Quartz Reference Update

Graphics & Imaging > Quartz

2007-07-18

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

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

Simultaneously published in the United States and Canada.

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

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

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

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

Contents

Introduction to Quartz Reference Update

This document summarizes the symbols that have been added to the Quartz framework. The full reference documentation notes in what version a symbol was introduced, but sometimes it's useful to see only the new symbols for a given release.

If you are not familiar with this framework you should refer to the complete framework reference documentation.

Organization of This Document

Symbols are grouped by class or protocol for Objective-C and by header file for C. For each symbol there is a link to complete documentation, if available, and a brief description, if available.

See Also

For reference documentation on this framework, see *Quartz Framework Reference*.

10.5 Symbol Changes

This article lists the symbols added to `Quartz.framework` in Mac OS X v10.5.

Classes

All of the classes with new symbols are listed alphabetically, with their new class, instance, and delegate methods described.

CIFilter

Complete reference information is available in the `CIFilter` reference.

Instance Methods

`viewForUIConfiguration:excludedKeys:`	Returns a filter view for the filter.

IKFilterBrowserPanel (New)

Complete reference information is available in the `IKFilterBrowserPanel` reference.

Class Methods

`filterBrowserPanelWithStyleMask:`	Creates a shared instance of the IKFilterBrowserPanel class.

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.
`beginWithOptions:modelessDelegate:didEndSelector:contextInfo:`	Displays the filter browser in a new utility window, unless the filter browser is already open.
`filterBrowserViewWithOptions:`	Returns a view that contains a filter browser.
`filterName`	Returns the name of the filter that is currently selected in the filter browser.

`finish:`	Closes a filter browser view.
`runModalWithOptions:`	Displays the filter browser in a modal dialog that must be dismissed by the user but that is not attached to a window.

IKFilterBrowserView (New)

Complete reference information is available in the `IKFilterBrowserView` reference.

Instance Methods

`filterName`	Returns the name of the filter that is currently selected in the filter browser.
`setPreviewState:`	Sets the preview state.

IKFilterUIView (New)

Complete reference information is available in the `IKFilterUIView` reference.

Class Methods

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

Instance Methods

`filter`	Returns the Core Image filter associated with the view.
`initWithFrame:filter:`	Initializes a view that contains controls for the input parameters of a filter.
`objectController`	Returns the object controller for the bindings between the filter and its view.

IKImageBrowserView (New)

Complete reference information is available in the `IKImageBrowserView` reference.

Instance Methods

allowsEmptySelection	Returns whether an empty selection is allowed.
allowsMultipleSelection	Returns whether multiple selections are allowed.
allowsReordering	Returns whether the user can reorder items.
animates	Returns whether the receiver animates reordering and changes of the data source.
cellSize	Returns the cell size.
cellsStyleMask	Returns the appearance style mask for the cell.
collapseGroupAtIndex:	Collapses a group at the specified index.
constrainsToOriginalSize	Returns whether the receiver constrains the cell's image to its original size.
contentResizingMask	Returns the receiver's content resizing mask, which determines how its content is resized while zooming.
dataSource	Returns the data source of the receiver.
delegate	Returns the delegate of the receiver.
draggingDestinationDelegate	Returns the dragging destination delegate of the receiver.
expandGroupAtIndex:	Expands a group at the specified index.
indexAtLocationOfDroppedItem	Returns the index of the cell where the drop operation occurred.
indexOfItemAtPoint:	Returns the index of the item at the specified location.
initWithFrame:	Initializes a newly allocated image browser view with the provided frame rectangle.
isGroupExpandedAtIndex:	Returns whether the group at the provided index is expanded.
itemFrameAtIndex:	Returns the frame rectangle for the item located at the specified index.
reloadData	Marks the receiver as needing its data reloaded.
scrollIndexToVisible:	Scrolls the receiver to the item at the specified index.
selectionIndexes	Returns the indexes of the selected cells.

setAllowsEmptySelection:	Controls whether an empty selection is allowed.
setAllowsMultipleSelection:	Controls whether the user can select more than one cell at a time.
setAllowsReordering:	Controls whether the user can reorder items.
setAnimates:	Controls whether the receiver animates reordering and changes of the data source.
setCellSize:	Sets the cell size.
setCellsStyleMask:	Defines the appearance style of the cells.
setConstrainsToOriginalSize:	Sets whether the receiver constrains the cell's image to its original size.
setContentResizingMask:	Determines how the receiver resizes its content when zooming.
setDataSource:	Sets the data source of the receiver.
setDelegate:	Sets the delegate of the receiver.
setDraggingDestinationDelegate:	Sets the dragging destination delegate of the receiver.
setSelectionIndexes:byExtendingSelection:	Selects cells at the specified indexes.
setZoomValue:	Sets the zoom value.
zoomValue	Returns the current zoom value.

Delegate Methods

imageBrowser:backgroundWasRightClickedWithEvent:	Performs custom tasks when the user right-clicks the image browser view background.
imageBrowser:cellWasDoubleClickedAtIndex:	Performs custom tasks when the user double-clicks an item in the image browser view.
imageBrowser:cellWasRightClickedAtIndex:withEvent:	Performs custom tasks when the user right-clicks an item in the image browser view.
imageBrowser:groupAtIndex:	Returns the group at the specified index.
imageBrowser:itemAtIndex:	Returns an object for the item in an image browser view that corresponds to the specified index.

imageBrowser:moveItemsAtIndexes:toIndex:	Signals that the specified items should be moved to the specified destination.
imageBrowser:removeItemsAtIndexes:	Signals that a remove operation should be applied to the specified items.
imageBrowser:writeItemsAtIndexes:toPasteboard:	Signals that a drag should begin.
imageBrowserSelectionDidChange:	Performs custom tasks when the selection changes.
numberOfGroupsInImageBrowser:	Returns the number of groups in an image browser view.
numberOfItemsInImageBrowser:	Returns the number of records managed by the data source object.

IKImageEditPanel (New)

Complete reference information is available in the `IKImageEditPanel` reference.

Class Methods

sharedImageEditPanel	Creates a shared instance of an image editing panel.

Instance Methods

dataSource	Returns the data source associated with an image editing panel.
reloadData	Reloads the data from the data associated with an image editing panel.
setDataSource:	Sets a data source for an image editing panel.

IKImagePicker (New)

Complete reference information is available in the `IKImagePicker` reference.

Class Methods

imagePicker	

Instance Methods

beginImagePickerSheetForWindow:withDelegate: didEndSelector:contextInfo:	
beginImagePickerWithDelegate:didEndSelector: contextInfo:	

IKImageView (New)

Complete reference information is available in the IKImageView reference.

Instance Methods

autohidesScrollers	Specifies the automatic-hiding scroll bar state for the image view.
autoresizes	Specifies the automatic resizing state for the image view.
backgroundColor	Specifies the background color for the image view.
convertImagePointToViewPoint:	Converts an image coordinate to an image view coordinate.
convertImageRectToViewRect:	Converts an image rectangle to an image view rectangle.
convertViewPointToImagePoint:	Converts an image view coordinate to an image coordinate.
convertViewRectToImageRect:	Converts an image view rectangle to an image rectangle.
currentToolMode	Specifies the current tool mode for the image view.
delegate	Specifies the delegate object of the receiver.
doubleClickOpensImageEditPanel	Specifies the image-opening state of the editing pane in the image view.
editable	Specifies the editable state for the image view.
flipImageHorizontal:	Flips an image along the horizontal axis.
flipImageVertical:	Flips an image along the vertical axis.
hasHorizontalScroller	Specifies the horizontal scroll bar state for the image view.
hasVerticalScroller	Specifies the vertical scroll bar state for the image view.
image	Returns the image associated with the view, after any image corrections.
imageCorrection	Specifies a Core Image filter for image correction.
imageProperties	Returns the metadata for the image in the view.
imageSize	Returns the size of the image in the image view.

overlayForType:	Returns the Core Animation layer associated with a layer type.
rotationAngle	Specifies the rotation angle for the image view.
scrollToPoint:	Scrolls the view to the specified point.
scrollToRect:	Scrolls the view so that it includes the provided rectangular area.
setImage:imageProperties:	Sets the image to display in an image view.
setImageWithURL:	Initializes an image view with the image specified by a URL.
setImageZoomFactor:centerPoint:	Sets the zoom factor at the provided origin.
setOverlay:forType:	Sets an overlay type for a Core Animation layer.
setRotationAngle:centerPoint:	Sets the rotation angle at the provided origin.
supportsDragAndDrop	Specifies the drag-and-drop support state for the image view.
zoomFactor	Specifies the zoom factor for the image view.
zoomImageToActualSize:	Zooms the image so that it is displayed using its true size.
zoomImageToFit:	Zooms the image so that it fits in the image view.
zoomImageToRect:	Zooms the image so that it fits in the specified rectangle.

IKPictureTaker (New)

Complete reference information is available in the IKPictureTaker reference.

Class Methods

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

Instance Methods

beginPictureTakerSheetForWindow:withDelegate:didEndSelector:contextInfo:	Opens a picture taker as a sheet whose parent is the specified window.
beginPictureTakerWithDelegate:didEndSelector:contextInfo:	Opens a picture taker pane.
inputImage	Returns the input image associated with the picture taker.
mirroring	Returns whether video mirroring is enabled during snapshots.

`outputImage`	Returns the edited image.
`popUpRecentsMenuForView:withDelegate:didEndSelector:contextInfo:`	Displays the Open Recent popup menu associated with the picture taker.
`runModal`	Opens a modal picture taker dialog.
`setInputImage:`	Set the image input for the picture taker.
`setMirroring:`	Controls whether the receiver enables video mirroring during snapshots.

IKSaveOptions (New)

Complete reference information is available in the `IKSaveOptions` reference.

Instance Methods

`addSaveOptionsAccessoryViewToSavePanel:`	Adds a save options accessory view to a save panel.
`imageProperties`	Returns a dictionary of image properties that reflects the user's selection.
`imageUTType`	Returns the uniform type identifier that reflects the user's selection.
`initWithImageProperties:imageUTType:`	Initializes a save options accessory pane for the provided image properties and uniform type identifier.
`userSelection`	Returns a dictionary that contains the save options selected by the user.

IKSlideshow (New)

Complete reference information is available in the `IKSlideshow` reference.

Class Methods

`canExportToApplication:`	Finds out whether the slideshow can export its contents to an application.
`exportSlideshowItem:toApplication:`	Exports a slideshow item to the application that has the provided bundle identifier.
`sharedSlideshow`	Returns a shared instance of a slideshow.

