain this object or use it outside of the scope of this method.

Discussion

The Quartz Composer engine calls this method when results start to be pulled from the custom patch. You can optionally override this execution method to perform custom tasks at that time.

Availability

Available in Mac OS X v10.5 and later.

See Also

`- disableExecution:` (page 356)

Declared In

`QCPlugIn.h`

execute:atTime:withArguments:

Performs the processing or rendering tasks appropriate for the custom patch.

```
- (BOOL) execute:(id<QCPlugInContext>)context atTime:(NSTimeInterval)time
    withArguments:(NSDictionary*)arguments
```

Parameters

context

An opaque object , conforming to the `QCPlugInContext Protocol` protocol, that represents the execution context of the `QCPlugIn` object. Do not retain this object or use it outside of the scope of this method.

time

The execution interval.

arguments

A dictionary of arguments that can be used during execution. See "`Execution Arguments`" (page 365).

Return Value

`NO` indicates the custom patch was not able to execute successfully. In this case, the Quartz Composer engine stops rendering the current frame.

Discussion

The Quartz Composer engine calls this method each time your custom patch needs to execute. You must implement this method. The method should perform whatever tasks are appropriate for the custom patch, such as:

- reading values from the input ports
- computing output values

- updating the values on the output ports
- rendering to the execution context

For example implementations of this method, see *Quartz Composer Custom Patch Programming Guide*.

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

Declared In
`QCPlugIn.h`

removeInputPortForKey:

Removes the input port for a given key.

```
- (void) removeInputPortForKey:(NSString*)key
```

Parameters

key

The key associated with the port that you want to remove.

Discussion
This method throws an exception if from within the `execute:atTime:withArguments:` (page 357) method, if there is not an input port with that key, or if the port is created from a property.

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

See Also
`- addInputPortWithType:forKey:withAttributes:` (page 354)

Declared In
`QCPlugIn.h`

removeOutputPortForKey:

Removes the output port for a given key.

```
- (void) removeOutputPortForKey:(NSString*)key
```

Parameters

key

The key associated with the port that you want to remove.

Discussion
This method throws an exception if called from within the `execute:atTime:withArguments:` (page 357) method, if there is not an output port with that key, or if the port is created from a property.

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

See Also
`- addOutputPortWithType:forKey:withAttributes:` (page 354)

Declared In
QCPlugIn.h

serializedValueForKey:

Provides custom serialization for patch internal settings that do not comply to the NSCoding protocol.

```
- (id) serializedValueForKey:(NSString*)key
```

Parameters

key
The key for the value to retrieve.

Return Value
Either nil or a value that's compliant with property lists: NSString, NSNumber, NSDate, NSData, NSArray, or NSDictionary.

Discussion
If your patch has internal settings that do not conform to the NSCoding protocol, you must implement this method.

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

See Also
- setSerializedValue:forKey: (page 359)

Declared In
QCPlugIn.h

setSerializedValue:forKey:

Provides custom deserialization for patch internal settings that were previously serialized using the method serializedValueForKey: (page 359).

```
- (void) setSerializedValue:(id)serializedValue forKey:(NSString*)key
```

Parameters

serializedValue
The value to deserialize.

key
The key for the value to deserialize.

Discussion
If your patch has internal settings that do not conform to the NSCoding protocol, you must implement this method. After you deserialize the value, you need to call [self set:value forKey:key] to set the corresponding internal setting of the custom patch instance to the deserialized value.

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

Declared In
QCPlugIn.h

setValue:forOutputKey:

Sets the value of an output port.

```
- (BOOL) setValue:(id)value forOutputKey:(NSString*)key
```

Parameters

key

The key associated with the output port whose value you want to set.

Return Value

`YES` if successful; `NO` if called outside of the `execute:atTime:withArguments:` (page 357) method.

Discussion

You call this method from within your `execute:atTime:withArguments:` (page 357) method to set the output values of your custom patch.

Availability

Available in Mac OS X v10.5 and later.

See Also

- `valueForInputKey:` (page 361)
- `didValueForInputKeyChange:` (page 356)

Declared In

`QCPlugIn.h`

startExecution:

Allows you to perform custom setup tasks before the Quartz Composer engine starts rendering.

```
- (BOOL) startExecution:(id<QCPlugInContext>)context
```

Parameters

context

An opaque object , conforming to the `QCPlugInContext Protocol` protocol, that represents the execution context of the `QCPlugIn` object. Do not retain this object or use it outside of the scope of this method.

Return Value

`NO` indicates a fatal error occurred and prevents the Quartz Composer engine from starting.

Discussion

The Quartz Composer engine calls this method when your custom patch starts to render. You can optionally override this execution method to perform setup tasks.

Availability

Available in Mac OS X v10.5 and later.

See Also

- `stopExecution:` (page 361)

Declared In

`QCPlugIn.h`

stopExecution:

Allows you to perform custom tasks when the `QCPlugIn` object stops executing.

```
- (void) stopExecution:(id<QCPlugInContext>)context
```

Parameters

context

An opaque object , conforming to the `QCPlugInContext Protocol` protocol, that represents the execution context of the `QCPlugIn` object. Do not retain this object or use it outside of the scope of this method.

Discussion
The Quartz Composer engine calls this method when it stops executing. You can optionally override this execution method to perform cleanup tasks.

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

See Also
`- startExecution:` (page 360)

Declared In
`QCPlugIn.h`

valueForInputKey:

Returns the current value for an input port.

```
- (id) valueForInputKey:(NSString*)key
```

Parameters

key

The key for the input port you want to check.

Return Value
The value associated with the key or `nil` if called outside of the `execute:atTime:withArguments:` (page 357) method.

Discussion
You call this method from within your `execute:atTime:withArguments:` (page 357) method to retrieve the input values of your custom patch.

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

See Also
`- setValue:forOutputKey:` (page 360)
`- didValueForInputKeyChange:` (page 356)

Declared In
`QCPlugIn.h`

Constants

Patch Attributes

Attributes for custom patches.

```
extern NSString* const QCPlugInAttributeNameKey;
extern NSString* const QCPlugInAttributeDescriptionKey;
extern NSString* const QCPlugInAttributeCopyrightKey;
```

Constants

`QCPlugInAttributeNameKey`

The key for the custom patch name. The associated value is an `NSString` object.

Available in Mac OS X v10.5 and later.

Declared in `QCPlugIn.h`.

`QCPlugInAttributeDescriptionKey`

The key for the custom patch description. The associated value is an `NSString` object.

Available in Mac OS X v10.5 and later.

Declared in `QCPlugIn.h`.

`QQCPlugInAttributeCopyrightKey`

The key for the custom patch copyright information. The associated value is an `NSString` object.

Declared In

`QCPlugIn.h`

Input and Output Port Attributes

Attributes for input and output ports.

```
extern NSString* const QCPortAttributeTypeKey;
extern NSString* const QCPortAttributeNameKey;
extern NSString* const QCPortAttributeDefaultValueKey;
extern NSString* const QCPortAttributeMinimumValueKey;
extern NSString* const QCPortAttributeMaximumValueKey;
extern NSString* const QCPortAttributeDefaultValueKey;
extern NSString* const QCPortAttributeMenuItemsKey;
```

Constants

`QCPortAttributeTypeKey`

The key for the port type. The associated value can be of any of the following constants: `QCPortTypeBoolean` (page 364), `QCPortTypeIndex` (page 364), `QCPortTypeNumber` (page 364), `QCPortTypeString` (page 364), `QCPortTypeColor` (page 364), `QCPortTypeImage` (page 364), or `QCPortTypeStructure` (page 364).

Available in Mac OS X v10.4 and later.

Declared in `QCPlugIn.h`.

`QCPortAttributeNameKey`

The key for the port name. The associated value is an `NSString` object.

Available in Mac OS X v10.4 and later.

Declared in `QCPlugIn.h`.

`QCPortAttributeMinimumValueKey`
The key for the port minimum value. The associated value is an `NSNumber` object that specifies the minimum numerical value accepted by the port.

Available in Mac OS X v10.4 and later.

Declared in `QCPlugIn.h`.

`QCPortAttributeMaximumValueKey`
The key for the port maximum value. The associated value is an `NSNumber` object that specifies the maximum numerical value accepted by the port.

Available in Mac OS X v10.4 and later.

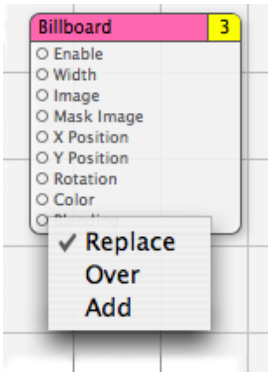
Declared in `QCPlugIn.h`.

`QCPortAttributeDefaultValueKey`
The key for the port default value. You can use this key only for value ports (Boolean, Index, Number, Color and String).

Available in Mac OS X v10.5 and later.

Declared in `QCPlugIn.h`.

`QCPortAttributeMenuItemsKey`
The key for the menu items. The associated value is an array of strings that are displayed in the user interface as a pop-up menu when the user double-clicks a port, as shown for the Blending input port



of the Billboard patch. You can use this key only for an index port.

Available in Mac OS X v10.5 and later.

Declared in `QCPlugIn.h`.

Declared In
`QCPlugIn.h`

Port Input and Output Types

Data types for input and output ports.

```
extern NSString* const QCPortTypeBoolean;
extern NSString* const QCPortTypeIndex;
extern NSString* const QCPortTypeNumber;
extern NSString* const QCPortTypeString;
extern NSString* const QCPortTypeColor;
extern NSString* const QCPortTypeImage;
extern NSString* const QCPortTypeStructure;
```

Constants

`QCPortTypeBoolean`

The port type for a Boolean value. The associated value can be an `NSNumber` object or any object that responds to the `-intValue`, `-floatValue`, or `-doubleValue` methods.

Available in Mac OS X v10.4 and later.

Declared in `QCPlugIn.h`.

`QCPortTypeIndex`

The port type for an index value. The associated value can be an `NSNumber` object or any object that responds to the `-intValue`, `-floatValue`, or `-doubleValue` methods.

Available in Mac OS X v10.4 and later.

Declared in `QCPlugIn.h`.

`QCPortTypeNumber`

The port type for a number value. The associated value can be an `NSNumber` object or any object that responds to the `-intValue`, `-floatValue`, or `-doubleValue` methods.

Available in Mac OS X v10.4 and later.

Declared in `QCPlugIn.h`.

`QCPortTypeString`

The port type for a string. The associated value can be an `NSString` object or any object that responds to the `-stringValue` or `-description` methods.

Available in Mac OS X v10.4 and later.

Declared in `QCPlugIn.h`.

`QCPortTypeColor`

The port type for a color value. The associated value must be an `NSColor` object.

Available in Mac OS X v10.4 and later.

Declared in `QCPlugIn.h`.

`QCPortTypeImage`

The port type for an image. The associated value can be an `NSImage` object or a `CIImage` object.

Available in Mac OS X v10.4 and later.

Declared in `QCPlugIn.h`.

`QCPortTypeStructure`

The port type for an array, dictionary, or other structure, such as an `NSArray` or `NSDictionary` object.

Available in Mac OS X v10.4 and later.

Declared in `QCPlugIn.h`.

Declared In

`QCPlugIn.h`

Pixel Formats

Supported image pixel formats.

```
extern NSString* const QCPlugInPixelFormatARGB8;
extern NSString* const QCPlugInPixelFormatBGRA8;
extern NSString* const QCPlugInPixelFormatRGBAf;
extern NSString* const QCPlugInPixelFormatI8;
extern NSString* const QCPlugInPixelFormatIf;
```

Constants

`QCPlugInPixelFormatARGB8`

An ARGB8 format. The alpha component is stored in the most significant bits of each pixel. Each pixel component is 8 bits. For best performance, use this format on PowerPC-based Macintosh computers, as it represents of the order of the data in memory.

Available in Mac OS X v10.5 and later.

Declared in `QCPlugIn.h`.

`QCPlugInPixelFormatBGRA8`

A BGRA8 format. The alpha component is stored in the least significant bits of each pixel. Each pixel component is 8 bits. For best performance, use this format on Intel-PC-based Macintosh computers, as it represents of the order of the data in memory.

Available in Mac OS X v10.5 and later.

Declared in `QCPlugIn.h`.

`QCPlugInPixelFormatRGBAf`

An RGBAf format. Pixel components are represented as floating-point values.

Available in Mac OS X v10.5 and later.

Declared in `QCPlugIn.h`.

`QCPlugInPixelFormatI8`

An I8 format. Intensity information is represented as an 8-bit value.

Available in Mac OS X v10.5 and later.

Declared in `QCPlugIn.h`.

`QCPlugInPixelFormatIf`

An If format. Intensity information is represented as a floating-point value.

Available in Mac OS X v10.5 and later.

Declared in `QCPlugIn.h`.

Declared In

`QCPlugIn.h`

Execution Arguments

Arguments to the method `execute:atTime:withArguments:` (page 357).

```
extern NSString* const QCPlugInExecutionArgumentEventKey;
extern NSString* const QCPlugInExecutionArgumentMouseLocationKey;
```

Constants

`QCPlugInExecutionArgumentEventKey`

The current `NSEvent` if available.

Available in Mac OS X v10.5 and later.

Declared in `QCPlugIn.h`.