Instance Methods

`autoPlayDelay`	Controls the interval of time before a slideshow starts to play automatically.
`indexOfCurrentSlideshowItem`	Returns the index of the current slideshow item.
`reloadData`	Reloads the data for a slideshow.
`reloadSlideshowItemAtIndex:`	Reloads the data for a slideshow, starting at the specified index.
`runSlideshowWithDataSource:inMode:options:`	Runs a slideshow that contains the specified kind of items, provided from a data source.
`stopSlideshow:`	Stops a slideshow.

NSObject

Complete reference information is available in the `NSObject` reference.

Instance Methods

`imageBrowser:cellAtIndex:`	
`imageBrowser:moveCellsAtIndexes:toIndex:`	
`imageBrowser:removeCellsAtIndexes:`	
`imageBrowser:writeCellsAtIndexes:toPasteboard:`	
`imageRepresentation`	Returns the image to display.
`imageRepresentationType`	Returns the representation type of the image to display.
`imageSubtitle`	Returns the display subtitle of the image.
`imageTitle`	Returns the display title of the image.
`imageUID`	Returns a unique string that identifies the data source item.
`imageVersion`	Returns the version of the item.
`isSelectable`	Returns whether this item is selectable.
`numberOfCellsInImageBrowser:`	

PDFAction (New)

Complete reference information is available in the `PDFAction` reference.

Instance Methods

`type`	Returns the type of the action.

PDFActionGoTo (New)

Complete reference information is available in the `PDFActionGoTo` reference.

Instance Methods

`destination`	Returns the destination associated with the action.
`initWithDestination:`	
`setDestination:`	

PDFActionNamed (New)

Complete reference information is available in the `PDFActionNamed` reference.

Instance Methods

`initWithName:`	Initializes the PDFActionName object with the specified named action.
`name`	Returns the name of the named action.
`setName:`	Sets the name of the named action.

PDFActionRemoteGoTo (New)

Complete reference information is available in the `PDFActionRemoteGoTo` reference.

Instance Methods

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

pageIndex	Returns the zero-based page index referenced by the remote go-to action.
point	Returns the point, in page space, on the page referenced by the remote go-to action.
setPageIndex:	Sets the zero-based page index referenced by the remote go-to action.
setPoint:	Sets the point, in page space, on the page referenced by the remote go-to action.
setURL:	Sets the URL of the document referenced by the remote go-to action.
URL	Returns the URL of the document referenced by the remote go-to action.

PDFActionResetForm (New)

Complete reference information is available in the PDFActionResetForm reference.

Instance Methods

fields	Returns an array of fields associated with the reset action.
fieldsIncludedAreCleared	Returns whether the fields associated with the reset action are cleared when the action is performed.
init	Initializes a reset form action.
setFields:	Sets the array of fields associated with the reset action.
setFieldsIncludedAreCleared:	Sets whether the fields associated with the reset action are cleared when the action is performed.

PDFActionURL (New)

Complete reference information is available in the PDFActionURL reference.

Instance Methods

initWithURL:	Initializes a URL action with the specified URL.
setURL:	Sets the URL associated with the URL action.
URL	Returns the URL associated with the URL action.

PDFAnnotation

Complete reference information is available in the PDFAnnotation reference.

Instance Methods

modificationDate	Returns the modification date of the annotation.
mouseUpAction	Returns the optional action performed when a user releases the mouse button within an annotation.
popup	Returns the pop-up annotation associated with an annotation.
removeAllAppearanceStreams	
setModificationDate:	Sets the modification date of the annotation.
setMouseUpAction:	Sets the action performed when a user releases the mouse button within an annotation.
setPopup:	Sets the pop-up annotation associated with an annotation.
setUserName:	Sets the name of the user who created the annotation.
userName	Returns the name of the user who created the annotation.

PDFAnnotationButtonWidget

Complete reference information is available in the PDFAnnotationButtonWidget reference.

Instance Methods

allowsToggleToOff	Returns a Boolean value indicating whether a radio button behaves in a toggling manner.
caption	Returns the text of the label on a push button control.
fieldName	Returns the internal name of a field (used for reset-form actions).
font	Returns the font used in the control's label.
fontColor	Returns the font color used in the control's label.
isHighlighted	Returns a Boolean value that indicates whether the control is highlighted when it is drawn.
onStateValue	Returns the string associated with the on state of a radio button or checkbox control.
setCaption:	Sets the text of the label on a push button control.

setControlType:	Sets the type of the control.
setFieldName:	Sets the internal name of a field (used for reset-form actions).
setFont:	Sets the font of the control's label.
setFontColor:	Sets the font color used in the control's label.
setOnStateValue:	Sets the string that is associated with the on state of a radio button or checkbox control.

PDFAnnotationChoiceWidget (New)

Complete reference information is available in the PDFAnnotationChoiceWidget reference.

Instance Methods

backgroundColor	Returns the color of the widget annotation background.
choices	Returns an array of strings that represent the items available in the list or pop-up menu of the choice widget annotation.
fieldName	Returns the internal field name associated with the widget annotation.
font	Returns the font used to display the text in the widget annotation.
fontColor	Returns the font color used to display the text in the widget annotation.
isListChoice	Returns a Boolean value indicating whether the widget annotation is a list.
setBackgroundColor:	Sets the background color of the widget annotation.
setChoices:	Sets the items available in the list or pop-up menu of the choice widget annotation.
setFieldName:	Sets the internal field name associated with the widget annotation's value.
setFont:	Sets the font used to display the text in the widget annotation.
setFontColor:	Sets the font color used to display the text in the widget annotation.
setIsListChoice:	Sets whether the widget annotation is a list.
setStringValue:	Sets the selection in the widget annotation.
stringValue	Returns the selection in the widget annotation.

PDFAnnotationFreeText

Complete reference information is available in the `PDFAnnotationFreeText` reference.

Instance Methods

`fontColor`	Returns the font color used in the text field of the annotation.
`setFontColor:`	Sets the font color used in the text field of the annotation.

PDFAnnotationLine

Complete reference information is available in the `PDFAnnotationLine` reference.

Instance Methods

`interiorColor`	Returns the color used to fill the ornament at the ends of the line.
`setInteriorColor:`	Sets the color used to fill the ornament at the ends of the line.

PDFAnnotationPopup (New)

Complete reference information is available in the `PDFAnnotationPopup` reference.

Instance Methods

`isOpen`	Returns a Boolean value indicating whether the pop-up is open.
`setIsOpen:`	Sets the open state of the pop-up menu.

PDFAnnotationStamp (New)

Complete reference information is available in the `PDFAnnotationStamp` reference.

Instance Methods

`name`	Returns name associated with the stamp annotation.
`setName:`	Sets the name associated with the stamp annotation.

PDFAnnotationTextWidget

Complete reference information is available in the PDFAnnotationTextWidget reference.

Instance Methods

backgroundColor	Returns the background color of the annotation text field.
fieldName	Returns the internal name for the annotation text field.
fontColor	Returns the font color used for the annotation's text field.
maximumLength	Returns the maximum number of characters allowed in the annotation string.
rotation	Returns the rotation angle of the annotation text field in degrees.
setBackgroundColor:	Sets the background color of the annotation text field.
setFieldName:	Sets the internal field name for the annotation text field.
setFontColor:	Sets the font color used for the annotation's text field.
setMaximumLength:	Sets the maximum number of characters allowed in the annotation string.
setRotation:	Sets the rotation angle of the annotation text field in degrees.

PDFDestination

Complete reference information is available in the PDFDestination reference.

Instance Methods

compare:	Returns a comparison result that indicates the location of the destination in the document, relative to the current position.

PDFDocument

Complete reference information is available in the PDFDocument reference.

Instance Methods

beginFindStrings:withOptions:	Asynchronously finds all instances of the specified array of strings in the document.

`pageClass`	Returns the class that is allocated and initialized when page objects are created for the document.
`setOutlineRoot:`	Sets the document's root outline to a PDF outline object.

PDFOutline

Complete reference information is available in the `PDFOutline` reference.

Instance Methods

`action`	Returns the action performed when users click the outline.
`index`	Returns the index of the outline.
`init`	Initializes a PDFOutline object.
`insertChild:atIndex:`	Inserts the specified outline object at the specified index.
`isOpen`	Returns a Boolean value that indicates whether the outline object is initially disclosed.
`parent`	Returns the parent outline object of the outline (returns NULL if called on the root outline object).
`removeFromParent`	Removes the outline object from its parent (does nothing if outline object is the root outline object).
`setAction:`	Sets the action associated with the outline.
`setDestination:`	Sets the destination associated with the outline.
`setIsOpen:`	Sets the initial disclosure state of the outline object.
`setLabel:`	Sets the label for the outline (has no effect on the root outline object).

PDFPage

Complete reference information is available in the `PDFPage` reference.

Instance Methods

`initWithImage:`	Creates a new PDFPage object and initializes it with the specified NSImage object.
`transformContextForBox:`	Transforms the current context, given the specified box.

PDFSelection

Complete reference information is available in the `PDFSelection` reference.

Instance Methods

`addSelections:`	Adds the specified array of selections to the receiving selection.
`color`	Returns the color used to draw the selection.
`initWithDocument:`	Returns an empty PDFSelection object.
`selectionsByLine`	Returns an array of selections, one for each line of text covered by the receiver.
`setColor:`	Sets the color used for the drawing of a selection in both active and inactive states.

PDFThumbnailView (New)

Complete reference information is available in the `PDFThumbnailView` reference.

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.
`allowsMultipleSelection`	Returns a Boolean value indicating whether users can select multiple thumbnails in the thumbnail view at one time.
`backgroundColor`	Returns the color used in the background of the thumbnail view.
`labelFont`	Returns the font used to label the thumbnails.
`maximumNumberOfColumns`	Returns the maximum number of columns of thumbnails the thumbnail view can display.
`PDFView`	Returns the PDFView object associated with the thumbnail view.
`selectedPages`	Returns an array of PDF pages that correspond to the selected thumbnails in the thumbnail view.
`setAllowsDragging:`	Sets whether users can drag thumbnails within the thumbnail view; that is, re-order pages in the document.
`setAllowsMultipleSelection:`	Sets whether the thumbnail view allows users to select more than one thumbnail at a time.
`setBackgroundColor:`	Sets the color used in the background of the thumbnail view.

setLabelFont:	Sets the font used to label the thumbnails.
setMaximumNumberOfColumns:	Sets the maximum number of columns of thumbnails the thumbnail view can display.
setPDFView:	Associates the specified PDFView object with the thumbnail view.
setThumbnailSize:	Sets the maximum width and height of the thumbnails in the thumbnail view.
thumbnailSize	Returns the maximum width and height of the thumbnails in the thumbnail view.