`QCPlugInExecutionArgumentMouseLocationKey`

The current location of the mouse (as an `NSPoint` object stored in an `NSValue` object) in normalized coordinates relative to the OpenGL context viewport ([0,1]x[0,1] with the origin (`0,0`) at the lower-left corner).

Available in Mac OS X v10.5 and later.

Declared in `QCPlugIn.h`.

Declared In

`QCPlugIn.h`

Execution Modes

Execution modes for custom patches.

```
typedef enum {
    kQCPlugInExecutionModeProvider = 1,
    kQCPlugInExecutionModeProcessor,
    kQCPlugInExecutionModeConsumer
} QCPlugInExecutionMode;
```

Constants

`kQCPlugInExecutionModeProvider`

A provider execution mode. The custom patch executes on demand—that is, whenever data is requested of it, but at most once per frame.

Available in Mac OS X v10.5 and later.

Declared in `QCPlugIn.h`.

`kQCPlugInExecutionModeProcessor`

A processor execution mode. The custom patch executes whenever its inputs change or if the time change (assuming it's time-dependent).

Available in Mac OS X v10.5 and later.

Declared in `QCPlugIn.h`.

`kQCPlugInExecutionModeConsumer`

A consumer execution mode. The custom patch always executes assuming the value of its Enable input port is `true`. (The Enable port is automatically added by the system.)

Available in Mac OS X v10.5 and later.

Declared in `QCPlugIn.h`.

Declared In

`QCPlugIn.h`

Time Modes

Time modes for custom patches.

```
typedef enum {
    kQCPlugInTimeModeNone = 0,
    kQCPlugInTimeModeIdle,
    kQCPlugInTimeModeTimeBase
} QCPlugInTimeMode;
```

Constants

`kQCPlugInTimeModeNone`

No time dependency. The custom patch does not depend on time at all. (It does not use the `time` parameter of the `execute:atTime:withArguments:` method.)

Available in Mac OS X v10.5 and later.

Declared in `QCPlugIn.h`.

`kQCPlugInTimeModeIdle`

An idle time dependency. The custom patch does not depend on time but needs the system to execute it periodically. For example if the custom patch connects to a piece of hardware, to ensure that it pulls data from the hardware, you would set the custom patch time dependency to idle time mode. This time mode is typically used with providers.]]

Available in Mac OS X v10.5 and later.

Declared in `QCPlugIn.h`.

`kQCPlugInTimeModeTimeBase`

A time base dependency. The custom patch does depend on time explicitly and has a time base defined by the system. (It uses the `time` parameter of the `execute:atTime:withArguments:` method.)

Available in Mac OS X v10.5 and later.

Declared in `QCPlugIn.h`.

Declared In

`QCPlugIn.h`

QCPlugInViewController Class Reference

Inherits from	NSViewController : NSResponder : NSObject
Conforms to	NSCoding (NSResponder) NSObject (NSObject)
Framework	/System/Library/Frameworks/Quartz.framework/Frameworks/QuartzComposer.framework
Availability	Available in Mac OS X v10.5 and later.
Declared in	QuartzComposer/QCPlugInViewController.h
Companion guides	Quartz Composer Custom Patch Programming Guide Quartz Composer Programming Guide

Overview

The `QCPlugInViewController` class communicates (through Cocoa bindings) between a custom patch and the view used for the internal settings of the custom patch. Only custom patches that use internal settings exposed to the user need to use the `QCPlugInViewController` class.

You access the internal settings of a custom patch through key-value coding (KVC). All the KVC keys that represent the internal settings of the custom patch must be listed in its `plugInKeys` method.

The view controller for a custom patch expects

- the nib file `File's Owner` class set to the `QCPlugInViewController` class
- the view outlet connected to the view that contains the editing controls

The controls are bound to the `File's Owner` as the target and `plugIn.XXX` as the model key path, where `XXX` is the KVC key for a given internal setting of the custom patch instance.

Tasks

Creating a Controller

`- initWithPlugIn:viewNibName:` (page 370)
Creates and initializes a controller for the specified `QCPlugIn` object and nib file.

Getting the QCPlugIn Object

`- plugIn` (page 370)
Returns the `QCPlugIn` object associated with the view controller for the custom patch.

Instance Methods

initWithPlugIn:viewNibName:

Creates and initializes a controller for the specified `QCPlugIn` object and nib file.

```
- (id) initWithPlugIn:(QCPlugIn*)plugIn viewNibName:(NSString*)name
```

Parameters

plugIn
A `QCPlugIn` object that uses internal settings.

name
The name of the nib file that contains the view for the custom patch.

Return Value
A `QCPlugInViewController` object.

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

Declared In
`QCPlugInViewController.h`

plugIn

Returns the `QCPlugIn` object associated with the view controller for the custom patch.

```
- (QCPlugIn*) plugIn
```

Return Value
The `QCPlugIn` object associated with the view controller for the custom patch.

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

Declared In
`QCPlugInViewController.h`

QCRenderer Class Reference

Inherits from	NSObject
Conforms to	QCCompositionRenderer NSObject (NSObject)
Framework	/System/Library/Frameworks/Quartz.framework/Frameworks/QuartzComposer.framework
Declared in	QuartzComposer/QCRenderer.h
Availability	Available in Mac OS X v10.4 and later.

Overview

A `QCRenderer` class is designed for low-level rendering of Quartz Composer compositions. This is the class to use if you want to be in charge of rendering a composition to a specific OpenGL context—either using the `NSOpenGLContext` class or a `CGLContextObj` object. `QCRenderer` also allows you to load, play, and control a composition.

To render a composition to a specific OpenGL context:

- Create an instance of `QCRenderer` using one of the initialization methods, such as `initWithOpenGLContext:pixelFormat:file:` (page 375).
- Render frames by calling the method `renderAtTime:arguments:` (page 375)
- If you use double buffering in OpenGL, you must swap the OpenGL buffers.
- Release the renderer with you no longer need it.

This code snippet shows how to implement these tasks:

```
NSOpenGLContext*     context = [myNSOpenGLView openGLContext];
NSOpenGLPixelFormat*  format = [myNSOpenGLView pixelFormat];
NSString*               path = @"/Users/MyName/MyComposition.qtz";
QCRenderer* myRenderer;
// Create a Quartz Composer renderer.
myRenderer = [[QCRenderer alloc] initWithOpenGLContext:context
                                           pixelFormat:format
                                                  file:path];
// Render the first 10 seconds of the composition with steps of 1/25s.
for(double t = 0.0; t <= 10.0; t += 1.0/25.0)
{
  [myRenderer renderAtTime:t arguments:nil];
  [context flushBuffer]; //Required on double-buffered contexts
}
// Clean up
```

```
[renderer release];
```

Tasks

Creating and Initializing a Renderer

- `initWithComposition:colorSpace:` (page 374)
  Creates a renderer object with a composition object and a color space.
- `initWithOpenGLContext:pixelFormat:file:` (page 375)
  Creates a renderer object with an `NSOpenGLContext` object and a composition file.
- `initWithCGLContext:pixelFormat:colorSpace:composition:` (page 374)
  Creates a renderer object with a `CGLContextObj` object, a pixel format, a color space, and a composition object.
- `initOffScreenWithSize:colorSpace:composition:` (page 373)
  Creates an offscreen renderer of a given size with the provided color space and composition object.

Rendering a Composition

- `renderAtTime:arguments:` (page 375)
  Renders a frame of a composition at the specified time.

Getting the Composition Object

- `composition` (page 372)
  Returns the composition object associated with the renderer.

Taking Snapshot Images

- `snapshotImage` (page 376)
  Returns an `NSImage` object of the current image in the OpenGL context associated with the renderer.
- `createSnapshotImageOfType:` (page 373)
  Returns the current image in the OpenGL context associated with the renderer, as an image object of the provided image type.

Instance Methods

composition

Returns the composition object associated with the renderer.

```
- (QCComposition*) composition
```

Return Value
The composition object.

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

Declared In
`QCRenderer.h`

createSnapshotImageOfType:

Returns the current image in the OpenGL context associated with the renderer, as an image object of the provided image type.

```
- (id) createSnapshotImageOfType:(NSString*)type
```

Parameters

type

A string that specifies any of the following image types: `NSBitmapImageRep`, `NSImage`, `CIImage`, `CGImage`, `CVOpenGLBuffer`, `CVPixelBuffer`.

Return Value
The snapshot image in the provided image type. You are responsible for releasing this object when you no longer need it.

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

Declared In
`QCRenderer.h`

initOffScreenWithSize:colorSpace:composition:

Creates an offscreen renderer of a given size with the provided color space and composition object.

```
- (id) initOffScreenWithSize:(NSSize)size colorSpace:(CGColorSpaceRef)colorSpace
    composition:(QCComposition*)composition
```

Parameters

size

The size of the offscreen renderer.

colorSpace

A Quartz color space object. This must be an RGB color space. Pass `NULL` to use the default RGB color space. For more information on Quartz color spaces, see *Quartz 2D Programming Guide*.

composition

A `QCComposition` object.

Return Value
The initialized `QCRenderer` object or `nil` if initialization is not successful.

Discussion
This method creates an internal OpenGL context and pixel buffer. Because offscreen rendering is performed on the GPU, the maximum rendering size is limited to the GPU capacity. On typical hardware, the limit is at least 2048 by 2048, but is often 4096 by 4096. The available VRAM affects performance.

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

Declared In
`QCRenderer.h`

initWithCGLContext:pixelFormat:colorSpace:composition:

Creates a renderer object with a `CGLContextObj` object, a pixel format, a color space, and a composition object.

```
- (id) initWithCGLContext:(CGLContextObj)context
    pixelFormat:(CGLPixelFormatObj)format colorSpace:(CGColorSpaceRef)colorSpace
    composition:(QCComposition*)composition;
```

Parameters

context
A `CGLContextObj` object. The object that you supply must have both a color and a depth buffer.

format
A `CGLPixelFormatObj` object.

colorSpace
A Quartz color space object. This must be an RGB color space. Pass `NULL` to use the default RGB color space. For more information on Quartz color spaces, see *Quartz 2D Programming Guide*.

composition
A `QCComposition` object.

Return Value
The initialized `QCRenderer` object or `nil` if initialization is not successful.

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

Declared In
`QCRenderer.h`

initWithComposition:colorSpace:

Creates a renderer object with a composition object and a color space.

```
- (id) initWithComposition:(QCComposition*)composition
    colorSpace:(CGColorSpaceRef)colorSpace;
```

Parameters

composition
A `QCComposition` object. The composition must not contain any consumer patches. That is, the composition can receive data, process it, and produce output values, but it cannot perform any rendering.

colorSpace
A Quartz color space object. This must be an RGB color space. Pass `NULL` to use the default RGB color space. The color space is used only for the images produced by the output image ports of the composition. For more information on Quartz color spaces, see *Quartz 2D Programming Guide*.

Return Value
The initialized `QCRenderer` object or `nil` if initialization is not successful.

Discussion
Note that `snapshotImage` (page 376) and `createSnapshotImageOfType:` (page 373) always returns `nil` on such `QCRenderer` instances.

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

Declared In
`QCRenderer.h`

initWithOpenGLContext:pixelFormat:file:

Creates a renderer object with an `NSOpenGLContext` object and a composition file.

```
- (id)initWithOpenGLContext:(NSOpenGLContext *)context
    pixelFormat:(NSOpenGLPixelFormat *)format file:(NSString *)path
```

Parameters

context
An `NSOpenGLContext` object. The object that you supply must have both a color and a depth buffer.

format
An `NSOpenGLPixelFormat` object.

path
A string that specifies the location of a composition(`.qtz`) file.

Return Value
An initialized `QCRenderer` object or `nil` if initialization is not successful.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`QCRenderer.h`

renderAtTime:arguments:

Renders a frame of a composition at the specified time.

```
- (BOOL)renderAtTime:(NSTimeInterval)time arguments:(NSDictionary *)arguments
```

Parameters

time
The time, in seconds, at which to render a composition frame. The time must be a positive value or zero.

arguments
An optional dictionary that can have any of the entries defined in "Rendering Arguments" (page 376).

Return Value
YES if successful.

Discussion
You need to call this method each time you want to render a frame of the composition.

All OpenGL states are preserved *except* the following:

- States defined by GL_CURRENT_BIT
- Textures on each unit and the environment mode
- Matrix mode

If you are using double buffers, keep in mind that the renderAtTime:arguments: method does not swap the front and back buffers of the OpenGL context. You must perform the swap yourself by calling the OpenGL command flushBuffer on the context associated with the renderer.

If you are interleaving OpenGL code with rendering of a composition, make sure that the OpenGL context is current. If you are using the NSOpenGLContext class, call the makeCurrentContext method prior to rendering. If you are using the CGL API, call the function CGLSetCurrentContext.

Availability
Available in Mac OS X v10.4 and later.

Declared In
QCRenderer.h

snapshotImage

Returns an NSImage object of the current image in the OpenGL context associated with the renderer.

```
- (NSImage*) snapshotImage
```

Return Value
The snapshot image.

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

Declared In
QCRenderer.h

Constants

Rendering Arguments

Arguments that you can pass to the renderAtTime:arguments: (page 375) method.

```
extern NSString* const QCRendererEventKey;
extern NSString* const QCRendererMouseLocationKey;
```

Constants

`QCRendererEventKey`

A key for a renderer event. The associated value is an `NSEvent` object.

Available in Mac OS X v10.4 and later.

Declared in `QCRenderer.h`.

`QCRendererMouseLocationKey`

A key for the mouse location. The associated value is an `NSPoint` object stored in an `NSValue` object. The mouse location is in normalized coordinates relative to the OpenGL context viewport (`[0,1]x[0,1]` with the origin `(0,0)` at the lower-left corner).

Available in Mac OS X v10.4 and later.

Declared in `QCRenderer.h`.

Declared In

`QCRenderer.h`

QCView Class Reference

Inherits from	NSView : NSResponder : NSObject
Conforms to	QCCompositionRenderer NSAnimatablePropertyContainer (NSView) NSCoding (NSResponder) NSObject (NSObject)
Framework	/System/Library/Frameworks/Quartz.framework/Frameworks/QuartzComposer.framework
Declared in	QuartzComposer/QCView.h
Availability	Available in Mac OS X v10.4 and later.
Companion guide	Quartz Composer Programming Guide

Overview

The `QCView` class is a custom `NSView` class that loads, plays, and controls Quartz Composer compositions. It is an autonomous view that is driven by an internal timer running on the main thread.

The view can be set to render a composition automatically when it is placed onscreen. The view stops rendering when it is placed offscreen. When not rendering, the view is filled with the current erase color. The rendered composition automatically synchronizes to the vertical retrace of the monitor.

When you archive a `QCView` object, it saves the composition that's loaded at the time the view is archived.

If you want to perform custom operations while a composition is rendering such as setting input parameters or drawing OpenGL content, you need to subclass `QCView` and implement the `renderAtTime:arguments:` (page 387) method.

Tasks

Performing Custom Operations During Rendering

`- renderAtTime:arguments:` (page 387)
Overrides to perform your custom operations prior to or after rendering a frame of a composition.

Loading a Composition

- `loadCompositionFromFile:` (page 384)
  Loads the composition file located at the specified path.
- `loadComposition:` (page 384)
  Loads a `QCComposition` object into the view.
- `loadedComposition` (page 385)
  Returns the composition loaded in the view.
- `unloadComposition` (page 394)
  Unloads the composition from the view.

Managing the Erase Color

- `erase` (page 382)
  Clears the view using the current erase color.
- `eraseColor` (page 383)
  Retrieves the current color used to erase the view.
- `setEraseColor:` (page 390)
  Sets the color used to erase the view.

Setting and Getting Event Masks

- `eventForwardingMask` (page 383)
  Retrieves the mask used to filter which types of events are forwarded from the view to the composition during rendering.
- `setEventForwardingMask:` (page 390)
  Sets the mask used to filter which types of events are forwarded from the view to the composition during rendering.

Setting and Getting the Maximum Frame Rate

- `maxRenderingFrameRate` (page 385)
  Returns the maximum frame rate for rendering.
- `setMaxRenderingFrameRate:` (page 391)
  Sets the maximum rendering frame rate.

Managing Rendering

- `startRendering` (page 393)
  Starts rendering the composition that is in the view.
- `isRendering` (page 384)
  Checks whether a composition is rendering in the view.

- `autostartsRendering` (page 381)
  Checks whether the view is set to start rendering automatically.
- `setAutostartsRendering:` (page 389)
  Sets whether the composition that is in the view starts rendering automatically when the view is put on the screen.
- `stopRendering` (page 393)
  Stops rendering the composition that is in the view.
- `pauseRendering` (page 386)
  Pauses rendering in the view.
- `isPausedRendering` (page 383)
  Returns whether or not the rendering in the view is paused.
- `resumeRendering` (page 389)
  Resumes rendering a paused composition.

Using Interface Builder

- `play:` (page 387)
  Plays or pauses a composition in a view.
- `start:` (page 392)
  Starts rendering a composition in a view.
- `stop:` (page 393)
  Stops rendering a composition in a view.

Taking Snapshot Images

- `snapshotImage` (page 392)
  Returns an `NSImage` object of the current image in the view.
- `createSnapshotImageOfType:` (page 382)
  Returns the current image in the view as an image object of the provided image type.

Working With OpenGL

- `openGLContext` (page 386)
  Returns the OpenGL context used by the view.
- `openGLPixelFormat` (page 386)
  Returns the OpenGL pixel format used by the view.

Instance Methods

autostartsRendering

Checks whether the view is set to start rendering automatically.

```
- (BOOL)autostartsRendering
```

Return Value
Returns `YES` if the view is set to start rendering automatically when the view is put on screen.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- setAutostartsRendering:` (page 389)

Declared In
`QCView.h`

createSnapshotImageOfType:

Returns the current image in the view as an image object of the provided image type.

```
- (id) createSnapshotImageOfType:(NSString*)type
```

Parameters
type

A string that specifies any of the following image types: `NSBitmapImageRep`, `NSImage`, `CIImage`, `CGImage`, `CVOpenGLBuffer`, `CVPixelBuffer`.

Return Value
The snapshot image in the provided image type. You are responsible for releasing this object when you no longer need it.

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

See Also
`- snapshotImage` (page 392)

Declared In
`QCView.h`

erase

Clears the view using the current erase color.

```
- (void)erase
```

Availability
Available in Mac OS X v10.4 and later.

See Also
`- eraseColor` (page 383)

Declared In
`QCView.h`

eraseColor

Retrieves the current color used to erase the view.

```
- (NSColor *)eraseColor
```

Return Value
The color object previously set using the `setEraseColor:` (page 390) method.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `erase` (page 382)

Declared In
`QCView.h`

eventForwardingMask

Retrieves the mask used to filter which types of events are forwarded from the view to the composition during rendering.

```
- (NSUInteger)eventForwardingMask
```

Return Value
The event filtering mask.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `setEventForwardingMask:` (page 390)

Declared In
`QCView.h`

isPausedRendering

Returns whether or not the rendering in the view is paused.

```
- (BOOL) isPausedRendering;
```

Return Value
`YES` if the rendering is paused; otherwise `NO`.

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

See Also
- `pauseRendering` (page 386)
- `resumeRendering` (page 389)

Declared In
QCView.h

isRendering

Checks whether a composition is rendering in the view.

```
- (BOOL)isRendering
```

Return Value
Returns YES if a composition is rendering in the view; NO otherwise.

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

Declared In
QCView.h

loadComposition:

Loads a QCComposition object into the view.

```
- (BOOL) loadComposition:(QCComposition*)composition
```

Parameters
composition
The QCComposition object to load.

Return Value
YES if successful; otherwise NO. If unsuccessful, any composition that's already loaded in the view remains loaded.

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

See Also
- loadCompositionFromFile: (page 384)
- unloadComposition (page 394)
- loadedComposition (page 385)

Declared In
QCView.h

loadCompositionFromFile:

Loads the composition file located at the specified path.

```
- (BOOL)loadCompositionFromFile:(NSString *)path
```

Parameters
path
A string that specifies the location of a Quartz Composer composition file.

Return Value
If unsuccessful, returns `NO`; any composition that's already loaded in the view remains loaded.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `- loadComposition:` (page 384)
- `- unloadComposition` (page 394)
- `- loadedComposition` (page 385)

Declared In
`QCView.h`

loadedComposition

Returns the composition loaded in the view.

```
- (QCComposition*) loadedComposition
```

Return Value
The composition loaded in the view; otherwise `nil`.

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

See Also
- `- loadCompositionFromFile:` (page 384)
- `- loadComposition:` (page 384)
- `- unloadComposition` (page 394)

Declared In
`QCView.h`

maxRenderingFrameRate

Returns the maximum frame rate for rendering.

```
- (float)maxRenderingFrameRate
```

Return Value
The maximum frame rate for rendering. A value of `0.0` specifies that there is no limit.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `- setMaxRenderingFrameRate:` (page 391)

Declared In
`QCView.h`

openGLContext

Returns the OpenGL context used by the view.

```
- (NSOpenGLContext*) openGLContext
```

Return Value
An `NSOpenGLContext` object.

Discussion
This context as a read-only object . Do not attempt to change any of its settings. If you subclass `QCView` so that you can perform custom OpenGL drawing, you'll need to use this method to retrieve the view's OpenGL context.

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

See Also
`- renderAtTime:arguments:` (page 387)

Declared In
`QCView.h`

openGLPixelFormat

Returns the OpenGL pixel format used by the view.

```
- (NSOpenGLPixelFormat*) openGLPixelFormat
```

Return Value
An `NSOpenGLPixelFormat` object.

Discussion
This pixel format as a read-only object. Do not attempt to change any of its settings.

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

Declared In
`QCView.h`

pauseRendering

Pauses rendering in the view.

```
- (void) pauseRendering
```

Discussion
You can nest calls to this method.

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

See Also
- `resumeRendering` (page 389)
- `isPausedRendering` (page 383)

Declared In
`QCView.h`

play:

Plays or pauses a composition in a view.

```
- (IBAction) play:(id)sender
```

Parameters

sender

The object (such as a button or menu item) sending the message to play the composition. You need to connect the object in the interface to the action.

Return Value
The message sent to the target.

Discussion
This method starts rendering a composition if it is not already rendering, pauses a composition that is rendering, or resumes rendering for a composition whose rendering is paused. The method is invoked when the user clicks a button or issues a command from some other user interface element, such as a menu.

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

See Also
- `stop:` (page 393)

Declared In
`QCView.h`

renderAtTime:arguments:

Overrides to perform your custom operations prior to or after rendering a frame of a composition.

```
- (BOOL) renderAtTime:(NSTimeInterval)time arguments:(NSDictionary*)arguments
```

Parameters

time

The rendering time, in seconds, of the composition frame.

arguments

An optional dictionary that can contain `QCRendererEventKey` or `QCRendererMouseLocationKey` and the associated values. (See *QCRenderer Class Reference* or more information.)

Return Value
`NO` if your custom rendering fails, otherwise, `YES`.

Discussion
Do not call this method directly. You override this method only for subclasses of the `QCView` class and only if you want to perform custom operations or OpenGL rendering before and/or after Quartz Composer renders a frame of the composition.

The most common reasons to override this method are to:

- synchronize communication with the composition. For example, you might want to set input parameters of the composition. By overriding this method, you can set parameters only when necessary and only at a specific time.
- underlay or overlay custom OpenGL rendering.

To synchronize communication between a composition and another part of the application, the implementation looks similar to the following:

```
- (BOOL) renderAtTime:(NSTimeInterval)time
            arguments:(NSDictionary*)arguments
{
  // Your code to computer the value of myParameterValue
  [self setValue:myParameterValue forInputKey:@"myInput"];

  BOOL success = [super renderAtTime:time arguments:arguments];

  id result = [self valueForOutputKey:@"myOutput"];
  //Your code to perform some operation on the result

  return success;
}
```

To perform OpenGL drawing in a `QCView` object, follow these guidelines:

- Use the OpenGL context of the `QCView` object to do drawing. You can retrieve the OpenGL context by calling `[self openGLContext]`. Note that this context won't necessarily be set as the current OpenGL context.
- Use CGL macros instead of managing the current OpenGL context yourself.

  OpenGL performs a global context and renderer lookup for each command it executes to ensure that all OpenGL commands are issued to the correct rendering context and renderer. There is significant overhead associated with these lookups that can measurably affect performance. CGL macros let you provide a local context variable and cache the current renderer in that variable. They are simple to use, taking only a few lines of code to set up.
- Save and restore all state changes except the ones that are part of `GL_CURRENT_BIT` (RGBA color, color index, normal vector, texture coordinates, and so forth).
- Check for OpenGL errors with `glGetError`.

Here's an example implementation of this method using OpenGL to draw an overlay:

```
#import <OpenGL/CGLMacro.h>  // Set up using macros

- (BOOL) renderAtTime:(NSTimeInterval)time
            arguments:(NSDictionary*)arguments
{
```

```
    BOOL success = [super renderAtTime:time arguments:arguments];

    // Use the OpenGL context of the view for drawing.
    CGLContextObj cgl_ctx = [[self openGLContext] CGLContextObj];

    // Save and set OpenGL states appropriately.
    glGetIntegerv(GL_MATRIX_MODE, &saveMode);
    glMatrixMode(GL_MODELVIEW);
    glPushMatrix();
    glRotatef(45.0, 0.0, 0.0, 1.0);

    // The code that performs OpenGL drawing goes here.
    //After drawing, restore original OpenGL states.
    glPopMatrix();
    glMatrixMode(saveMode);

    // Check for errors.
    glGetError();
    return success;
}
```

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

Declared In
QCView.h

resumeRendering

Resumes rendering a paused composition.

```
- (void) resumeRendering
```

Discussion
You can nest calls to this method.

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

See Also
- pauseRendering (page 386)
- isPausedRendering (page 383)

Declared In
QCView.h

setAutostartsRendering:

Sets whether the composition that is in the view starts rendering automatically when the view is put on the screen.

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

Parameters

flag

Pass YES to enable autostart mode; NO otherwise.

Availability
Available in Mac OS X v10.4 and later.

See Also
- autostartsRendering (page 381)

Declared In
QCView.h

setEraseColor:

Sets the color used to erase the view.

```
- (void)setEraseColor:(NSColor *)color
```

Parameters

color

A color object.