PDFView

Complete reference information is available in the PDFView reference.

Instance Methods

annotationsChangedOnPage:	Tells the PDF view that an annotation on the specified page has changed.
drawPagePost:	For use by subclasses of PDFView for post-page rendering.
goToRect:onPage:	Navigates to the specified rectangle on the specified page.
highlightedSelections	Returns the array of selections that are highlighted using setHighlightedSelections.
performAction:	Performs the specified action.
printWithInfo:autoRotate:pageScaling:	Prints the document with the specified printer and page-scaling information.
setCurrentSelection:animate:	Sets the selection, in an animated way, if desired.
setHighlightedSelections:	Highlights the specified array of selections.
visiblePages	Returns an array of PDFPage objects that represent the currently visible pages.

Delegate Methods

PDFViewOpenPDF:forRemoteGoToAction:	Delegate method that opens a specified page.
PDFViewPerformFind:	Delegate method that performs a find operation.
PDFViewPerformGoToPage:	Delegate method that performs a go-to operation.

`PDFViewPerformPrint:`	Delegate method that prints the current document.
`PDFViewPrintJobTitle:`	Delegate method that overrides the job title used when the PDFView is printed.

QCComposition (New)

Complete reference information is available in the `QCComposition` reference.

Class Methods

`compositionWithData:`	Returns an autoreleased composition object initialized with the contents of a Quartz Composer composition file.
`compositionWithFile:`	Returns an autoreleased composition object initialized with a Quartz Composer composition file.

Instance Methods

`attributes`	Returns the attributes of the composition.
`identifier`	Returns the unique and persistent identifier for the composition from the composition repository.
`inputKeys`	Returns an array listing the keys that identify the input ports of the root patch of the composition.
`outputKeys`	Returns an array listing the keys that identify the output ports of the root patch of the composition.
`protocols`	Returns the list of protocols to which the composition conforms.

QCCompositionLayer (New)

Complete reference information is available in the `QCCompositionLayer` reference.

Class Methods

`compositionLayerWithComposition:`	Creates and returns an instance of a composition layer using the provided Quartz Composer composition.
`compositionLayerWithFile:`	Creates and returns an instance of a composition layer using the Quartz Composer composition in the specified file.

Instance Methods

`composition`	Returns the composition associated with the layer.
`initWithComposition:`	Initializes and returns a composition layer using the provided Quartz Composer composition.
`initWithFile:`	Initializes and returns a composition layer using the Quartz Composer composition in the specified file.

QCCompositionParameterView (New)

Complete reference information is available in the `QCCompositionParameterView` reference.

Instance Methods

`backgroundColor`	Retrieves the background color of the composition parameter view.
`compositionRenderer`	Returns the renderer object associated with the composition parameter view.
`delegate`	Returns the composition parameter view delegate.
`drawsBackground`	Returns whether the composition parameter view draws its background.
`hasParameters`	Checks whether the composition that is currently edited by the composition parameter view has any input parameters.
`setBackgroundColor:`	Sets the background color of the composition parameter view.
`setCompositionRenderer:`	Sets the composition parameter view for editing the input parameters of the provided renderer object.
`setDelegate:`	Sets the composition parameter view delegate.
`setDrawsBackground:`	Sets whether the composition parameter view draws its background.

Delegate Methods

`compositionParameterView:`
`didChangeParameterWithKey:`	

`compositionParameterView:`
`shouldDisplayParameterWithKey:attributes:`
	Allows you to define which composition parameters are visible in the user interface when the composition parameter view refreshes.

QCCompositionPickerPanel (New)

Complete reference information is available in the QCCompositionPickerPanel reference.

Class Methods

sharedCompositionPickerPanel	Returns the shared instance of the composition picker panel.

Instance Methods

compositionPickerView	Returns the composition picker view used by the panel so that it can be configured.

QCCompositionPickerView (New)

Complete reference information is available in the QCCompositionPickerView reference.

Instance Methods

allowsEmptySelection	Retrieves the empty-selection state of the composition picker view.
backgroundColor	Returns the background color of the composition picker view.
compositionAspectRatio	Retrieves the aspect ratio used to display compositions in the composition picker view.
compositions	Returns the list of compositions that are currently in the composition picker view.
delegate	Retrieves the composition picker view delegate.
drawsBackground	Returns whether the composition picker view draws its background.
isAnimating	Returns whether or not the composition picker view is currently animating its composition.
maxAnimationFrameRate	Retrieves the maximum frame rate for animating compositions.
numberOfColumns	Retrieves the number of columns in the composition picker view.
numberOfRows	Retrieves the number of rows in the composition picker view.

`resetDefaultInputValues`	Clears all previously set default values for composition input parameters.
`selectedComposition`	Returns the composition that is currently selected in the composition picker view.
`setAllowsEmptySelection:`	Sets whether to allow an empty selection in the composition picker view.
`setBackgroundColor:`	Sets the background color for the composition picker view.
`setCompositionAspectRatio:`	Sets the aspect ratio used to display compositions in the composition picker view.
`setCompositionsFromRepositoryWithProtocol:andAttributes:`	Sets the compositions in the composition picker view to those that match the specified criteria.
`setDefaultValue:forInputKey:`	Sets the default value to use for a composition input parameter.
`setDelegate:`	Sets the composition picker view delegate.
`setDrawsBackground:`	Sets whether the composition picker view draws its background.
`setMaxAnimationFrameRate:`	Sets the maximum frame rate for animating compositions.
`setNumberOfColumns:`	Sets the number of columns in the composition picker view.
`setNumberOfRows:`	Sets the number of rows in the composition picker view.
`setSelectedComposition:`	Sets a composition as selected in the composition picker view.
`setShowsCompositionNames:`	Enables the display of composition names in the composition picker view.
`showsCompositionNames`	Retrieves whether composition names can be shown in the composition picker view.
`startAnimation:`	Starts animating the composition in the composition picker view.
`stopAnimation:`	Stops animating the composition that is currently animating in the composition picker view.

Delegate Methods

`compositionPickerView:didSelectComposition:`	Performs custom tasks when the selected composition in the composition picker view changes.
`compositionPickerViewDidStartAnimating:`	Performs custom tasks when the composition picker view starts animating a composition.
`compositionPickerViewWillStopAnimating:`	Performs custom tasks when the composition picker view stops animating a composition.

QCCompositionRepository (New)

Complete reference information is available in the `QCCompositionRepository` reference.

Class Methods

`sharedCompositionRepository`	Returns the shared instance of the composition repository.

Instance Methods

`allCompositions`	Returns an array that contains all compositions currently in the composition repository.
`compositionsWithProtocols:andAttributes:`	Returns an array of compositions that match a set of criteria.
`compositionWithIdentifier:`	Returns the composition that corresponds to the identifier.

QCPlugIn (New)

Complete reference information is available in the `QCPlugIn` reference.

Class Methods

`attributes`	Returns a dictionary that contains strings for the user interface that describe the custom patch.
`attributesForPropertyPortWithKey:`	Returns a dictionary that contains strings for the user interface that describe the optional attributes for ports created from properties.
`executionMode`	Returns the execution mode of the custom patch.

loadPlugInAtPath:	Loads a Quartz Composer plug-in bundle from the specified path.
plugInKeys	Returns the keys for the internal settings of a custom patch.
registerPlugInClass:	Registers a QCPlugIn subclass.
sortedPropertyPortKeys	Returns and array of property port keys in the order you want them to appear in the user interface.
timeMode	Returns the time mode for the custom patch.

Instance Methods

addInputPortWithType:forKey:withAttributes:	Adds an input port of the specified type and associates a key and attributes with the port.
addOutputPortWithType:forKey:withAttributes:	Adds an output port of the specified type and associates a key and attributes with the port.
createViewController	Creates and returns a view controller for the Settings pane of a custom patch.
didValueForInputKeyChange:	Returns whether the input port value changed since the last execution of the custom patch.
disableExecution:	Allows you to perform custom tasks when the execution of the QCPlugIn object is paused.
enableExecution:	Allows you to perform custom tasks when the execution of the QCPlugIn object is resumed.
execute:atTime:withArguments:	Performs the processing or rendering tasks appropriate for the custom patch.
removeInputPortForKey:	Removes the input port for a given key.
removeOutputPortForKey:	Removes the output port for a given key.
serializedValueForKey:	Provides custom serialization for patch internal settings that do not comply to the NSCoding protocol.
setSerializedValue:forKey:	Provides custom deserialization for patch internal settings that were previously serialized using the method serializedValueForKey:.
setValue:forOutputKey:	Sets the value of an output port.
startExecution:	Allows you to perform custom setup tasks before the Quartz Composer engine starts rendering.

`stopExecution:`	Allows you to perform custom tasks when the QCPlugIn object stops executing.
`valueForInputKey:`	Returns the current value for an input port.

QCPlugInViewController (New)

Complete reference information is available in the `QCPlugInViewController` reference.

Instance Methods

`initWithPlugIn:viewNibName:`	Creates and initializes a controller for the specified QCPlugIn object and nib file.
`plugIn`	Returns the QCPlugIn object associated with the view controller for the custom patch.

QCRenderer

Complete reference information is available in the `QCRenderer` reference.

Instance Methods

`composition`	Returns the composition object associated with the renderer.
`createSnapshotImageOfType:`	Returns the current image in the OpenGL context associated with the renderer, as an image object of the provided image type.
`initOffScreenWithSize:colorSpace:composition:`	Creates an offscreen renderer of a given size with the provided color space and composition object.
`initWithCGLContext:pixelFormat:colorSpace:composition:`	Creates a renderer object with a CGLContextObj object, a pixel format, a color space, and a composition object.
`initWithComposition:colorSpace:`	Creates a renderer object with a composition object and a color space.
`snapshotImage`	Returns an NSImage object of the current image in the OpenGL context associated with the renderer.

QCView

Complete reference information is available in the `QCView` reference.

Instance Methods

`createSnapshotImageOfType:`	Returns the current image in the view as an image object of the provided image type.
`isPausedRendering`	Returns whether or not the rendering in the view is paused.
`loadComposition:`	Loads a QCComposition object into the view.
`loadedComposition`	Returns the composition loaded in the view.
`openGLContext`	Returns the OpenGL context used by the view.
`openGLPixelFormat`	Returns the OpenGL pixel format used by the view.
`pauseRendering`	Pauses rendering in the view.
`play:`	Plays or pauses a composition in a view.
`renderAtTime:arguments:`	Overrides to perform your custom operations prior to or after rendering a frame of a composition.
`resumeRendering`	Resumes rendering a paused composition.
`snapshotImage`	Returns an NSImage object of the current image in the view.
`unloadComposition`	Unloads the composition from the view.