Availability
Available in Mac OS X v10.4 and later.

See Also
- erase (page 382)
- eraseColor (page 383)

Declared In
QCView.h

setEventForwardingMask:

Sets the mask used to filter which types of events are forwarded from the view to the composition during rendering.

```
- (void)setEventForwardingMask:(NSUInteger)mask
```

Parameters

mask

An event filtering mask. The mask can be a combination of any of the mask constants listed in Table 48-1 or the constant `NSAnyEventMask`.

Table 48-1 Events that can be forwarded to a composition

Event	Description
`NSLeftMouseDownMask`	The user pressed the left button.
`NSLeftMouseDraggedMask`	The user moved the mouse with the left button down.
`NSLeftMouseUpMask`	The user released the left button.
`NSRightMouseDownMask`	The user pressed the right button.
`NSRightMouseDraggedMask`	The user moved the mouse with the right button down.
`NSRightMouseUpMask`	The user released the right button.
`NSOtherMouseDownMask`	The user pressed the middle button, or some button other than the left or right button.
`NSOtherMouseDraggedMask`	The user moved the mouse with the middle button down, or some button other than the left or right button.
`NSOtherMouseUpMask`	The user released the middle button, or some button other than the left or right button.
`NSMouseMovedMask`	The user moved the mouse without holding down a mouse button.
`NSScrollWheelMask`	The user moved the mouse scroll wheel.
`NSKeyDownMask`	The user generated a character or characters by pressing a key.
`NSKeyUpMask`	The user released a key.
`NSFlagsChangedMask`	The user pressed or released a modifier key, or toggled the Caps Lock key.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- eventForwardingMask` (page 383)

Declared In
`QCView.h`

setMaxRenderingFrameRate:

Sets the maximum rendering frame rate.

```
- (void)setMaxRenderingFrameRate:(float)maxFPS
```

Parameters

maxFPS

The frame rate to set. Pass `0.0` to specify that there is no limit.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `maxRenderingFrameRate` (page 385)

Declared In
`QCView.h`

snapshotImage

Returns an `NSImage` object of the current image in the view.

```
- (NSImage*) snapshotImage
```

Return Value
The snapshot image.

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

See Also
- `createSnapshotImageOfType:` (page 382)

Declared In
`QCView.h`

start:

Starts rendering a composition in a view.

```
- (IBAction)start:(id)sender
```

Parameters

sender

The object (such as a button or menu item) sending the message to start rendering. You need to connect the object in the interface to the action.

Return Value
The message sent to the target.

Discussion
The method is invoked when the user clicks a button or issues a command from some other user interface element, such as a menu. It is equivalent to the `startRendering` (page 393) method.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `stop:` (page 393)

Declared In
QCView.h

startRendering

Starts rendering the composition that is in the view.

```
- (BOOL)startRendering
```

Return Value
Returns NO if the composition fails to start rendering; YES otherwise.

Availability
Available in Mac OS X v10.4 and later.

See Also
- stopRendering (page 393)

Declared In
QCView.h

stop:

Stops rendering a composition in a view.

```
- (IBAction)stop:(id)sender
```

Parameters

sender
The object (such as a button or menu item) sending the message to stop rendering. You need to connect the object in the interface to the action.

Return Value
The message sent to the target.

Discussion
The method is invoked when the user clicks a button or issues a command from some other user interface element, such as a menu. It is equivalent to the stopRendering (page 393) method.

Availability
Available in Mac OS X v10.4 and later.

See Also
- start: (page 392)

Declared In
QCView.h

stopRendering

Stops rendering the composition that is in the view.

```
- (void)stopRendering
```

Availability
Available in Mac OS X v10.4 and later.

See Also
- `startRendering` (page 393)

Declared In
`QCView.h`

unloadComposition

Unloads the composition from the view.

```
- (void) unloadComposition;
```

Discussion
If necessary, this method calls `stopRendering` (page 393) prior to unloading the composition.

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

See Also
- `loadCompositionFromFile:` (page 384)
- `loadComposition:` (page 384)
- `loadedComposition` (page 385)

Declared In
`QCView.h`

Notifications

QCViewDidStartRenderingNotification

Posted when the view starts rendering.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`QCView.h`

QCViewDidStopRenderingNotification

Posted when the view stops rendering.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`QCView.h`

Protocols

IKFilterCustomUIProvider Protocol Reference

Framework	System/Library/Frameworks/Quartz.framework/ImageKit.framework
Availability	Available in Mac OS X v10.5 and later.
Declared in	ImageKit/IKFilterUI.h

Overview

The `IKFilterCustomUIProvider` protocol is an addition to the `CIFilter` class that defines a method for providing a view for a filter. This protocol is implemented by any filter that provides its own user interface.

Tasks

Providing a Custom View

`- provideViewForUIConfiguration:excludedKeys:` (page 397)
Provides a custom view for a filter.

Instance Methods

provideViewForUIConfiguration:excludedKeys:

Provides a custom view for a filter.

```
-(IKFilterUIView*)provideViewForUIConfiguration:(NSDictionary*)inUIConfiguration
    excludedKeys:(NSArray*)inKeys
```

Parameters

inUIConfiguration
A dictionary that specifies the size of the controls. Provide the key `IKUISizeFlavor` and one of the following values: `IKUISizeMini`, `IKUISizeSmall`, or `IKUISizeRegular`. For more information on these constants, see *User Interface Options* in *CIFilter Image Kit Additions*.

inKeys
An array of the input keys for which you do *not* want to provide a user interface. Pass `nil` if you want all input keys to be represented in the user interface.

Return Value
An `IKFilterUIView` object or `nil` if the filter is unable to provide a view. If `nil`, the Image Kit framework will attempt to provide a user interface.

Discussion
This method overrides the method `viewForUIConfiguration:excludedKeys:` (page 20).

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

Declared In
`IKFilterUI.h`

IKImageBrowserDataSource Protocol Reference

(informal protocol)

Adopted by	IKImageBrowserView
Framework	/System/Library/Frameworks/Quartz.framework/ImageKit.framework
Declared in	ImageKit/IKImageBrowserView.h

Overview

The `IKImageBrowserDataSource` informal protocol declares the methods that an instance of the `IKImageBrowserView` class uses to access the contents of its data source object.

Tasks

Providing Information About Items (Required)

- `numberOfItemsInImageBrowser:` (page 403)
  Returns the number of records managed by the data source object.
- `imageBrowser:itemAtIndex:` (page 400)
  Returns an object for the item in an image browser view that corresponds to the specified index.

Supporting Item Editing (Optional)

- `imageBrowser:removeItemsAtIndexes:` (page 401)
  Signals that a remove operation should be applied to the specified items.
- `imageBrowser:moveItemsAtIndexes:toIndex:` (page 401)
  Signals that the specified items should be moved to the specified destination.
- `imageBrowser:writeItemsAtIndexes:toPasteboard:` (page 402)
  Signals that a drag should begin.

Providing Information About Groups (Optional)

- `numberOfGroupsInImageBrowser:` (page 402)
  Returns the number of groups in an image browser view.

Instance Methods

imageBrowser:groupAtIndex:

Returns the group at the specified index.

```
- (NSDictionary *) imageBrowser:(IKImageBrowserView *) aBrowser
    groupAtIndex:(NSUInteger) index;
```

Parameters

aBrowser

An image browser view.

index

The index of the group you want to retrieve.

Return Value

A dictionary that defines the group. The keys in this dictionary can be any of the following constants: `IKImageBrowserGroupStyle`, `IKImageBrowserGroupBackgroundColorKey`, `IKImageBrowserGroupTitleKey`, and `IKImageBrowserGroupRangeKey`. For more information on these constants, see *IKImageBrowserView Class Reference*.

Discussion

This method is optional.

Availability

Available in Mac OS X v10.5 and later.

Declared In

`IKImageBrowserView.h`

imageBrowser:itemAtIndex:

Returns an object for the item in an image browser view that corresponds to the specified index.

```
- (id) imageBrowser:(IKImageBrowserView *) aBrowser itemAtIndex:(NSUInteger)index;
```

Parameters

aBrowser

An image browser view.

index

The index of the item you want to retrieve.

Return Value

An `IKImageBrowserItem` object.

Discussion
Your data source must implement this method. The returned object must implement the required methods of the `IKImageBrowserItem` protocol.

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

Declared In
`IKImageBrowserView.h`

imageBrowser:moveItemsAtIndexes:toIndex:

Signals that the specified items should be moved to the specified destination.

```
- (BOOL) imageBrowser:(IKImageBrowserView *) aBrowser moveItemsAtIndexes: (NSIndexSet
    *)indexes toIndex:(NSUInteger)destinationIndex;
```

Parameters

aBrowser
An image browser view.

indexes
The indexes of the items that should be reordered.

destinationIndex
The starting index of the destination the items should be moved to.

Return Value
`YES` if successful; `NO` otherwise.

Discussion
This method is optional. It is invoked by the image browser view after Image Kit determines that a reordering operation should be applied. The data source should update itself by reordering its elements.

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

See Also
`- setAllowsReordering:` (page 48)

Declared In
`IKImageBrowserView.h`

imageBrowser:removeItemsAtIndexes:

Signals that a remove operation should be applied to the specified items.

```
- (void) imageBrowser:(IKImageBrowserView *) aBrowser
    removeItemsAtIndexes:(NSIndexSet *) indexes;
```

Parameters

aBrowser
An image browser view.

indexes
The indexes of the items that should be removed.

Discussion
This method is optional. It is invoked by the image browser after Image Kit determines that a remove operation should be applied. In response, the data source should update itself by removing the specified items.

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

Declared In
`IKImageBrowserView.h`

imageBrowser:writeItemsAtIndexes:toPasteboard:

Signals that a drag should begin.

```
- (NSUInteger) imageBrowser:(IKImageBrowserView *) aBrowser
    writeItemsAtIndexes:(NSIndexSet *) itemIndexes toPasteboard:(NSPasteboard
    *)pasteboard;
```

Parameters
aBrowser
An image browser view.
itemIndexes
The indexes of the items that should be dragged.
pasteboard
The pasteboard to copy the items to.

Return Value
The number of items written to the pasteboard.

Discussion
This method is optional. It is invoked after Image Kit determines that a drag should begin, but before the drag has been started.

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

Declared In
`IKImageBrowserView.h`

numberOfGroupsInImageBrowser:

Returns the number of groups in an image browser view.

```
- (NSUInteger) numberOfGroupsInImageBrowser:(IKImageBrowserView *) aBrowser;
```

Parameters
aBrowser
An image browser view.

Return Value
The number of groups.

Discussion
This method is optional.

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

Declared In
`IKImageBrowserView.h`

numberOfItemsInImageBrowser:

Returns the number of records managed by the data source object.

```
- (NSUInteger) numberOfItemsInImageBrowser:(IKImageBrowserView *) aBrowser;
```

Parameters

aBrowser
An image browser view.

Return Value
The number of records managed by the image browser view.

Discussion
Your data source must implement this method. An `IKImageView` object uses this method to determine how many cells it should create and display.

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

Declared In
`IKImageBrowserView.h`

IKImageBrowserDelegate Protocol Reference

(informal protocol)

Adopted by	IKImageBrowserView
Framework	System/Library/Frameworks/Quartz.framework/ImageKit.framework
Declared in	ImageKit/IKImageBrowserView.h

Overview

The `IKImageBrowserDelegate` is an informal protocol for the delegate of an `IKImageBrowserView` object. You can implement these methods to perform custom tasks when in response to events in the image browser view.

Tasks

Performing Custom Tasks in Response to User Events

- `imageBrowser:backgroundWasRightClickedWithEvent:` (page 405)
  Performs custom tasks when the user right-clicks the image browser view background.
- `imageBrowser:cellWasRightClickedAtIndex:withEvent:` (page 406)
  Performs custom tasks when the user right-clicks an item in the image browser view.
- `imageBrowser:cellWasDoubleClickedAtIndex:` (page 406)
  Performs custom tasks when the user double-clicks an item in the image browser view.
- `imageBrowserSelectionDidChange:` (page 407)
  Performs custom tasks when the selection changes.

Instance Methods

imageBrowser:backgroundWasRightClickedWithEvent:

Performs custom tasks when the user right-clicks the image browser view background.

```
- (void) imageBrowser:(IKImageBrowserView *) aBrowser
    backgroundWasRightClickedWithEvent:(NSEvent *) event;
```

Parameters

aBrowser

An image browser view.

event

The event that invoked the method.

Discussion

This method signals that the user either right-clicked the background or left-clicked it with the Alt key pressed. You can implement this method if you want to perform custom tasks at that time.

Availability

Available in Mac OS X v10.5 and later.

Declared In

`IKImageBrowserView.h`

imageBrowser:cellWasDoubleClickedAtIndex:

Performs custom tasks when the user double-clicks an item in the image browser view.

```
- (void) imageBrowser:(IKImageBrowserView *) aBrowser
    cellWasDoubleClickedAtIndex:(NSUInteger) index;
```

Parameters

aBrowser

An image browser view.

index

The index of the cell.

Discussion

This method signals that the user double-clicked an item in the image browser view. You can implement this method if you want to perform custom tasks at that time.

Availability

Available in Mac OS X v10.5 and later.