QuartzFilter (New)

Complete reference information is available in the `QuartzFilter` reference.

Class Methods

`quartzFilterWithOutputIntents:`	
`quartzFilterWithProperties:`	
`quartzFilterWithURL:`	

Instance Methods

`applyToContext:`	

`localizedName`	
`properties`	
`removeFromContext:`	
`url`	

QuartzFilterManager (New)

Complete reference information is available in the `QuartzFilterManager` reference.

Class Methods

`filterManager`	
`filtersInDomains:`	

Instance Methods

`delegate`	
`filterPanel`	
`filterView`	
`importFilter:`	
`selectedFilter`	
`selectFilter:`	
`setDelegate:`	

Delegate Methods

`quartzFilterManager:didAddFilter:`	
`quartzFilterManager:didModifyFilter:`	
`quartzFilterManager:didRemoveFilter:`	
`quartzFilterManager:didSelectFilter:`	

QuartzFilterView (New)

Complete reference information is available in the `QuartzFilterView` reference.

Instance Methods

sizeToFit	

Protocols

All of the protocols with new symbols are listed alphabetically, with their new methods described.

IKFilterCustomUIProvider (New)

Complete reference information is available in the IKFilterCustomUIProvider reference.

Instance Methods

provideViewForUIConfiguration:excludedKeys:	Provides a custom view for a filter.

IKImageEditPanelDataSource (New)

Complete reference information is available in the IKImageEditPanelDataSource reference.

Instance Methods

image	Returns an image.
imageProperties	Returns a dictionary of the image properties associated with the image in the image edit panel.
setImage:imageProperties:	Sets an image with the specified properties.
thumbnailWithMaximumSize:	Returns a thumbnail image whose size is no larger than the specified size.

IKSlideshowDataSource (New)

Complete reference information is available in the IKSlideshowDataSource reference.

Instance Methods

canExportSlideshowItemAtIndex:toApplication:	Reports whether the export button should be enabled for a a slideshow item.

nameOfSlideshowItemAtIndex:	Returns the display name for item at the specified index.
numberOfSlideshowItems	Returns the number of items in a slideshow.
slideshowDidChangeCurrentIndex:	Performs custom tasks when the slideshow changes to the item at the specified index.
slideshowDidStop	Performs custom tasks when the slideshow stops.
slideshowItemAtIndex:	Returns the item for a given index
slideshowWillStart	Performs custom tasks when the slideshow is about to start.

QCCompositionRenderer (New)

Complete reference information is available in the `QCCompositionRenderer` reference.

Instance Methods

attributes	Returns the attributes of the composition associated with the renderer.
inputKeys	Returns an array that contains the keys that identify the input ports of the root patch of the composition.
outputKeys	Returns an array that contains the keys that identify the output ports of the root patch of the composition.
propertyListFromInputValues	Returns a property list object that represents the current values for all the input keys of the composition.
setInputValuesWithPropertyList:	Sets the values for the input keys of the composition from a previously saved property list.
setValue:forInputKey:	Sets the value for an input port of a composition.
userInfo	Returns a mutable dictionary for storing arbitrary information.
valueForInputKey:	Returns the value for an input port of a composition.
valueForOutputKey:	Returns the value for an output port of a composition.
valueForOutputKey:ofType:	Returns the current value on an output port (identified by its key) of the root patch of the composition.

QCPlugInContext (New)

Complete reference information is available in the QCPlugInContext reference.

Instance Methods

bounds	Returns the bounds of the rendering context.
CGLContextObj	Returns the destination CGL context to use for OpenGL rendering from within the execution method.
colorSpace	Returns the color space used by the rendering context.
compositionURL	
logMessage:	Writes a message to the Quartz Composer log.
outputImageProviderFromBufferWithPixelFormat: pixelsWide:pixelsHigh:baseAddress:bytesPerRow: releaseCallback:releaseContext:colorSpace: shouldColorMatch:	Returns an image provider from a single memory buffer.
outputImageProviderFromTextureWithPixelFormat: pixelsWide:pixelsHigh:name:flipped: releaseCallback:releaseContext:colorSpace: shouldColorMatch:	Returns an image provider from an OpenGL texture.
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.

QCPlugInInputImageSource (New)

Complete reference information is available in the QCPlugInInputImageSource reference.

Instance Methods

bindTextureRepresentationToCGLContext:textureUnit: normalizeCoordinates:	Binds the texture to a given texture unit and optionally scales or flips the texture.
bufferBaseAddress	Returns the base address of the image buffer.

`bufferBytesPerRow`	Returns the bytes per row of the buffer representation.
`bufferColorSpace`	Returns the color space of the image buffer representation.
`bufferPixelFormat`	Returns the pixel format of the image buffer representation.
`bufferPixelsHigh`	Returns the height of the image buffer representation.
`bufferPixelsWide`	Returns the width of the image buffer representation.
`imageBounds`	Returns the actual bounds of the image source expressed in pixels and aligned to integer boundaries.
`imageColorSpace`	Returns the color space of the image source.
`lockBufferRepresentationWithPixelFormat:colorSpace:forBounds:`	Creates a memory buffer representation from a subregion of the image source using the provided pixel format and color space.
`lockTextureRepresentationWithColorSpace:forBounds:`	Creates an OpenGL texture representation from a subregion of the image source using the provided color space.
`shouldColorMatch`	Returns whether or not the image source should be color matched.
`textureColorSpace`	Returns the color space of the texture representation.
`textureFlipped`	Returns whether or not the contents of the texture are flipped vertically.
`textureMatrix`	Returns a texture matrix.
`textureName`	Returns the texture name.
`texturePixelsHigh`	Returns the height of the texture representation.
`texturePixelsWide`	Returns the width of the texture representation.
`textureTarget`	Returns the texture target.
`unbindTextureRepresentationFromCGLContext:textureUnit:`	Unbinds the texture from a texture unit.

unlockBufferRepresentation	Releases the memory buffer representation of the image source.
unlockTextureRepresentation	Releases the OpenGL texture representation of the image source.

QCPlugInOutputImageProvider (New)

Complete reference information is available in the QCPlugInOutputImageProvider reference.

Instance Methods

canRenderWithCGLContext:	Returns whether the image data can be rendered into the provided CGL context.
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.
imageBounds	Returns the bounds of the image expressed in pixels and aligned to integer boundaries.
imageColorSpace	Returns the color space of the image or NULL if the image should not be color matched.
releaseRenderedTexture:forCGLContext:	Releases the previously copied texture.
renderToBuffer:withBytesPerRow: pixelFormat:forBounds:	Renders a subregion of the image into the supplied memory buffer using the specified pixel format.
renderWithCGLContext:forBounds:	Renders a subregion of the image to the provided CGL context.
shouldColorMatch	Returns whether the image should be color matched.
supportedBufferPixelFormats	Returns a list of pixel formats that are supported for rendering to a memory buffer.
supportedRenderedTexturePixelFormats	Returns a list of pixel formats that are supported for rendering to an onscreen OpenGL context.

C Symbols

All of the header files with new symbols are listed alphabetically, with their new symbols described.

ImageKit

IKFilterBrowserPanel.h

Data Types & Constants

All of the new data types and constants in this header file are listed alphabetically, with links to documentation and abstracts, if available.

`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.
`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.
`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.
`IKFilterBrowserFilterDoubleClickNotification`	Posted when the user double-clicks a filter in the filter browser.
`IKFilterBrowserFilterSelectedNotification`	Posted when the user clicks a filter name in the filter browser.
`IKFilterBrowserShowCategories`	The key for showing categories. The associated value is a BOOL value that determines if the filter browser should show the category list.
`IKFilterBrowserShowPreview`	The associated value is a BOOL value that determines if the filter browser should provide a preview.
`IKFilterBrowserWillPreviewFilterNotification`	Posted before showing a filter preview, allowing an application to set the parameters of a filter.

IKFilterUI.h

Data Types & Constants

All of the new data types and constants in this header file are listed alphabetically, with links to documentation and abstracts, if available.

`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.
`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.
`IKUISizeFlavor`	A key for the size of the controls in a filter view. The associated value can be IKUISizeMini, IKUISizeSmall, or IKUISizeRegular.
`IKUISizeMini`	Controls whose size is mini, as defined by Interface Builder 2.5.
`IKUISizeRegular`	Controls whose size is regular or normal, as defined by Interface Builder 2.5.
`IKUISizeSmall`	Controls whose size is small, as defined by Interface Builder 2.5.

IKImageBrowserView.h

Data Types & Constants

All of the new data types and constants in this header file are listed alphabetically, with links to documentation and abstracts, if available.

`IKCellsStyleNone`	No style.
`IKCellsStyleOutlined`	Cells are outlined.
`IKCellsStyleShadowed`	Cells use shadows.
`IKCellsStyleSubtitled`	Cells display a subtitle.
`IKCellsStyleTitled`	Cells display a title.
`IKGroupBezelStyle`	A bezel style.
`IKGroupDisclosureStyle`	A disclosure triangle.
`IKImageBrowserBackgroundColorKey`	A key for the background color of the image browser view. The associated value is an NSColor object.

IKImageBrowserCellsHighlightedTitleAttributesKey	A key for the highlighted title attribute for an item in the image browser view. The associated value is an NSDictionary object.
IKImageBrowserCellsOutlineColorKey	A key for the outline color for an item in the image browser view. The associated value is an NSColor object.
IKImageBrowserCellsSubtitleAttributesKey	A key for a subtitle attribute for an item in the image browser view. The associated value is an NSDictionary object.
IKImageBrowserCellsTitleAttributesKey	A key for title attribute of an item in the image browser view. The associated value is an NSDictionary object.
IKImageBrowserCGImageRepresentationType	A CGImageRef object.
IKImageBrowserCGImageSourceRepresentationType	A CGImageSourceRef object.
IKImageBrowserGroupBackgroundColorKey	A key for the background color of a group. The associated value is an NSColor object.
IKImageBrowserGroupRangeKey	A key for the range of a group. The associated value is an NSValue object.
IKImageBrowserGroupStyleKey	A key for the style of a group. The associated value is one of the constants defined in "Group Style Attributes".
IKImageBrowserGroupTitleKey	A key for the title of a group. The associated value is an NSString object.
IKImageBrowserIconRefPathRepresentationType	A path to an icon.
IKImageBrowserIconRefRepresentationType	An icon.
IKImageBrowserNSBitmapImageRepresentationType	An NSBitmapImageRep object.
IKImageBrowserNSDataRepresentationType	An NSData object.
IKImageBrowserNSImageRepresentationType	An NSImage object.
IKImageBrowserNSURLRepresentationType	An NSURLobject.
IKImageBrowserPathRepresentationType	A path representation (NSString).
IKImageBrowserQCCompositionPathRepresentationType	A path (NSString) or URL (NSURL) to a Quartz Composer composition.
IKImageBrowserQCCompositionRepresentationType	A QCComposition object.

IKImageBrowserQTMoviePathRepresentationType	A path (NSString) or URL (NSURL) to a QuickTime movie.
IKImageBrowserQTMovieRepresentationType	A QTMovie object.
IKImageBrowserQuickLookPathRepresentationType	A path (NSString) or URL (NSURL) to load data using QuickLook.
IKImageBrowserSelectionColorKey	A key for the color that indicates a selection. The associated value is an NSColor object.

IKImageView.h

Data Types & Constants

All of the new data types and constants in this header file are listed alphabetically, with links to documentation and abstracts, if available.

IKOverlayTypeBackground	A background.
IKOverlayTypeImage	An image.
IKToolModeAnnotate	The annotation tool.
IKToolModeCrop	The crop tool.
IKToolModeMove	The move tool.
IKToolModeNone	
IKToolModeRotate	The rotation tool.
IKToolModeSelect	The selection tool.

IKPictureTaker.h

Data Types & Constants

All of the new data types and constants in this header file are listed alphabetically, with links to documentation and abstracts, if available.

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

`IKPictureTakerAllowsVideoCaptureKey`	A key for allowing video capture. The associated value is an NSNumber value (BOOL) whose default value is YES.
`IKPictureTakerCropAreaSizeKey`	A key for the cropping area size. The associated value is an NSValue object (NSSize).
`IKPictureTakerImageTransformsKey`	A n image transformation key. The associated value is an NSDictionary object that can be serialized.
`IKPictureTakerInformationalTextKey`	A key for informational text. The associated value is an NSString or NSAttributedString object whose default value is "Drag Image Here".
`IKPictureTakerOutputImageMaxSizeKey`	A key for the maximum size of the output image. The associated value is an NSValue object (NSSize).
`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.
`IKPictureTakerShowEffectsKey`	A key for showing effects. The associated value is an NSNumber object that contains a BOOL value whose default value is NO.
`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."
`IKPictureTakerShowRecentPictureKey`	
`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.

IKSlideshow.h

Data Types & Constants

All of the new data types and constants in this header file are listed alphabetically, with links to documentation and abstracts, if available.

`IK_iPhotoBundleIdentifier`	The iPhoto application—com.apple.iPhoto.
`IKSlideshowModeImages`	All items in the slideshow are images.

IKSlideshowModeOther	There are a mixture of items in the slideshow (image, PDF, text, HTML, and so on).
IKSlideshowModePDF	All items in the slideshow are PDF documents.
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.
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.
IKSlideshowPDFDisplaysAsBook	A key for displaying the slideshow as a book. The associated value is a Boolean data type.
IKSlideshowStartIndex	A key for the slideshow item index. The associated value is an index.
IKSlideshowStartPaused	A key for starting in a paused state. The associated value is a Boolean data type.
IKSlideshowWrapAround	A key for starting the slideshow over after the last slide shows. The associated value is a Boolean data type.

ImageKitDeprecated.h

Data Types & Constants

All of the new data types and constants in this header file are listed alphabetically, with links to documentation and abstracts, if available.

IKImagePickerAllowsEditingKey	
IKImagePickerAllowsFileChoosingKey	
IKImagePickerAllowsVideoCaptureKey	
IKImagePickerCropAreaSizeKey	
IKImagePickerImageTransformsKey	
IKImagePickerInformationalTextKey	
IKImagePickerOutputImageMaxSizeKey	
IKImagePickerShowEffectsKey	
IKImagePickerShowRecentPictureKey	
IKImagePickerUpdateRecentPictureKey	

IKPictureTakerShowAddressBookPicture	
IKPictureTakerShowEmptyPicture	

PDFKit

PDFActionNamed.h

Data Types & Constants

All of the new data types and constants in this header file are listed alphabetically, with links to documentation and abstracts, if available.

kPDFActionNamedFind	The Find action.
kPDFActionNamedFirstPage	The First Page action.
kPDFActionNamedGoBack	The Go Back action.
kPDFActionNamedGoForward	The Go Forward action.
kPDFActionNamedGoToPage	The Go to Page action.
kPDFActionNamedLastPage	The Last Page action.
kPDFActionNamedNextPage	The Next Page action.
kPDFActionNamedNone	The action has no name.
kPDFActionNamedPreviousPage	The Previous Page action.
kPDFActionNamedPrint	The Print action.
kPDFActionNamedZoomIn	The Zoom In action.
kPDFActionNamedZoomOut	The Zoom Out action.
PDFActionNamedName	

PDFAnnotationMarkup.h

Data Types & Constants

All of the new data types and constants in this header file are listed alphabetically, with links to documentation and abstracts, if available.

kPDFMarkupTypeHighlight	Highlight style for the markup.
kPDFMarkupTypeStrikeOut	Strikethrough style for the markup.

kPDFMarkupTypeUnderline	Underline style for the markup.
PDFMarkupType	

PDFDestination.h

Data Types & Constants

All of the new data types and constants in this header file are listed alphabetically, with links to documentation and abstracts, if available.

kPDFDestinationUnspecifiedValue	Unspecified value used when a destination's actual x or y value is unimportant.

PDFDocument.h

Data Types & Constants

All of the new data types and constants in this header file are listed alphabetically, with links to documentation and abstracts, if available.

kPDFPrintPageScaleDownToFit	Scale large pages down to fit the paper size (smaller pages do not get scaled up).
kPDFPrintPageScaleNone	Do not apply scaling to the page when printing.
kPDFPrintPageScaleToFit	Scale each page up or down to best fit the paper size.
PDFDocumentDidBeginPageWriteNotification	Posted each time a write operation begins working on a page in a document.
PDFDocumentDidBeginWriteNotification	Posted each time a write operation begins working on a document.
PDFDocumentDidEndPageWriteNotification	Posted each time a write operation finishes working on a page in a document.
PDFDocumentDidEndWriteNotification	Posted each time a write operation finishes working on a document.
PDFPrintScalingMode	The type of scaling to be used when printing a page (see "PDF Page Scaling Modes for Printing").

PDFView.h

Data Types & Constants

All of the new data types and constants in this header file are listed alphabetically, with links to documentation and abstracts, if available.

`kPDFIconArea`	The mouse is over an icon.
`PDFViewAnnotationWillHitNotification`	Posted before the user clicks an annotation.
`PDFViewDisplayBoxChangedNotification`	Posted when the display box has changed.
`PDFViewDisplayModeChangedNotification`	Posted when the display mode has changed.
`PDFViewSelectionChangedNotification`	Posted when the current selection has changed.

QuartzComposer

QCComposition.h

Data Types & Constants

All of the new data types and constants in this header file are listed alphabetically, with links to documentation and abstracts, if available.

`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).
`QCCompositionAttributeCategoryKey`	The composition category. The associated value is a category constant. See “Composition Categories.”
`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.
`QCCompositionAttributeIsTimeDependentKey`	
`QCCompositionCategoryDistortion`	A composition that produces a distortion effect.
`QCCompositionCategoryStylize`	A composition that produces a stylize effect.
`QCCompositionCategoryUtility`	A utility composition.
`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.

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.
QCCompositionInputDestinationImageKey	An image input port whose key is inputDestinationImage.
QCCompositionInputImageKey	An image input port whose key is inputImage.
QCCompositionInputPaceKey	A number input port whose key is inputPace. The value must be in the [0,1] range.
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.
QCCompositionInputPrimaryColorKey	A color input port whose key is inputPrimaryColor.
QCCompositionInputRSSArticleDurationKey	A number input port whose key is inputRSSArticleDuration. The value must be expressed in seconds.
QCCompositionInputRSSFeedURLKey	A string input port whose key is inputRSSFeedURL. This port must be passed an http or feed scheme URL.
QCCompositionInputScreenImageKey	An image input port whose key is inputScreenImage.
QCCompositionInputSecondaryColorKey	A color input port whose key is inputSecondaryColor.
QCCompositionInputSourceImageKey	An image input port whose key is inputSourceImage.
QCCompositionInputTrackInfoKey	A structure input port whose key is inputTrackInfo. The structure contains optional entries, such as “name”, “artist”, "album", "duration", "artwork", and so on.
QCCompositionInputTrackPositionKey	A number input port whose key is inputTrackPosition. The value must be expressed in seconds.
QCCompositionInputTrackSignalKey	A Boolean input port whose key is inputTrackSignal.
QCCompositionInputXKey	A number input port whose key is inputX. The value must be normalized to the image width with the origin on the left.

`QCCompositionInputYKey`	A number input port whose key is inputY. The value must be normalized to the image height with the origin at the bottom.
`QCCompositionOutputImageKey`	An image output port whose key is outputImage.
`QCCompositionOutputWebPageURLKey`	A string output port whose key is outputWebPageURL.
`QCCompositionProtocolGraphicAnimation`	A composition that renders a generic graphical animation. It has the option to use QCCompositionInputPrimaryColorKey for the primary color of the animation, QCCompositionInputSecondaryColorKey for the secondary color of the animation, QCCompositionInputPaceKey for the global pace of the animation, and QCCompositionInputPreviewModeKey to indicate if the animation should run in lower-quality for preview purposes.
`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 for the starting image and QCCompositionInputDestinationImageKey for the image to transition to. The composition can optionally use QCCompositionInputPreviewModeKey to indicate if the animation should run in lower-quality for preview purposes.
`QCCompositionProtocolImageFilter`	A composition that applies an effect to a source image. A conforming composition must use the input key QCCompositionInputImageKey for the source image and QCCompositionOutputImageKey for the output image. The composition can optionally use QCCompositionInputXKey to specify the X position of the center point of the effect, QCCompositionInputYKey to specify the Y position of the center point of the effect, andQCCompositionInputPreviewModeKey to indicate if the animation should run in lower-quality for preview purposes.

QCCompositionProtocolMusicVisualizer	A composition that acts as a visualizer for music. A conforming composition must use the input key QCCompositionInputAudioPeakKey for the instantaneous audio peak and the QCCompositionInputAudioSpectrumKey for the instantaneous audio spectrum. It can optionally use the QCCompositionInputTrackInfoKey to indicate it receives information about the current track and the QCCompositionInputTrackSignalKey to indicate the start of a new track.
QCCompositionProtocolRSSVisualizer	A composition that acts as a visualizer for an RSS feed. A conforming composition must use the input key QCCompositionInputRSSFeedURLKey for the URL to use for the RSS feed. It can optionally use QCCompositionInputRSSArticleDurationKey to specify the duration of each feed article.
QCCompositionProtocolScreenSaver	A composition that can be used as a screen saver. The composition has the option to use QCCompositionInputScreenImageKey for a screenshot image of the screen that the screen saver runs on, QCCompositionInputPreviewModeKey to indicate if the animation should run in lower-quality for preview purposes, and QCCompositionOutputWebPageURLKey for a URL to open in the default web browser when screen saver exits (only allowed if screen saver password is disabled).