Declared In

`IKImageBrowserView.h`

imageBrowser:cellWasRightClickedAtIndex:withEvent:

Performs custom tasks when the user right-clicks an item in the image browser view.

```
- (void) imageBrowser:(IKImageBrowserView *) aBrowser
    cellWasRightClickedAtIndex:(NSUInteger) index withEvent:(NSEvent *) event;
```

Parameters

aBrowser

An image browser view.

index

The index of the cell.

event
The event that invoked the method.

Discussion
This method signals that the user either right-clicked an item in the browser or left-clicked the item with the Alt key pressed. You can implement this method if you want to perform custom tasks at that time.

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

Declared In
`IKImageBrowserView.h`

imageBrowserSelectionDidChange:

Performs custom tasks when the selection changes.

```
- (void) imageBrowserSelectionDidChange:(IKImageBrowserView *) aBrowser;
```

Parameters
aBrowser
An image browser view.

Discussion
This method signals that the user changes the selection in the image browser view. You can implement this method if you want to perform custom tasks at that time.

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

Declared In
`IKImageBrowserView.h`

IKImageBrowserItem Protocol Reference

(informal protocol)

Framework	System/Library/Frameworks/Quartz.framework/ImageKit.framework
Declared in	ImageKit/IKImageBrowserView.h

Overview

The `IKImageBrowserItem` informal protocol declares the methods that an instance of the `IKImageBrowserView` class uses to access the contents of its data source for a given item. Some of the methods in this protocol are needed frequently, so you should implement them efficiently.

Tasks

Providing Required Information for an Image

- `imageUID` (page 411)
  Returns a unique string that identifies the data source item.
- `imageRepresentationType` (page 410)
  Returns the representation type of the image to display.
- `imageRepresentation` (page 410)
  Returns the image to display.

Providing Optional Information for an Image

- `imageVersion` (page 411)
  Returns the version of the item.
- `imageTitle` (page 411)
  Returns the display title of the image.
- `imageSubtitle` (page 410)
  Returns the display subtitle of the image.
- `isSelectable` (page 412)
  Returns whether this item is selectable.

Instance Methods

imageRepresentation

Returns the image to display.

```
- (id) imageRepresentation;
```

Return Value
The image to display; can return `nil` if the item has no image to display.

Discussion
Your data source must implement this method. This method is called frequently, so the receiver should cache the returned instance.

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

Declared In
`IKImageBrowserView.h`

imageRepresentationType

Returns the representation type of the image to display.

```
- (NSString *) imageRepresentationType;
```

Return Value
A string that specifies the image representation type. The string can be any of the constants defined in "Image Representation Types" (page 412).

Discussion
Your data source must implement this method.

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

Declared In
`IKImageBrowserView.h`

imageSubtitle

Returns the display subtitle of the image.

```
- (NSString *) imageSubtitle
```

Return Value
The display subtitle of the image.

Discussion
This method is optional.

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

Declared In
`IKImageBrowserView.h`

imageTitle

Returns the display title of the image.

```
- (NSString *) imageTitle;
```

Return Value
The display title of the image.

Discussion
This method is optional.

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

Declared In
`IKImageBrowserView.h`

imageUID

Returns a unique string that identifies the data source item.

```
- (NSString *)  imageUID;
```

Return Value
The string that identifies the data source item

Discussion
Your data source must implement this method. The image browser view uses this identifier to associate the data source item and its cache.

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

Declared In
`IKImageBrowserView.h`

imageVersion

Returns the version of the item.

```
- (NSUInteger) imageVersion;
```

Return Value
The version of the item.

Discussion
This method is optional. The receiver can return a new version to let the image browser knows that it should not use its cache for the item.

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

Declared In
`IKImageBrowserView.h`

isSelectable

Returns whether this item is selectable.

```
- (BOOL) isSelectable;
```

Return Value
`YES` if the item is selectable; `NO` otherwise.

Discussion
This method is optional. You can prevent selection of this item by returning `NO`.

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

Declared In
`IKImageBrowserView.h`

Constants

Image Representation Types

Representation types for images.

```
NSString * const IKImageBrowserPathRepresentationType;
NSString * const IKImageBrowserNSURLRepresentationType;
NSString * const IKImageBrowserNSImageRepresentationType;
NSString * const IKImageBrowserCGImageRepresentationType;
NSString * const IKImageBrowserCGImageSourceRepresentationType;
NSString * const IKImageBrowserNSDataRepresentationType;
NSString * const IKImageBrowserNSBitmapImageRepresentationType;
NSString * const IKImageBrowserQTMovieRepresentationType;
NSString * const IKImageBrowserQTMoviePathRepresentationType;
NSString * const IKImageBrowserQCCompositionRepresentationType;
NSString * const IKImageBrowserQCCompositionPathRepresentationType;
NSString * const IKImageBrowserQuickLookPathRepresentationType;
NSString * const IKImageBrowserIconRefPathRepresentationType;
NSString * const IKImageBrowserIconRefRepresentationType;
```

Constants

`IKImageBrowserPathRepresentationType`
A path representation (`NSString`).
Available in Mac OS X v10.5 and later.
Declared in `IKImageBrowserView.h`.

`IKImageBrowserNSURLRepresentationType`
An `NSURL`object.
Available in Mac OS X v10.5 and later.
Declared in `IKImageBrowserView.h`.

`IKImageBrowserNSImageRepresentationType`
An `NSImage` object.
Available in Mac OS X v10.5 and later.
Declared in `IKImageBrowserView.h`.

`IKImageBrowserCGImageRepresentationType`
A `CGImageRef` object.
Available in Mac OS X v10.5 and later.
Declared in `IKImageBrowserView.h`.

`IKImageBrowserCGImageSourceRepresentationType`
A `CGImageSourceRef` object.
Available in Mac OS X v10.5 and later.
Declared in `IKImageBrowserView.h`.

`IKImageBrowserNSDataRepresentationType`
An `NSData` object.
Available in Mac OS X v10.5 and later.
Declared in `IKImageBrowserView.h`.

`IKImageBrowserNSBitmapImageRepresentationType`
An `NSBitmapImageRep` object.
Available in Mac OS X v10.5 and later.
Declared in `IKImageBrowserView.h`.

`IKImageBrowserQTMovieRepresentationType`
A `QTMovie` object.

Available in Mac OS X v10.5 and later.

Declared in `IKImageBrowserView.h`.

`IKImageBrowserQTMoviePathRepresentationType`
A path (NSString) or URL (NSURL) to a QuickTime movie.

Available in Mac OS X v10.5 and later.

Declared in `IKImageBrowserView.h`.

`IKImageBrowserQCCompositionRepresentationType`
A `QCComposition` object.

Available in Mac OS X v10.5 and later.

Declared in `IKImageBrowserView.h`.

`IKImageBrowserQCCompositionPathRepresentationType`
A path (NSString) or URL (NSURL) to a Quartz Composer composition.

Available in Mac OS X v10.5 and later.

Declared in `IKImageBrowserView.h`.

`IKImageBrowserQuickLookPathRepresentationType`
A path (NSString) or URL (NSURL) to load data using QuickLook.

Available in Mac OS X v10.5 and later.

Declared in `IKImageBrowserView.h`.

`IKImageBrowserIconRefPathRepresentationType`
A path to an icon.

Available in Mac OS X v10.5 and later.

Declared in `IKImageBrowserView.h`.

`IKImageBrowserIconRefRepresentationType`
An icon.

Available in Mac OS X v10.5 and later.

Declared in `IKImageBrowserView.h`.

Declared In
`IKImageBrowserView.h`

IKImageEditPanelDataSource Protocol Reference

Framework	System/Library/Frameworks/Quartz.framework/ImageKit.framework
Availability	Available in Mac OS X v10.5 and later.
Declared in	ImageKit/IKImageEditPanel.h

Overview

The `IKImageEditPanelDataSource` informal protocol describes the methods that an `IKImageEditPanel` object uses to access the contents of its data source object.

Tasks

Getting and Setting Image Properties

- `imageProperties` (page 416)
  Returns a dictionary of the image properties associated with the image in the image edit panel.
- `setImage:imageProperties:` (page 416)
  Sets an image with the specified properties.

Getting Images From the Data Source

- `image` (page 415)
  Returns an image.
- `thumbnailWithMaximumSize:` (page 417)
  Returns a thumbnail image whose size is no larger than the specified size.

Instance Methods

image

Returns an image.

```
- (CGImageRef)image;
```

Return Value
An image.

Discussion
Your data source must implement this method.

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

Declared In
`IKImageEditPanel.h`

imageProperties

Returns a dictionary of the image properties associated with the image in the image edit panel.

```
- (NSDictionary*)imageProperties;
```

Return Value
A dictionary that contains the properties of the image.

Discussion
This method is optional.

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

See Also
`- setImage:imageProperties` (page 416)

Declared In
`IKImageEditPanel.h`

setImage:imageProperties:

Sets an image with the specified properties.

```
- (void)setImage: (CGImageRef)image imageProperties: (NSDictionary*)metaData;
```

Discussion
Your data source must implement this method.

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

See Also
`- imageProperties` (page 416)

Declared In
`IKImageEditPanel.h`

thumbnailWithMaximumSize:

Returns a thumbnail image whose size is no larger than the specified size.

```
- (CGImageRef)thumbnailWithMaximumSize: (NSSize)size;
```

Return Value
An image.

Discussion
This method is optional.

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

Declared In
`IKImageEditPanel.h`

IKSlideshowDataSource Protocol Reference

Adopted by	IKSlideshow
Framework	System/Library/Frameworks/Quartz.framework/ImageKit.framework
Availability	Available in Mac OS X v10.5 and later.
Declared in	ImageKit/IKSllideShow.h

Overview

The `IKSlideshowDataSource` protocol describes the methods that an `IKSlideshow` object uses to access the contents of its data source object.

Important: Slide show data source methods may be called on secondary threads. When you implement these methods, you must ensure that they are safe to run on threads other than the main thread.

Tasks

Providing Slideshow Information

- `numberOfSlideshowItems` (page 421)
  Returns the number of items in a slideshow.
- `slideshowItemAtIndex:` (page 422)
  Returns the item for a given index
- `nameOfSlideshowItemAtIndex:` (page 420)
  Returns the display name for item at the specified index.
- `canExportSlideshowItemAtIndex:toApplication:` (page 420)
  Reports whether the export button should be enabled for a a slideshow item.

Performing Custom Tasks

- `slideshowWillStart` (page 422)
  Performs custom tasks when the slideshow is about to start.

- `slideshowDidStop` (page 421)
  Performs custom tasks when the slideshow stops.
- `slideshowDidChangeCurrentIndex:` (page 421)
  Performs custom tasks when the slideshow changes to the item at the specified index.

Instance Methods

canExportSlideshowItemAtIndex:toApplication:

Reports whether the export button should be enabled for a a slideshow item.

```
- (BOOL)canExportSlideshowItemAtIndex: (NSUInteger)index toApplication: (NSString
    *)applicationBundleIdentifier;
```

Return Value
`YES` if the export button should be enabled for an item; otherwise `NO`.

Discussion
This method is optional.

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

Declared In
`IKSlideshow.h`

nameOfSlideshowItemAtIndex:

Returns the display name for item at the specified index.

```
- (NSString*)nameOfSlideshowItemAtIndex: (NSUInteger)index;
```

Parameters
index
The index for a slideshow item.

Return Value
The display name. For the best user experience, you should provide the localized name, because this string appears in the user interface.

Discussion
This method is optional.

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

Declared In
`IKSlideshow.h`

numberOfSlideshowItems

Returns the number of items in a slideshow.

```
- (NSUInteger)numberOfSlideshowItems;
```

Return Value
The number of items in the slideshow.

Discussion
Your data source must implement this method.

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

Declared In
`IKSlideshow.h`

slideshowDidChangeCurrentIndex:

Performs custom tasks when the slideshow changes to the item at the specified index.

```
- (void)slideshowDidChangeCurrentIndex: (NSUInteger)newIndex;
```

Parameters
newIndex
The index of the current item.

Discussion
This method is optional. Image Kit invokes this method when the slideshow changes to the specified item. Implement this method to perform custom tasks at that time.

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

Declared In
`IKSlideshow.h`

slideshowDidStop

Performs custom tasks when the slideshow stops.

```
- (void)slideshowDidStop;
```

Discussion
This method is optional. Image Kit invokes this method when the slideshow stops. Implement this method to perform custom tasks at that time.

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

See Also
`- slideshowWillStart` (page 422)

Declared In
IKSlideshow.h

slideshowItemAtIndex:

Returns the item for a given index

```
- (id)slideshowItemAtIndex: (NSUInteger)index;
```

Parameters

index

An index of an item in the slideshow.

Return Value
The object that corresponds to the item at the specified index. The item can be any of the following objects: `NSImage`, `NSString` (to specify a path name), `NSURL`, `NSFileWrapper`, `CGImageRef`, or `PDFPage`.

Discussion
Your data source must implement this method.

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

Declared In
IKSlideshow.h

slideshowWillStart

Performs custom tasks when the slideshow is about to start.

```
- (void)slideshowWillStart;
```

Discussion
This method is optional. Image Kit invokes this method when the slideshow is about to start. Implement this method to perform custom tasks at that time.