QCCompositionPickerPanel.h

Data Types & Constants

All of the new data types and constants in this header file are listed alphabetically, with links to documentation and abstracts, if available.

QCCompositionPickerPanelDidSelect-CompositionNotification	Posted when the user chooses a composition.

QCCompositionPickerView.h

Data Types & Constants

All of the new data types and constants in this header file are listed alphabetically, with links to documentation and abstracts, if available.

QCCompositionPickerViewDidSelect-CompositionNotification	Posted when the user selects a composition in the picker view.

QCCompositionRepository.h

Data Types & Constants

All of the new data types and constants in this header file are listed alphabetically, with links to documentation and abstracts, if available.

`QCCompositionRepositoryDidUpdateNotification`	Posted whenever the list of compositions in the composition repository is updated.

QCPlugIn.h

Data Types & Constants

All of the new data types and constants in this header file are listed alphabetically, with links to documentation and abstracts, if available.

`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.)
`kQCPlugInExecutionModeProcessor`	A processor execution mode. The custom patch executes whenever its inputs change or if the time change (assuming it's time-dependent).
`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.
`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.]]
`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.)
`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.)
`QCPlugInAttributeCopyrightKey`	

`QCPlugInAttributeDescriptionKey`	The key for the custom patch description. The associated value is an NSString object.
`QCPlugInAttributeNameKey`	The key for the custom patch name. The associated value is an NSString object.
`QCPlugInBufferReleaseCallback`	
`QCPlugInExecutionArgumentEventKey`	The current NSEvent if available.
`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).
`QCPlugInExecutionMode`	Execution modes for custom patches.
`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.
`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.
`QCPlugInPixelFormatI8`	An I8 format. Intensity information is represented as an 8-bit value.
`QCPlugInPixelFormatIf`	An If format. Intensity information is represented as a floating-point value.
`QCPlugInPixelFormatRGBAf`	An RGBAf format. Pixel components are represented as floating-point values.
`QCPlugInTextureReleaseCallback`	
`QCPlugInTimeMode`	Time modes for custom patches.
`QCPortAttributeDefaultValueKey`	The key for the port default value. You can use this key only for value ports (Boolean, Index, Number, Color and String).

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.

QuartzFilters

QuartzFilterManager.h

Data Types & Constants

All of the new data types and constants in this header file are listed alphabetically, with links to documentation and abstracts, if available.

kQuartzFilterApplicationDomain	
kQuartzFilterManagerDidAddFilterNotification	
kQuartzFilterManagerDidModifyFilterNotification	
kQuartzFilterManagerDidRemoveFilterNotification	
kQuartzFilterManagerDidSelectFilterNotification	
kQuartzFilterPDFWorkflowDomain	
kQuartzFilterPrintingDomain	

10.4 Symbol Changes

This article lists the symbols added to Quartz.framework in Mac OS X v10.4.

Classes

All of the classes with new symbols are listed alphabetically, with their new class, instance, and delegate methods described.

NSObject

Complete reference information is available in the NSObject reference.

Instance Methods

didMatchString:	Called for every match found during a find operation.
documentDidBeginDocumentFind:	Called when the PDFDocumentDidBeginFindNotification notification is posted.
documentDidBeginPageFind:	Called when the PDFDocumentDidBeginPageFindNotification notification is posted.
documentDidEndDocumentFind:	Called when the PDFDocumentDidEndFindNotification notification is posted.
documentDidEndPageFind:	Called when the PDFDocumentDidEndPageFindNotification notification is posted.
documentDidFindMatch:	Called when the PDFDocumentDidFindMatchNotification notification is posted.
documentDidUnlock:	Called when the PDFDocumentDidUnlockNotification notification is posted.

PDFAnnotation (New)

Complete reference information is available in the PDFAnnotation reference.

Instance Methods

`border`	Returns the border style for the annotation.
`bounds`	Returns the bounding box for the annotation in page space.
`color`	Returns the stroke color for the annotation.
`contents`	Returns the textual content (if any) associated with the annotation.
`drawWithBox:`	Draws the annotation on its associated page.
`hasAppearanceStream`	Returns a Boolean value that indicates whether the annotation has an appearance stream associated with it.
`initWithBounds:`	Initializes a PDF annotation object.
`page`	Returns the page that the annotation is associated with.
`setBorder:`	Sets the border style for the annotation.
`setBounds:`	Sets the bounding box for the annotation.
`setColor:`	Sets the stroke color for the annotation.
`setContents:`	Specifies the textual content associated with the annotation.
`setShouldDisplay:`	Specifies whether the annotation should be displayed.
`setShouldPrint:`	Specifies whether the annotation should appear when the document is printed.
`shouldDisplay`	Returns a Boolean value indicating whether the annotation should be displayed.
`shouldPrint`	Returns a Boolean value indicating whether the annotation should appear when the document is printed.
`toolTip`	Returns text for display as a help tag.
`type`	Returns the type of the annotation.

PDFAnnotationButtonWidget (New)

Complete reference information is available in the `PDFAnnotationButtonWidget` reference.

Instance Methods

`backgroundColor`	Returns the background color of the control.
`controlType`	Returns the type of the control.

`parentID`	Gets the ID of the parent object.
`setBackgroundColor:`	Sets the control's background color.
`setHighlighted:`	Sets the control's highlighting when it is drawn.
`setState:`	Sets the state of the control.
`state`	Returns the state of the control.

PDFAnnotationCircle (New)

Complete reference information is available in the `PDFAnnotationCircle` reference.

Instance Methods

`interiorColor`	Returns the fill color used for drawing the annotation.
`setInteriorColor:`	Sets the fill color used for drawing the annotation.

PDFAnnotationFreeText (New)

Complete reference information is available in the `PDFAnnotationFreeText` reference.

Instance Methods

`alignment`	Returns the horizontal alignment of text within the bounds of the annotation.
`font`	Returns the font used for the annotation's text field.
`setAlignment:`	Sets the horizontal alignment of text within the bounds of the annotation.
`setFont:`	Sets the font used in the text field of the annotation.

PDFAnnotationInk (New)

Complete reference information is available in the `PDFAnnotationInk` reference.

Instance Methods

`addBezierPath:`	Adds a Bezier path to an annotation.
`paths`	Returns an array containing the Bezier paths that make up an annotation.

removeBezierPath:	Removes a Bezier path from an annotation.

PDFAnnotationLine (New)

Complete reference information is available in the PDFAnnotationLine reference.

Instance Methods

endLineStyle	Returns the line ending style for the ending point of the line.
endPoint	Returns the ending point for the line in page space.
setEndLineStyle:	Sets the line ending style for the ending point of the line.
setEndPoint:	Sets the ending point for the line.
setStartLineStyle:	Sets the line ending style for the starting point of the line.
setStartPoint:	Sets the starting point for the line.
startLineStyle	Returns the line ending style for the starting point of the line.
startPoint	Returns the starting point for the line.

PDFAnnotationLink (New)

Complete reference information is available in the PDFAnnotationLink reference.

Instance Methods

destination	Gets the destination for the link when the destination was specified as a PDFDestination object.
setDestination:	Sets the destination for the link as a PDFDestination object.
setHighlighted:	Sets the highlighting state for the link.
setURL:	Sets the destination for the link as a URL.
URL	Gets the destination for the link when the destination was specified as a URL.

PDFAnnotationMarkup (New)

Complete reference information is available in the PDFAnnotationMarkup reference.

Instance Methods

`markupType`	Gets the markup style.
`quadrilateralPoints`	Gets the array of quadrilateral points defining the bounds of the markup.
`setMarkupType:`	Sets the markup style.
`setQuadrilateralPoints:`	Sets the array of quadrilateral points defining the bounds of the markup.

PDFAnnotationSquare (New)

Complete reference information is available in the `PDFAnnotationSquare` reference.

Instance Methods

`interiorColor`	Gets the fill color used for drawing the annotation.
`setInteriorColor:`	Sets the fill color used for drawing the annotation.

PDFAnnotationText (New)

Complete reference information is available in the `PDFAnnotationText` reference.

Instance Methods

`iconType`	Returns the icon type for the annotation.
`setIconType:`	Sets the icon type for the annotation.
`setWindowIsOpen:`	Sets the open/closed state of the annotation to the specified value.
`windowIsOpen`	Returns a Boolean value indicating whether the annotation's note window is open.

PDFAnnotationTextWidget (New)

Complete reference information is available in the `PDFAnnotationTextWidget` reference.

Instance Methods

`alignment`	Returns the text alignment setting for the annotation.

`font`	Returns the font used for the annotation's text field.
`setAlignment:`	Sets the text alignment for the annotation.
`setFont:`	Sets the font used in the text field of the annotation.
`setStringValue:`	Sets the string for the annotation.
`stringValue`	Returns the string assigned to the annotation.

PDFBorder (New)

Complete reference information is available in the `PDFBorder` reference.

Instance Methods

`dashPattern`	Gets the dash pattern for the border as an array of NSNumber objects.
`drawInRect:`	Draws the border.
`horizontalCornerRadius`	Gets the horizontal corner radius (in points) used for a rounded-rectangle border.
`lineWidth`	Gets the line width for the border, in points.
`setDashPattern:`	Sets the dash pattern for the border.
`setHorizontalCornerRadius:`	Sets the horizontal corner radius (in points) used for a rounded-rectangle border.
`setLineWidth:`	Sets the line width (in points) for the border.
`setStyle:`	Sets the border style.
`setVerticalCornerRadius:`	Sets the vertical corner radius (in points) used for a rounded-rectangle border.
`style`	Gets the border style.
`verticalCornerRadius`	Gets the vertical corner radius used for a rounded-rectangle border, in points.

PDFDestination (New)

Complete reference information is available in the `PDFDestination` reference.

Instance Methods

`initWithPage:atPoint:`	Initializes the destination.
`page`	Returns the page that the destination refers to.
`point`	Returns the point, in page space, that the destination refers to.

PDFDocument (New)

Complete reference information is available in the `PDFDocument` reference.

Instance Methods

`allowsCopying`	Returns a Boolean value indicating whether the document allows copying of content to the Pasteboard.
`allowsPrinting`	Returns a Boolean value indicating whether the document allows printing.
`beginFindString:withOptions:`	Asynchronously finds all instances of the specified string in the document.
`cancelFindString`	Cancels a search initiated with beginFindString:withOptions:.
`dataRepresentation`	Returns a representation of the document as an NSData object.
`delegate`	Returns the object acting as the delegate for the PDFDocument object.
`documentAttributes`	Returns a dictionary of document metadata.
`documentURL`	Returns the URL for the document.
`exchangePageAtIndex:withPageAtIndex:`	Swaps one page with another.
`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.
`findString:withOptions:`	Synchronously finds all instances of the specified string in the document.
`indexForPage:`	Gets the index number for the specified page.
`initWithData:`	Initializes a PDFDocument object with the passed-in data.

initWithURL:	Initializes a PDFDocument object with the contents at the specified URL (if the URL is invalid, this method returns NULL).
insertPage:atIndex:	Inserts a page at the specified index point.
isEncrypted	Returns a Boolean value specifying whether the document is encrypted.
isFinding	Returns a Boolean value indicating whether an asynchronous find operation is in progress.
isLocked	Returns a Boolean value indicating whether the document is locked.
majorVersion	Returns the major version of the document.
minorVersion	Returns the minor version of the document.
outlineItemForSelection:	Returns the most likely parent PDF outline object for the selection.
outlineRoot	Returns the root PDF outline object for the document.
pageAtIndex:	Returns the page at the specified index number.
pageCount	Returns the number of pages in the document.
removePageAtIndex:	Removes the page at the specified index point.
selectionForEntireDocument	Returns a selection representing the textual content of the entire document.
selectionFromPage:atCharacterIndex: toPage:atCharacterIndex:	Returns the specified selection based on starting and ending character indexes.
selectionFromPage:atPoint:toPage:atPoint:	Returns the specified selection based on starting and ending points.
setDelegate:	Establishes the specified object as the delegate for the PDFDocument object.
setDocumentAttributes:	Sets the document attributes.
string	Returns a string representing the textual content for the entire document.
unlockWithPassword:	Attempts to unlock an encrypted document.
writeToFile:	Writes the document to a file at the specified path.
writeToFile:withOptions:	Writes the document to a file at the specified path with the specified options.

writeToURL:	Writes the document to a location specified by the passed-in URL.
writeToURL:withOptions:	Writes the document to the specified URL with the specified options.

PDFOutline (New)

Complete reference information is available in the PDFOutline reference.

Instance Methods

childAtIndex:	Returns the child outline object at the specified index.
destination	Returns the destination associated with the outline.
document	Returns the document with which the outline is associated.
initWithDocument:	Initializes an outline with the specified PDF document.
label	Returns the label for the outline.
numberOfChildren	Returns the number of child outline objects in the outline.

PDFPage (New)

Complete reference information is available in the PDFPage reference.

Instance Methods

addAnnotation:	Adds the specified annotation object to the page.
annotationAtPoint:	Returns the annotation, if there is one, at the specified point.
annotations	Returns an array containing the page's annotations.
attributedString	Returns an NSAttributedString object representing the text on the page.
boundsForBox:	Returns the bounds for the specified PDF display box.
characterBoundsAtIndex:	Returns the bounds, in page space, of the character at the specified index.
characterIndexAtPoint:	Returns the character index value for the specified point in page space.

`dataRepresentation`	Returns the PDF data (that is, a PDF document) representing this page. This method does not preserve external page links.
`displaysAnnotations`	Returns a Boolean value indicating whether annotations are displayed for the page.
`document`	Returns the PDFDocument object with which the page is associated.
`drawWithBox:`	Draws the page within the specified box.
`initWithDocument:`	Initializer for subclasses of PDFPage.
`label`	Returns the label for the page.
`numberOfCharacters`	Returns the number of characters on the page, including whitespace characters.
`removeAnnotation:`	Removes the specified annotation from the page.
`rotation`	Returns the page rotation angle in degrees.
`selectionForLineAtPoint:`	Returns the whole line of text that includes the specified point.
`selectionForRange:`	Returns the text contained within the specified range.
`selectionForRect:`	Returns the text enclosed within the specified rectangle, expressed in page (user) coordinates.
`selectionForWordAtPoint:`	Returns the whole word that includes the specified point.
`selectionFromPoint:toPoint:`	Returns the text between the two specified points in page space.
`setBounds:forBox:`	Sets the bounds for the specified box.
`setDisplaysAnnotations:`	Specifies whether or not to display annotations for the page.
`setRotation:`	Sets the rotation angle for the page in degrees.
`string`	Returns an NSString object representing the text on the page.

PDFSelection (New)

Complete reference information is available in the `PDFSelection` reference.

Instance Methods

`addSelection:`	Adds the specified selection to the receiving selection.
`attributedString`	Returns an NSAttributedString object representing the text contained in the selection (may contain linefeed characters).
`boundsForPage:`	Returns the bounds of the selection on the specified page.

`drawForPage:active:`	Calls drawForPage:withBox:active: with a default value for box parameter.
`drawForPage:withBox:active:`	Draws the selection relative to the origin of the specified box in page space.
`extendSelectionAtEnd:`	Extends the selection from its end toward the end of the document.
`extendSelectionAtStart:`	Extends the selection from its start toward the beginning of the document.
`pages`	Returns the array of pages contained in the selection.
`string`	Returns an NSString object representing the text contained in the selection (may contain linefeed characters).

PDFView (New)

Complete reference information is available in the `PDFView` reference.

Instance Methods

`allowsDragging`	Determines whether the view can accept new PDF documents dragged into it by the user.
`areaOfInterestForMouse:`	Returns the type of area the mouse cursor is over.
`autoScales`	Returns a Boolean value indicating whether autoscaling is set.
`backgroundColor`	Returns the view's background color.
`canGoBack`	Returns a Boolean value indicating whether the user can navigate to the previous page in the page history.
`canGoForward`	Returns a Boolean value indicating whether the user can navigate to the next page in the page history.
`canGoToFirstPage`	Returns a Boolean value indicating whether the user can navigate to the first page of the document.
`canGoToLastPage`	Returns a Boolean value indicating whether the user can navigate to the last page of the document.
`canGoToNextPage`	Returns a Boolean value indicating whether the user can navigate to the next page of the document.
`canGoToPreviousPage`	Returns a Boolean value indicating whether the user can navigate to the previous page of the document.
`canZoomIn`	Returns a Boolean value indicating whether the user can magnify the view—that is, zoom in.

canZoomOut	Returns a Boolean value indicating whether the user can view an expanded area—that is, zoom out.
clearSelection	Clears the selection.
convertPoint:fromPage:	Converts a point from page space to view space.
convertPoint:toPage:	Converts a point from view space to page space.
convertRect:fromPage:	Converts a rectangle from page space to view space.
convertRect:toPage:	Converts a rectangle from view space to page space.
copy:	Copies the text in the selection, if any, to the Pasteboard.
currentDestination	Returns a PDFDestination object representing the current page and the current point in the view specified in page space.
currentPage	Returns the current page.
currentSelection	Returns the current selection.
delegate	Returns the view's delegate.
displayBox	Returns the current style of display box.
displayMode	Returns the current display mode.
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).
displaysPageBreaks	Returns a Boolean value indicating whether the view is displaying page breaks.
document	Returns the document associated with a PDFView object.
documentView	Returns the innermost view used by PDFView or by your PDFView subclass.
drawPage:	For use by subclasses of PDFView for custom rendering of pages.
goBack:	Navigates back one step in the page history.
goForward:	Navigates forward one step in the page history.
goToDestination:	Navigates to the specified destination.
goToFirstPage:	Navigates to the first page of the document.
goToLastPage:	Navigates to the last page of the document.
goToNextPage:	Navigates to the next page of the document.
goToPage:	Scrolls to the specified page.

`goToPreviousPage:`	Navigates to the previous page of the document.
`goToSelection:`	Scrolls to the first character of the specified selection.
`greekingThreshold`	Returns the current greeking threshold for the view.
`layoutDocumentView`	Performs layout of the inner views.
`pageForPoint:nearest:`	Returns the page containing a point specified in view coordinates.
`printWithInfo:autoRotate:`	Prints the document with the specified printer information.
`rowSizeForPage:`	Returns the size needed to display a row of the current document page.
`scaleFactor`	Returns the current scale factor for the view.
`scrollSelectionToVisible:`	Scrolls the view until the selection is visible.
`selectAll:`	Selects all text in the document.
`setAllowsDragging:`	Specifies whether the view can accept drags.
`setAutoScales:`	Toggles whether the scaling factor applied to a view automatically responds to resizing.
`setBackgroundColor:`	Sets the view’s background color.
`setCurrentSelection:`	Sets the selection.
`setCursorForAreaOfInterest:`	Sets the type of mouse cursor according to the type of area the mouse cursor is over.
`setDelegate:`	Sets a delegate for the view.
`setDisplayBox:`	Specifies the box to display and to clip to.
`setDisplayMode:`	Sets the display mode for the view.
`setDisplaysAsBook:`	Specifies whether the view should treat the document’s first page as a book cover.
`setDisplaysPageBreaks:`	Toggles the display of page breaks.
`setDocument:`	Associates a document with a PDFView object.
`setGreekingThreshold:`	Sets the greeking threshold to use for displaying text.
`setScaleFactor:`	Sets the scale factor for the view.
`setShouldAntiAlias:`	Specifies whether to use anti-aliasing in the view.
`shouldAntiAlias`	Returns a Boolean value indicating whether the view is anti-aliased.
`takeBackgroundColorFrom:`	Sets the view’s background color to the specified color.

takePasswordFrom:	A convenience method that calls – [[self document] setpassword:] with the password from the specified sender.
zoomIn:	Zooms in by increasing the scaling factor.
zoomOut:	Zooms out by decreasing the scaling factor.

Delegate Methods

PDFViewWillChangeScaleFactor:toScale:	Delegate method for overriding changes to scale factor.
PDFViewWillClickOnLink:withURL:	Delegate method for handling clicks on URL links in a view.

QCRenderer (New)

Complete reference information is available in the `QCRenderer` reference.

Instance Methods

attributes	Returns the attributes of the composition in the view.
initWithOpenGLContext:pixelFormat:file:	Creates a renderer object with an NSOpenGLContext object and a composition file.
inputKeys	Returns an array that contains the keys that identify the input ports of the root patch of the composition.
outputKeys	Returns an array that contains the keys that identify the output ports of the root patch of the composition.
renderAtTime:arguments:	Renders a frame of a composition at the specified time.
setValue:forInputKey:	Sets the value for an input port of a composition.
valueForInputKey:	Returns the value for an input port of a composition.
valueForOutputKey:	Returns the value for an output port of a composition.