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

See Also
- slideshowDidStop (page 421)

Declared In
IKSlideshow.h

QCCompositionParameterViewDelegate Protocol Reference

(informal protocol)

Framework	/System/Library/Frameworks/Quartz.framework/Frameworks/QuartzComposer.framework
Declared in	QuartzComposer/QCCompositionParameterView.h
Companion guide	Quartz Composer Programming Guide

Overview

The `QCCompositionParameterViewDelegate` informal protocol allows your application it define which parameters should be visible in a `QCCompositionParameterView` object.

Tasks

Responding to Composition Selections

`- compositionParameterView:shouldDisplayParameterWithKey:attributes:` (page 423)
Allows you to define which composition parameters are visible in the user interface when the composition parameter view refreshes.

Instance Methods

compositionParameterView:shouldDisplayParameterWithKey:attributes:

Allows you to define which composition parameters are visible in the user interface when the composition parameter view refreshes.

```
- (BOOL) compositionParameterView:(QCCompositionParameterView *)parameterView
    shouldDisplayParameterWithKey:(NSString *)portKey attributes:(NSDictionary
    *)portAttributes;
```

Parameters

parameterView
The composition parameter view in which the selection changed.

portKey
A key for one of the composition parameters, which is provided to you by the Quartz Composer engine.

portAttributes
A dictionary of the attributes that you want to display in the user interface.

Return Value
`YES` if port attributes should be displayed; `NO` otherwise.

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

Declared In
`QCCompositionParameterView.h`

QCCompositionPickerViewDelegate Protocol Reference

(informal protocol)

Framework	/System/Library/Frameworks/Quartz.framework/Frameworks/QuartzComposer.framework
Declared in	QuartzComposer/QCCompositionPickerView.h
Companion guide	Quartz Composer Programming Guide

Overview

The `QCCompositionPickerViewDelegate` informal protocol defines methods that allow your application to respond to changes in a composition picker view (a `QCCompositionPickerView` object).

Tasks

Responding to Composition Selections

`- compositionPickerView:didSelectComposition:` (page 425)
Performs custom tasks when the selected composition in the composition picker view changes.

Responding to Animation State Changes

`- compositionPickerViewDidStartAnimating:` (page 426)
Performs custom tasks when the composition picker view starts animating a composition.

`- compositionPickerViewWillStopAnimating:` (page 426)
Performs custom tasks when the composition picker view stops animating a composition.

Instance Methods

compositionPickerView:didSelectComposition:

Performs custom tasks when the selected composition in the composition picker view changes.

```
- (void) compositionPickerView:(QCCompositionPickerView*)pickerView
    didSelectComposition:(QCComposition*)composition
```

Parameters

pickerView

The composition picker view in which the selection changed.

composition

The selected composition or `nil` if the previously selected composition is no longer selected.

Discussion

Quartz Composer invokes this method when the selected composition in the composition picker view changes. Implement this method if you want to perform custom tasks at that time.

Availability

Available in Mac OS X v10.5 and later.

Declared In

`QCCompositionPickerView.h`

compositionPickerViewDidStartAnimating:

Performs custom tasks when the composition picker view starts animating a composition.

```
- (void) compositionPickerViewDidStartAnimating:(QCCompositionPickerView*)pickerView
```

Parameters

pickerView

The composition picker view in which the composition started animating.

Discussion

Quartz Composer invokes this method when the composition picker view starts animating a composition. Implement this method if you want to perform custom tasks at that time.

Availability

Available in Mac OS X v10.5 and later.

Declared In

`QCCompositionPickerView.h`

compositionPickerViewWillStopAnimating:

Performs custom tasks when the composition picker view stops animating a composition.

```
 (void) compositionPickerViewWillStopAnimating:(QCCompositionPickerView*)pickerView
```

Parameters

pickerView

The composition picker view in which the composition stopped animating.

Discussion

Quartz Composer invokes this method whenever the composition picker view stops animating a composition. Implement this method if you want to perform custom tasks at that time.

Availability

Available in Mac OS X v10.5 and later.

Declared In
`QCCompositionPickerView.h`

QCCompositionRenderer Protocol Reference

Adopted by	QCRenderer QCView QCCompositionLayer
Framework	/System/Library/Frameworks/Quartz.framework/Frameworks/QuartzComposer.framework
Declared in	QuartzComposer/QCRenderer.h
Availability	Available in Mac OS X v10.5 and later.

Overview

The `QCRenderer` protocol defines the methods used to pass data to the input ports or retrieve data from the output ports of the root patch of a Quartz Composer composition. This protocol is adopted by the `QCRenderer`, `QCView`, and `QCCompositionLayer` classes.

Tasks

Passing and Retrieving Values From a Composition

- `setValue:forInputKey:` (page 432)
  Sets the value for an input port of a composition.
- `valueForInputKey:` (page 433)
  Returns the value for an input port of a composition.
- `valueForOutputKey:` (page 434)
  Returns the value for an output port of a composition.
- `valueForOutputKey:ofType:` (page 434)
  Returns the current value on an output port (identified by its key) of the root patch of the composition.

Getting Input and Output Keys

- `inputKeys` (page 431)
  Returns an array that contains the keys that identify the input ports of the root patch of the composition.

- `outputKeys` (page 431)
  Returns an array that contains the keys that identify the output ports of the root patch of the composition.

Getting Attributes

- `attributes` (page 430)
  Returns the attributes of the composition associated with the renderer.

Storing Arbitrary Information

- `userInfo` (page 433)
  Returns a mutable dictionary for storing arbitrary information.

Saving and Restoring Input Values

- `propertyListFromInputValues` (page 431)
  Returns a property list object that represents the current values for all the input keys of the composition.
- `setInputValuesWithPropertyList:` (page 432)
  Sets the values for the input keys of the composition from a previously saved property list.

Instance Methods

attributes

Returns the attributes of the composition associated with the renderer.

```
- (NSDictionary *)attributes
```

Return Value
A dictionary that contains the attributes that describe the composition, including the input and output ports of the root patch.

Discussion
The dictionary can define any of the attributes that are specified by the composition attribute keys. See `QCCompositionAttributeNameKey`, `QCCompositionAttributeDescriptionKey`, and `QCCompositionAttributeCopyrightKey`.

The dictionary can also contain dictionaries that correspond to the keys that identify the input and output ports of the root patch of the composition. See `QCPortAttributeTypeKey`, `QCPortAttributeNameKey`, `QCPortAttributeMinimumValueKey`, `QCPortAttributeMaximumValueKey`, and `QCPortAttributeMenuItemsKey` (page 363).

Availability
Available in Mac OS X v10.4 and later.

See Also
- `inputKeys` (page 431)
- `outputKeys` (page 431)

Declared In
`QCRenderer.h`

inputKeys

Returns an array that contains the keys that identify the input ports of the root patch of the composition.

```
- (NSArray *)inputKeys
```

Return Value
An array of keys associated with input ports.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `outputKeys` (page 431)

Declared In
`QCRenderer.h`

outputKeys

Returns an array that contains the keys that identify the output ports of the root patch of the composition.

```
- (NSArray *)outputKeys
```

Return Value
An array of keys associated with input ports.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `inputKeys` (page 431)

Declared In
`QCRenderer.h`

propertyListFromInputValues

Returns a property list object that represents the current values for all the input keys of the composition.

```
- (id) propertyListFromInputValues
```

Return Value
A property list object.

Discussion
This is a convenience method that allows you to easily save the set of input values on a composition. Typically, you store the set of values in application preferences.

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

See Also
`setInputValuesWithPropertyList:` (page 432)

Declared In
`QCRenderer.h`

setInputValuesWithPropertyList:

Sets the values for the input keys of the composition from a previously saved property list.

```
- (void) setInputValuesWithPropertyList:(id)plist
```

Discussion
This is a convenience method that allows you to restore the set of input values that you obtained previously by calling the method `propertyListFromInputValues` (page 431). If the property list object does not define a value for an input key, or if the value is not of the proper type, Quartz Composer does not set a value for the corresponding input port.

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

Declared In
`QCRenderer.h`

setValue:forInputKey:

Sets the value for an input port of a composition.

```
- (BOOL)setValue:(id)value forInputKey:(NSString *)key
```

Parameters

value

The value to set for the input port. The input port must be at the root patch of the composition. The data type of the `value` argument must match the input port. See `QCPortAttributeTypeKey` (page 362) for the data types accepted by a particular port type.

key

The key associated with the input port of the composition. This method throws an exception if `key` is invalid.

Return Value
Returns `NO` if it cannot set the value.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `valueForInputKey:` (page 433)
- `valueForOutputKey:` (page 434)

Declared In
`QCRenderer.h`

userInfo

Returns a mutable dictionary for storing arbitrary information.

```
- (NSMutableDictionary*) userInfo
```

Return Value
A mutable dictionary.

Discussion
The `userInfo` dictionary is shared—there is one per Quartz Composer context. In fact, it is the same dictionary as the one available for the plug-in execution context for instances of the `QCPlugIn` class.

When you add information to the dictionary, make sure that you use unique keys, such as `"com.myCompany.foo"`.

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

Declared In
`QCRenderer.h`

valueForInputKey:

Returns the value for an input port of a composition.

```
- (id)valueForInputKey:(NSString *)key
```

Parameters

key

The key associated with an input port for the root patch of a composition. This method throws an exception if `key` is invalid.

Return Value
The value. The data type of returned value depends on the type of the input port. See `QCPortAttributeTypeKey` (page 362) for more information.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `setValue:forInputKey:` (page 432)
- `valueForOutputKey:` (page 434)

Declared In
`QCRenderer.h`

valueForOutputKey:

Returns the value for an output port of a composition.

```
- (id)valueForOutputKey:(NSString *)key
```

Parameters

key

The key associated with an output port for the root patch of a composition. This method throws an exception if `key` is invalid.

Return Value

The value. The data type of returned value depends on the type of the output port. See `QCPortAttributeTypeKey` (page 362) for more information.

Availability

Available in Mac OS X v10.4 and later.

See Also

`- setValue:forInputKey:` (page 432)

`- valueForInputKey:` (page 433)

Declared In

`QCRenderer.h`

valueForOutputKey:ofType:

Returns the current value on an output port (identified by its key) of the root patch of the composition.

```
- (id) valueForOutputKey:(NSString*)key ofType:(NSString*)type
```

Parameters

key

The key associated with an output port for the root patch of a composition. This method throws an exception if `key` is invalid.

type

A string that specifies the class.

Return Value

The value.

Discussion

The value type depends on the type of the port type, as shown in the following table

Port type	Value type
Boolean, Index, or Number	`NSNumber` or any object that responds to the methods `integerValue`, `floatValue`, or `doubleValue`
String	`NSString` or any object that responds to the methods`stringValue` or `description`
Color	`NSColor`, `CIColor`, or `CGColor` object

Port type	Value type
Image	`NSImage`, `NSBitmapImageRep`, `CGImage` object, `CIImage`, `CVPixelBuffer` object, `CVOpenGLBuffer` object, `CVOpenGLTexture` object, or an opaque `QCImage` (that is, an optimized abstract image object only to be used with `setValue: forInputKey:` of another `<QCCompositionRenderer>`)
Structure	`NSArray` or `NSDictionary`

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

See Also
- `- setValue:forInputKey:` (page 432)
- `- valueForInputKey:` (page 433)

Declared In
`QCRenderer.h`

QCPlugInContext Protocol Reference

Framework	/System/Library/Frameworks/Quartz.framework/Frameworks/QuartzComposer.framework
Declared in	QuartzComposer/QCPlugIn.h
Availability	Available in Mac OS X v10.5 and later.

Overview

The `QCPlugInContext` protocol defines methods that you use only from within the execution method (`execute:atTime:withArguments:` (page 357)) of a `QCPlugIn` object.

Tasks

Getting the OpenGL Context

- `CGLContextObj` (page 438)
  Returns the destination CGL context to use for OpenGL rendering from within the execution method.

Logging Messages

- `logMessage:` (page 439)
  Writes a message to the Quartz Composer log.

Getting Execution Context Information

- `userInfo` (page 442)
  Returns a mutable dictionary that contains information that can be shared between all instances of the `QCPlugIn` subclass, running in the same Quartz Composer context.
- `bounds` (page 438)
  Returns the bounds of the rendering context.
- `colorSpace` (page 439)
  Returns the color space used by the rendering context.

Getting an Image Provider

- outputImageProviderFromBufferWithPixelFormat:pixelsWide:pixelsHigh:baseAddress:bytesPerRow:releaseCallback:releaseContext:colorSpace:shouldColorMatch: (page 440)

  Returns an image provider from a single memory buffer.

- outputImageProviderFromTextureWithPixelFormat:pixelsWide:pixelsHigh:name:flipped:releaseCallback:releaseContext:colorSpace:shouldColorMatch: (page 441)

  Returns an image provider from an OpenGL texture.

Instance Methods

bounds

Returns the bounds of the rendering context.

```
- (NSRect) bounds
```

Return Value
The bounds of the rendering context expressed in Quartz Composer units.

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

Declared In
`QCPlugIn.h`

CGLContextObj

Returns the destination CGL context to use for OpenGL rendering from within the execution method.

```
- (CGLContextObj) CGLContextObj
```

Return Value
The destination CGL context.

Discussion
To send commands to the OpenGL context:

- Use CGL macros instead of changing the current OpenGL context.
- Save and restore all OpenGL states except those defines by `GL_CURRENT_BIT` (vertex position, color, texture, and so on)

The following code shows how you'd use the method `CGLContextObj`:

```
// Set up using CGL macros.
#import <OpenGL/CGLMacro.h>

- (BOOL) execute:(id<QCPlugInContext>)context
              atTime:(NSTimeInterval)time
           withArguments:(NSDictionary *)arguments
```

```
{
    // Set the CGL context to a local variable.
    CGLContextObj cgl_ctx = [context CGLContextObj];
    if(cgl_ctx == NULL)
    return NO;

    // Save and set OpenGL states.
    // Put your OpenGL code here.
    // Restore the OpenGL states.
    return YES;
}
```

You can retrieve the corresponding OpenGL pixel format by calling the function `CGLGetPixelFormat`.

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

Declared In
`QCPlugIn.h`

colorSpace

Returns the color space used by the rendering context.

```
- (CGColorSpaceRef) colorSpace
```

Return Value
An RGB color space; `NULL` if the custom patch execution mode is not consumer.

Discussion
If the method returns a color space, it must be an RGB color space.

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

Declared In
`QCPlugIn.h`

logMessage:

Writes a message to the Quartz Composer log.

```
- (void) logMessage:(NSString*)format, ...
```

Parameters

format
The string to write to the log. The default location for the log is the standard output.

Discussion
This method is an alternative to using the functions `NSLog` or `printf`.

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

Declared In
`QCPlugIn.h`

outputImageProviderFromBufferWithPixelFormat:pixelsWide:pixelsHigh:baseAddress:bytesPerRow:releaseCallback:releaseContext:colorSpace:shouldColorMatch:

Returns an image provider from a single memory buffer.

```
- (id) outputImageProviderFromBufferWithPixelFormat:(NSString*)format
    pixelsWide:(NSUInteger)width pixelsHigh:(NSUInteger)height baseAddress:(const
     void*)baseAddress bytesPerRow:(NSUInteger)rowBytes
    releaseCallback:(QCPlugInBufferReleaseCallback)callback
    releaseContext:(void*)context colorSpace:(CGColorSpaceRef)colorSpace
    shouldColorMatch:(BOOL)colorMatch
```

Parameters

format
The pixel format of the memory buffer. This must be compatible with the color space.

width
The width, in bytes, of the memory buffer.

height
The height, in bytes, of the memory buffer.

baseAddress
The base address of the memory buffer, which must be multiple of 16.

rowBytes
The number of bytes per row of the memory buffer, which must be multiple of 16.

callback
The release callback. Your callback must use this type definition:

```
typedef void (*QCPlugInBufferReleaseCallback)(const void* address, void* context);
```

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

```
void MyQCPlugInBufferReleaseCallback (const void address,
                void * context);
```

Quartz Composer invokes your callback when the memory buffer is no longer needed. The callback can be called from any thread at any time

context
The context to pass to the release callback.

colorSpace
The color space of the memory buffer. This must be compatible with the pixel format.

colorMatch
A Boolean that specifies whether Quartz Composer should color match the image. Pass `NO` if the image is a mask or gradient or should not be color matched for some other reason. Otherwise, pass `YES`.

Return Value
An image provider.

Discussion
You must not modify the image until the release callback is invoked.

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

Declared In
`QCPlugIn.h`

outputImageProviderFromTextureWithPixelFormat:pixelsWide:pixelsHigh:name:flipped:releaseCallback:releaseContext:colorSpace:shouldColorMatch:

Returns an image provider from an OpenGL texture.

```
- (id) outputImageProviderFromTextureWithPixelFormat:(NSString*)format
    pixelsWide:(NSUInteger)width pixelsHigh:(NSUInteger)height name:(GLuint)name
    flipped:(BOOL)flipped releaseCallback:(QCPlugInTextureReleaseCallback)callback
     releaseContext:(void*)context colorSpace:(CGColorSpaceRef)colorSpace
    shouldColorMatch:(BOOL)colorMatch;
```

Parameters

format
The pixel format of the texture. This must be compatible with the color space.

width
The width, in bytes, of the texture.

height
The height, in bytes, of the texture.

name
An OpenGL texture of type `GL_TEXTURE_RECTANGLE_EXT` that is valid on the Quartz Composer OpenGL context. Note that textures do not have a retain and release mechanism. This means that your application must make sure that the texture exists for the life cycle of the image provider.

flipped
`YES` to have Quartz Composer flip the contents of the texture vertically.

callback
The release callback. Your callback must use this type definition:

```
typedef void (*QCPlugInTextureReleaseCallback)(CGLContextObj cgl_ctx, GLuint
name, void* context);
```

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

```
void MyQCPlugInTextureReleaseCallback (CGLContextObj cgl_ctx,
              GLuint name,
              void* context);
```

Quartz Composer invokes your callback when the memory buffer is no longer needed. The callback can be called from any thread at any time

context
The context to pass to the release callback.

colorSpace
The color space of the texture. This must be compatible with the pixel format.

colorMatch

A Boolean that specifies whether Quartz Composer should color match the texture. Pass `NO` if the texture is a mask or gradient or should not be color matched for some other reason. Otherwise, pass `YES`.

Return Value
An image provider.

Discussion
You must not modify the texture until the release callback is invoked.

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

Declared In
`QCPlugIn.h`

userInfo

Returns a mutable dictionary that contains information that can be shared between all instances of the `QCPlugIn` subclass, running in the same Quartz Composer context.

```
- (NSMutableDictionary*) userInfo
```

Return Value
A mutable dictionary.

Discussion
When you add information to the dictionary, make sure that you use unique keys, such as `com.myCompany.foo`. You can use this dictionary to cache data that you want to share.

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

Declared In
`QCPlugIn.h`

QCPlugInInputImageSource Protocol Reference

Framework	/System/Library/Frameworks/Quartz.framework/Frameworks/QuartzComposer.framework
Declared in	QuartzComposer/QCPlugIn.h
Availability	Available in Mac OS X v10.5 and later.

Overview

The `QCPlugInInputImageSource` protocol eliminates the need to use explicit image types for the image input ports on your custom patch. Not only does using the protocol avoid restrictions of a specific image type, but it avoids impedance mismatches, and provides better performance by deferring pixel computation until it is needed. When you need to access the pixels in an image, you simply convert the image to a representation (texture or buffer) using one of the methods defined by the `QCPlugInInputImageSource` protocol. Use a texture representation when you want to use input images on the GPU. Use a buffer representation when you want to use input images on the CPU.

Input images are opaque source objects that comply to this protocol. To create an image input port as an Objective-C 2.0 property, declare it as follows:

```
@property(dynamic) id<QCPlugInInputImageSource> inputImage;
```

To create an image input port dynamically. use the type `QCPortTypeImage`:

```
[self addInputPortWithType:QCPortTypeImage
                    forKey:@"inputImage"
            withAttributes:nil];
```

Tasks

Converting an Image to a Representation

- `lockTextureRepresentationWithColorSpace:forBounds:` (page 448)
  Creates an OpenGL texture representation from a subregion of the image source using the provided color space.
- `unlockTextureRepresentation` (page 452)
  Releases the OpenGL texture representation of the image source.
- `lockBufferRepresentationWithPixelFormat:colorSpace:forBounds:` (page 448)
  Creates a memory buffer representation from a subregion of the image source using the provided pixel format and color space.

- `bindTextureRepresentationToCGLContext:textureUnit:normalizeCoordinates:` (page 445)
  Binds the texture to a given texture unit and optionally scales or flips the texture.
- `unbindTextureRepresentationFromCGLContext:textureUnit:` (page 452)
  Unbinds the texture from a texture unit.
- `unlockBufferRepresentation` (page 452)
  Releases the memory buffer representation of the image source.

Getting Color Space Information

- `imageColorSpace` (page 447)
  Returns the color space of the image source.
- `shouldColorMatch` (page 449)
  Returns whether or not the image source should be color matched.

Getting Texture Information

- `texturePixelsWide` (page 451)
  Returns the width of the texture representation.
- `texturePixelsHigh` (page 451)
  Returns the height of the texture representation.
- `textureTarget` (page 451)
  Returns the texture target.
- `textureName` (page 450)
  Returns the texture name.
- `textureColorSpace` (page 449)
  Returns the color space of the texture representation.
- `textureFlipped` (page 449)
  Returns whether or not the contents of the texture are flipped vertically.
- `textureMatrix` (page 450)
  Returns a texture matrix.

Getting Image Buffer Information

- `imageBounds` (page 447)
  Returns the actual bounds of the image source expressed in pixels and aligned to integer boundaries.
- `bufferPixelsWide` (page 447)
  Returns the width of the image buffer representation.
- `bufferPixelsHigh` (page 446)
  Returns the height of the image buffer representation.
- `bufferPixelFormat` (page 446)
  Returns the pixel format of the image buffer representation.
- `bufferColorSpace` (page 446)
  Returns the color space of the image buffer representation.

- `bufferBaseAddress` (page 445)
  Returns the base address of the image buffer.
- `bufferBytesPerRow` (page 446)
  Returns the bytes per row of the buffer representation.

Instance Methods

bindTextureRepresentationToCGLContext:textureUnit:normalizeCoordinates:

Binds the texture to a given texture unit and optionally scales or flips the texture.

```
- (void) bindTextureRepresentationToCGLContext:(CGLContextObj)cgl_ctx
    textureUnit:(GLenum)unit normalizeCoordinates:(BOOL)flag
```

Parameters

cgl_ctx
  The CGL context to render to.)

unit
  The texture unit to bind to (such as, `GL_TEXTURE0`)

flag
  To apply a texture matrix to scale coordinates (from `[0, pixels]` to `[0,1]`) and flip them vertically (if necessary), pass `YES`.

Discussion
When you no longer need the texture, call `unbindTextureRepresentationFromCGLContext:textureUnit:` (page 452).

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

Declared In
`QCPlugIn.h`

bufferBaseAddress

Returns the base address of the image buffer.

```
- (const void*) bufferBaseAddress
```

Return Value
The base address of the buffer.

Discussion
The base address is guaranteed to be aligned on a 16-byte boundary.

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

Declared In
`QCPlugIn.h`

bufferBytesPerRow

Returns the bytes per row of the buffer representation.

```
- (NSUInteger) bufferBytesPerRow
```

Return Value
The number of bytes per row of the buffer.

Discussion
The number of bytes per row is guaranteed to be a multiple of 16.

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

Declared In
`QCPlugIn.h`

bufferColorSpace

Returns the color space of the image buffer representation.

```
- (CGColorSpaceRef) bufferColorSpace
```

Return Value
The color space of the image buffer.

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

Declared In
`QCPlugIn.h`

bufferPixelFormat

Returns the pixel format of the image buffer representation.

```
- (NSString*) bufferPixelFormat
```

Return Value
A string that specifies the pixel format. The supported formats are ARGB8 (8-bit alpha, red, green, blue), BGRA8 (8-bit blue, green, red, and alpha), RGBAf (floating-point, red, green, blue, alpha), I8 (8-bit intensity), and If (floating-point intensity).

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

Declared In
`QCPlugIn.h`

bufferPixelsHigh

Returns the height of the image buffer representation.

```
- (NSUInteger) bufferPixelsHigh
```

Return Value
The height, expressed in pixels.

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

See Also
- bufferPixelsHigh (page 446)

Declared In
QCPlugIn.h

bufferPixelsWide

Returns the width of the image buffer representation.

```
- (NSUInteger) bufferPixelsWide
```

Return Value
The width, expressed in pixels.

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

See Also
- bufferPixelsHigh (page 446)

Declared In
QCPlugIn.h

imageBounds

Returns the actual bounds of the image source expressed in pixels and aligned to integer boundaries.

```
- (NSRect) imageBounds;
```

Return Value
The bounds of the image source.

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

Declared In
QCPlugIn.h

imageColorSpace

Returns the color space of the image source.

```
- (CGColorSpaceRef) imageColorSpace
```

Return Value
The color space of the image source, typically RGB or Gray type.

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

Declared In
`QCPlugIn.h`

lockBufferRepresentationWithPixelFormat:colorSpace:forBounds:

Creates a memory buffer representation from a subregion of the image source using the provided pixel format and color space.

```
- (BOOL) lockBufferRepresentationWithPixelFormat:(NSString*)format
    colorSpace:(CGColorSpaceRef)colorSpace forBounds:(NSRect)bounds
```

Parameters

format
A pixel format that is compatible with the color space.

colorSpace
A Quartz color space that is compatible with the pixel format.

bounds
The bounds of the subregion, expressed as pixels, and aligned to integer boundaries.

Return Value
`YES` if successful; otherwise `NO`.

Discussion
The content of the buffer is read-only. You should not attempt to modify it.

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

See Also
`- unlockBufferRepresentation` (page 452)

Declared In
`QCPlugIn.h`

lockTextureRepresentationWithColorSpace:forBounds:

Creates an OpenGL texture representation from a subregion of the image source using the provided color space.

```
- (BOOL) lockTextureRepresentationWithColorSpace:(CGColorSpaceRef)colorSpace
    forBounds:(NSRect)bounds
```

Parameters

colorSpace
A Quartz color space.

bounds
The bounds of the subregion, expressed in pixels. They must be aligned to integer boundaries.