QCView (New)

Complete reference information is available in the `QCView` reference.

Instance Methods

`attributes`	Returns the attributes of the composition in the view.
`autostartsRendering`	Checks whether the view is set to start rendering automatically.
`erase`	Clears the view using the current erase color.
`eraseColor`	Retrieves the current color used to erase the view.
`eventForwardingMask`	Retrieves the mask used to filter which types of events are forwarded from the view to the composition during rendering.
`inputKeys`	Returns an array that contains the keys that identify the input ports of the root patch of the composition.
`isRendering`	Checks whether a composition is rendering in the view.
`loadCompositionFromFile:`	Loads the composition file located at the specified path.
`maxRenderingFrameRate`	Returns the maximum frame rate for rendering.
`outputKeys`	Returns an array that contains the keys that identify the output ports of the root patch of the composition.
`setAutostartsRendering:`	Sets whether the composition that is in the view starts rendering automatically when the view is put on the screen.
`setEraseColor:`	Sets the color used to erase the view.
`setEventForwardingMask:`	Sets the mask used to filter which types of events are forwarded from the view to the composition during rendering.
`setMaxRenderingFrameRate:`	Sets the maximum rendering frame rate.
`setValue:forInputKey:`	Sets the value for an input port of the composition in the view.
`start:`	Starts rendering a composition in a view.
`startRendering`	Starts rendering the composition that is in the view.
`stop:`	Stops rendering a composition in a view.
`stopRendering`	Stops rendering the composition that is in the view.
`valueForInputKey:`	Returns the value for an input port of the composition in the view.
`valueForOutputKey:`	Returns the value for an output port of the composition in the view.

C Symbols

All of the header files with new symbols are listed alphabetically, with their new symbols described.

PDFKit

PDFAnnotationButtonWidget.h

Data Types & Constants

All of the new data types and constants in this header file are listed alphabetically, with links to documentation and abstracts, if available.

`kPDFWidgetCheckBoxControl`	Check box control.
`kPDFWidgetPushButtonControl`	Push button control.
`kPDFWidgetRadioButtonControl`	Radio button control.
`kPDFWidgetUnknownControl`	Unknown control type.
`PDFWidgetControlType`	The types of annotation buttons.

PDFAnnotationLine.h

Data Types & Constants

All of the new data types and constants in this header file are listed alphabetically, with links to documentation and abstracts, if available.

`kPDFLineStyleCircle`	A circular line ending filled with the annotation's interior color, if any.
`kPDFLineStyleClosedArrow`	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.
`kPDFLineStyleDiamond`	A diamond-shaped line ending filled with the annotation's interior color, if any.
`kPDFLineStyleNone`	No line ending.
`kPDFLineStyleOpenArrow`	An open arrowhead line ending, composed from two short lines meeting in an acute angle at the line end.
`kPDFLineStyleSquare`	A square line ending filled with the annotation's interior color, if any.
`PDFLineStyle`	

PDFAnnotationMarkup.h

Data Types & Constants

All of the new data types and constants in this header file are listed alphabetically, with links to documentation and abstracts, if available.

kPDFMarkupTypeHighlight	Highlight style for the markup.
kPDFMarkupTypeStrikeOut	Strikethrough style for the markup.
kPDFMarkupTypeUnderline	Underline style for the markup.

PDFAnnotationText.h

Data Types & Constants

All of the new data types and constants in this header file are listed alphabetically, with links to documentation and abstracts, if available.

kPDFTextAnnotationIconComment	Comment annotation icon.
kPDFTextAnnotationIconHelp	Help annotation icon.
kPDFTextAnnotationIconInsert	Insert annotation icon.
kPDFTextAnnotationIconKey	Key annotation icon.
kPDFTextAnnotationIconNewParagraph	New Paragraph annotation icon.
kPDFTextAnnotationIconNote	Note annotation icon.
kPDFTextAnnotationIconParagraph	Paragraph annotation icon.
PDFTextAnnotationIconType	

PDFBorder.h

Data Types & Constants

All of the new data types and constants in this header file are listed alphabetically, with links to documentation and abstracts, if available.

kPDFBorderStyleBeveled	Beveled border.
kPDFBorderStyleDashed	Dashed border.
kPDFBorderStyleInset	Inset border.
kPDFBorderStyleSolid	Solid border.

kPDFBorderStyleUnderline	Underline border.
PDFBorderStyle	

PDFDocument.h

Data Types & Constants

All of the new data types and constants in this header file are listed alphabetically, with links to documentation and abstracts, if available.

PDFDocumentAuthorAttribute	An optional text string (an NSString) containing the name of the author of the document.
PDFDocumentCreationDateAttribute	An optional text string (an NSDate) containing the document's creation date.
PDFDocumentCreatorAttribute	An optional text string (an NSString) containing the name of the application that created the document content.
PDFDocumentDidBeginFindNotification	Posted when the beginFindString:withOptions: or findString:withOptions: method begins finding.
PDFDocumentDidBeginPageFindNotification	Posted each time a find operation begins working on a new page of a document.
PDFDocumentDidEndFindNotification	Posted when the beginFindString:withOptions: or findString:withOptions: method returns.
PDFDocumentDidEndPageFindNotification	Posted each time a find operation finishes working on a page in a document.
PDFDocumentDidFindMatchNotification	Posted each time a string match is found in a document.
PDFDocumentDidUnlockNotification	Posted when a document unlocks after a unlockWithPassword: message.
PDFDocumentKeywordsAttribute	An optional array of text strings (an NSArray of NSString objects) containing keywords for the document.
PDFDocumentModificationDateAttribute	An optional text string (an NSDate) containing the document's last-modified date.
PDFDocumentProducerAttribute	An optional text string (an NSString) containing the name of the application that produced the PDF data for the document.
PDFDocumentSubjectAttribute	An optional text string (an NSString) containing a description of the subject of the document.

`PDFDocumentTitleAttribute`	An optional text string (an NSString) containing the title of the document.

PDFPage.h

Data Types & Constants

All of the new data types and constants in this header file are listed alphabetically, with links to documentation and abstracts, if available.

`kPDFDisplayBoxArtBox`	A rectangle defining the boundaries of the page's meaningful content including surrounding white space intended for display. Default value equal to kPDFDisplayBoxCropBox.
`kPDFDisplayBoxBleedBox`	A rectangle defining the boundaries of the clip region for the page contents in a production environment. Default value equal to kPDFDisplayBoxCropBox.
`kPDFDisplayBoxCropBox`	A rectangle defining the boundaries of the visible region , expressed in default user-space units. Default value equal to kPDFDisplayBoxMediaBox.
`kPDFDisplayBoxMediaBox`	A rectangle defining the boundaries of the physical medium for display or printing, expressed in default user-space units.
`kPDFDisplayBoxTrimBox`	A rectangle defining the intended boundaries of the finished page. Default value equal to kPDFDisplayBoxCropBox.
`PDFDisplayBox`	

PDFView.h

Data Types & Constants

All of the new data types and constants in this header file are listed alphabetically, with links to documentation and abstracts, if available.

`kPDFAnnotationArea`	The mouse is over an annotation.
`kPDFControlArea`	The mouse is over a control.
`kPDFDisplaySinglePage`	The document displays one page at a time horizontally and vertically. Vertical and horizontal scrolling apply only to the current page.
`kPDFDisplaySinglePageContinuous`	The document displays in continuous mode vertically, with single-page width horizontally. Vertical scrolling applies to the entire document.
`kPDFDisplayTwoUp`	The document displays two pages side-by-side. Vertical and horizontal scrolling apply only to the pair of displayed pages

kPDFDisplayTwoUpContinuous	The document displays in continuous mode vertically and displays two pages side-by-side horizontally. Vertical scrolling applies to the entire document.
kPDFLinkArea	The mouse is over a link.
kPDFNoArea	The mouse is over an undefined area.
kPDFPageArea	The mouse is over a page.
kPDFPopupArea	The mouse is over a popup menu.
kPDFTextArea	The mouse is over text.
kPDFTextFieldArea	The mouse is over a text field.
PDFAreaOfInterest	
PDFDisplayMode	
PDFViewAnnotationHitNotification	Posted when the user clicks on an annotation.
PDFViewChangedHistoryNotification	Posted when the page history changes.
PDFViewCopyPermissionNotification	Posted when the user attempts to copy to the pasteboard without the appropriate permissions.
PDFViewDocumentChangedNotification	Posted when a new document is associated with the view.
PDFViewPageChangedNotification	Posted when a new page becomes the current page.
PDFViewPrintPermissionNotification	Posted when the user attempts to print without the appropriate permissions.
PDFViewScaleChangedNotification	Posted when the scale factor changes.

QuartzComposer

QCRenderer.h

Data Types & Constants

All of the new data types and constants in this header file are listed alphabetically, with links to documentation and abstracts, if available.

QCCompositionAttributeCopyrightKey	The key for composition copyright information. The associated value is an NSString object.
QCCompositionAttributeDescriptionKey	The key for the composition description. The associated value is an NSString object.

`QCCompositionAttributeNameKey`	The key for the composition name. The associated value is an NSString object.
`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.
`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.
`QCPortAttributeNameKey`	The key for the port name. The associated value is an NSString object.
`QCPortAttributeTypeKey`	The key for the port type. The associated value can be of any of the following constants: QCPortTypeBoolean, QCPortTypeIndex, QCPortTypeNumber, QCPortTypeString, QCPortTypeColor, QCPortTypeImage, or QCPortTypeStructure.
`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.
`QCPortTypeColor`	The port type for a color value. The associated value must be an NSColor object.
`QCPortTypeImage`	The port type for an image. The associated value can be an NSImage object or a CIImage object.
`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.
`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.
`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.
`QCPortTypeStructure`	The port type for an array, dictionary, or other structure, such as an NSArray or NSDictionary object.
`QCRendererEventKey`	A key for a renderer event. The associated value is an NSEvent object.
`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).

QCView.h

Data Types & Constants

All of the new data types and constants in this header file are listed alphabetically, with links to documentation and abstracts, if available.

`QCViewDidStartRenderingNotification`	Posted when the view starts rendering.
`QCViewDidStopRenderingNotification`	Posted when the view stops rendering.

Document Revision History

This table describes the changes to *Quartz Reference Update*.

Date	Notes
2007-07-18	Updated with the symbols added to the Quartz framework in Mac OS X v10.5.
2005-07-07	Fixed links.
2005-04-29	Minor wording changes.
	New document that summarizes the symbols added to the Quartz framework in Mac OS X v10.4.