Return Value
`YES` is successful; `NO` if texture can't be created.

Discussion
Neither the content of the texture nor its states (for example, the wrap mode) must be modified; you can only draw with it. The texture is valid only in the plug-in context.

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

See Also
`- unlockTextureRepresentation` (page 452)

Declared In
`QCPlugIn.h`

shouldColorMatch

Returns whether or not the image source should be color matched.

```
- (BOOL) shouldColorMatch
```

Return Value
`NO` if the source is a mask or gradient; `YES` otherwise.

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

Declared In
`QCPlugIn.h`

textureColorSpace

Returns the color space of the texture representation.

```
- (CGColorSpaceRef) textureColorSpace
```

Return Value
The color space of the texture.

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

Declared In
`QCPlugIn.h`

textureFlipped

Returns whether or not the contents of the texture are flipped vertically.

```
- (BOOL) textureFlipped
```

Return Value
`YES` if the contents of the texture are flipped (upside-down); `NO` otherwise.

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

Declared In
`QCPlugIn.h`

textureMatrix

Returns a texture matrix.

```
- (const GLfloat*) textureMatrix
```

Return Value
A 4x4 texture matrix created by scaling (from [`0`, pixels] to [`0`,`1`]) and vertically flipping the texture coordinates; `NULL` if coordinate transformation is not required.

Discussion
This method is provided as a convenience for 2D textures to take care of two issues:

- Coordinates for rectangular textures are expressed in pixels rather than the normalized units used for power-of-two textures. The coordinates need to be normalized before you can process the texture.
- Texture coordinates are typically flipped by OpenGL for processing on the GPU and need to be flipped to the original coordinates.

You can take care of these two issues simply by loading a the matrix returned by this method onto the OpenGL stack. If you are not sure that your texture needs either of these operations, you can load the matrix on the OpenGL stack anyway, as it acts as an identity matrix if it's not needed.

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

Declared In
`QCPlugIn.h`

textureName

Returns the texture name.

```
- (GLuint) textureName
```

Return Value
The texture name.

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

Declared In
`QCPlugIn.h`

texturePixelsHigh

Returns the height of the texture representation.

```
- (NSUInteger) texturePixelsHigh
```

Return Value
The height of the texture, expressed in pixels.

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

See Also
- texturePixelsWide (page 451)

Declared In
QCPlugIn.h

texturePixelsWide

Returns the width of the texture representation.

```
- (NSUInteger) texturePixelsWide
```

Return Value
The width of the texture, expressed in pixels.

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

See Also
- texturePixelsHigh (page 451)

Declared In
QCPlugIn.h

textureTarget

Returns the texture target.

```
- (GLenum) textureTarget
```

Return Value
The texture target, either GL_TEXTURE_2D or GL_TEXTURE_RECTANGLE_EXT.

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

Declared In
QCPlugIn.h

unbindTextureRepresentationFromCGLContext:textureUnit:

Unbinds the texture from a texture unit.

```
- (void) unbindTextureRepresentationFromCGLContext:(CGLContextObj)cgl_ctx
    textureUnit:(GLenum)unit
```

Parameters

cgl_ctx

A CGL context.)

unit

The texture unit to unbind from (such as, `GL_TEXTURE0`)

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

See Also
`- bindTextureRepresentationToTextureUnit:normalizeCoordinates:` (page 445)

Declared In
`QCPlugIn.h`

unlockBufferRepresentation

Releases the memory buffer representation of the image source.

```
- (void) unlockBufferRepresentation
```

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

See Also
`- lockBufferRepresentationWithPixelFormat:colorSpace:` (page 448)

Declared In
`QCPlugIn.h`

unlockTextureRepresentation

Releases the OpenGL texture representation of the image source.

```
- (void) unlockTextureRepresentation
```

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

See Also
`- lockTextureRepresentationWithTarget:colorSpace:forBounds:` (page 448)

Declared In
`QCPlugIn.h`

QCPlugInOutputImageProvider Protocol Reference

Framework	/System/Library/Frameworks/Quartz.framework/Frameworks/QuartzComposer.framework
Declared in	QuartzComposer/QCPlugIn.h
Availability	Available in Mac OS X v10.5 and later.

Overview

The `QCPlugInOuputImageProvider` protocol eliminates the need to use explicit image types for the image output ports on a custom patch. The methods in this protocol are called by the Quartz Composer engine when the output image is needed. If your custom patch has an image output port, you need to implement the appropriate methods for rendering image data and to supply information about the rendering destination and the image bounds.

Output images are opaque provider objects that comply to this protocol. To create an image output port as an Objective-C 2.0 property, declare it as follows:

```
@property(dynamic) id<QCPlugInOutputImageProvider> outputImage;
```

To create an image input port dynamically use the type `QCPortTypeImage`:

```
[self addOutputPortWithType:QCPortTypeImage
                     forKey:@"outputImage"
             withAttributes:nil];
```

To write images to that port, you need to implement the methods in this protocol and create an internal class that represents the images produced by the custom patch. For example, a simple interface for an image provider is:

```
@interface MyOutputImage : NSObject <QCPlugInOutputImageProvider>
{
    NSUInteger _width;
    NSUInteger _height;
}
```

Tasks

Rendering an Image to a Destination

`- renderToBuffer:withBytesPerRow:pixelFormat:forBounds:` (page 456)
Renders a subregion of the image into the supplied memory buffer using the specified pixel format.

- `copyRenderedTextureForCGLContext:pixelFormat:bounds:isFlipped:` (page 455)
  Returns the name of an OpenGL texture of type `GL_TEXTURE_RECTANGLE_EXT` that contains a subregion of the image in a given pixel format.
- `renderWithCGLContext:forBounds:` (page 457)
  Renders a subregion of the image to the provided CGL context.
- `releaseRenderedTexture:forCGLContext:` (page 456)
  Releases the previously copied texture.

Providing Information About the Image

- `imageBounds` (page 455)
  Returns the bounds of the image expressed in pixels and aligned to integer boundaries.
- `imageColorSpace` (page 456)
  Returns the color space of the image or `NULL` if the image should not be color matched.
- `shouldColorMatch` (page 458)
  Returns whether the image should be color matched.

Providing Information About the Rendering Destination

- `supportedBufferPixelFormats` (page 458)
  Returns a list of pixel formats that are supported for rendering to a memory buffer.
- `supportedRenderedTexturePixelFormats` (page 458)
  Returns a list of pixel formats that are supported for rendering to an onscreen OpenGL context.
- `canRenderWithCGLContext:` (page 454)
  Returns whether the image data can be rendered into the provided CGL context.

Instance Methods

canRenderWithCGLContext:

Returns whether the image data can be rendered into the provided CGL context.

```
- (BOOL) canRenderWithCGLContext:(CGLContextObj)cgl_ctx
```

Parameters

ctx
  The CGL context that your image will be rendered to.

Return Value
`YES` if the image can be rendered into this CGL context; otherwise `NO`, in which case `renderToBuffer:withBytesPerRow:pixelFormat:forBounds:` (page 456) is called.

Discussion
If your image can render using any OpenGL context, simply return `YES`. If your code requires special extensions, you'll need to check for them and then provide the appropriate return value. For more information on checking for OpenGL capabilities supported by the hardware, see *OpenGL Programming Guide for Mac OS X*.

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

Declared In
`QCPlugIn.h`

copyRenderedTextureForCGLContext:pixelFormat:bounds:isFlipped:

Returns the name of an OpenGL texture of type `GL_TEXTURE_RECTANGLE_EXT` that contains a subregion of the image in a given pixel format.

```
- (GLuint) copyRenderedTextureForCGLContext:(CGLContextObj)cgl_ctx
    pixelFormat:(NSString*)format bounds:(NSRect)bounds isFlipped:(BOOL*)flipped
```

Parameters

cgl_ctx
The CGL context to render to.

format
A string that represents the pixel format of the texture.

bounds
The bounds of the subregion of the image.

isFlipped
Set to `YES` on output if the contents of the returned texture are vertically flipped.

Return Value
The name of an OpenGL texture of type `GL_TEXTURE_RECTANGLE_EXT` that contains a subregion of the image in a given pixel format or `0` if the texture can't be provided.

Discussion
Implement this method if you want to create the texture yourself or use framebuffer objects (FBO). Use `<OpenGL/CGLMacro.h>` to send commands to the OpenGL context. Make sure to preserve all the OpenGL states except the ones defined by `GL_CURRENT_BIT`.

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

Declared In
`QCPlugIn.h`

imageBounds

Returns the bounds of the image expressed in pixels and aligned to integer boundaries.

```
- (NSRect) imageBounds;
```

Return Value
The bounds of the image. Note that the `QCPlugIn` class does not support images that have infinite bounds.

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

Declared In
`QCPlugIn.h`

imageColorSpace

Returns the color space of the image or `NULL` if the image should not be color matched.

```
- (CGColorSpaceRef) imageColorSpace
```

Return Value
The color space of the image or `NULL`.

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

Declared In
`QCPlugIn.h`

releaseRenderedTexture:forCGLContext:

Releases the previously copied texture.

```
- (void) releaseRenderedTexture:(GLuint)name forCGLContext:(CGLContextObj)cgl_ctx;
```

Parameters

name
The name of the previously bound texture.

cgl_ctx
The CGL context.

Discussion
Your OpenGL code should save and restore all states *except* for those that are part of `GL_CURRENT_BIT` (vertex position, color, texture, and so on). Also use CGL macros instead of changing the current context, by including this statement:

```
#import <OpenGL/CGLMacro.h>
```

For more details, see *Quartz Composer Custom Patch Programming Guide*.

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

Declared In
`QCPlugIn.h`

renderToBuffer:withBytesPerRow:pixelFormat:forBounds:

Renders a subregion of the image into the supplied memory buffer using the specified pixel format.

```
- (BOOL) renderToBuffer:(void*)baseAddress withBytesPerRow:(NSUInteger)rowBytes
    pixelFormat:(NSString*)format forBounds:(NSRect)bounds
```

Parameters

baseAddress

The base address of the memory buffer. The Quartz Composer engine passes you an address that is aligned on a 16-byte boundary.

rowBytes

The number of bytes per row of the image data. The Quartz Composer engine guarantees this value is a multiple of 16.

format

The pixel format of the image data.

bounds

The bounds of the subregion.

Return Value

`YES` if the image is rendered successfully into the buffer; `NO` on failure or if the image provider doesn't support CPU rendering.

Discussion

The Quartz Composer engine calls this method when it needs pixels. It gives you the base address, the number of row bytes, and the format. Then, you write pixels to the buffer.

Availability

Available in Mac OS X v10.5 and later.

See Also

`- renderWithCGLContext:forBounds:` (page 457)

Declared In

`QCPlugIn.h`

renderWithCGLContext:forBounds:

Renders a subregion of the image to the provided CGL context.

```
- (BOOL) renderWithCGLContext:(CGLContextObj)cgl_ctx forBounds:(NSRect)bounds
```

Parameters

cgl_ctx

The CGL context to render to.

bounds

The bounds of the subregion.

Return Value

`YES` if successful; `NO` on failure or if the image provider doesn't support GPU rendering.

Discussion

The view port is set for you. The model view and projection matrixes are set to the identity.

Your OpenGL code should save and restore all states *except* for those that are part of `GL_CURRENT_BIT` (vertex position, color, texture, and so on). Also use CGL macros instead of changing the current context, by including this statement:

```
#import <OpenGL/CGLMacro.h>
```

For more details, see *Quartz Composer Custom Patch Programming Guide*.

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

See Also
`- renderToBuffer:withBytesPerRow:pixelFormat:forBounds:` (page 456)

Declared In
`QCPlugIn.h`

shouldColorMatch

Returns whether the image should be color matched.

```
- (BOOL) shouldColorMatch
```

Return Value
`NO` if the image is a mask or gradient; otherwise `YES`, which is the default.

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

Declared In
`QCPlugIn.h`

supportedBufferPixelFormats

Returns a list of pixel formats that are supported for rendering to a memory buffer.

```
- (NSArray*) supportedBufferPixelFormats
```

Return Value
A list of pixel formats, in order of preference, that the image can be rendered to in memory, or `nil` if the image provider does not support rendering to the CPU.

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

See Also
`- supportedRenderedTexturePixelFormats` (page 458)

Declared In
`QCPlugIn.h`

supportedRenderedTexturePixelFormats

Returns a list of pixel formats that are supported for rendering to an onscreen OpenGL context.

```
- (NSArray*) supportedRenderedTexturePixelFormats
```

Return Value
Returns the list of texture pixel formats supported by `copyRenderedTextureForCGLContext:pixelFormat:bounds:isFlipped:` (page 455) or `nil` if not supported.

Discussion
If this method returns nil, then Quartz Composer calls `canRenderWithCGLContext:` (page 454) /`renderWithCGLContext:forBounds:` (page 457).

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

See Also
`- supportedBufferPixelFormats` (page 458)

Declared In
`QCPlugIn.h`

Document Revision History

This table describes the changes to *Quartz Framework Reference*.

Date	Notes
2007-12-11	Added links to several new PDF Kit classes.
2007-01-25	Updated for Mac OS X v10.5.
2006-05-23	First publication of this content as a collection of previously published documents.

Document Revision History

Index

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W

Z