QuickTime Kit Framework Reference

QuickTime > Cocoa

2007-10-31

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

Simultaneously published in the United States and Canada.

Contents

Tables

QuickTime Kit Framework Reference

Framework	QTKit.framework
Header file directories	QTKit/QTKit.h
Companion guide	QuickTime Kit Programming Guide
Declared in	QTCaptureAudioPreviewOutput.h QTCaptureConnection.h QTCaptureDecompressedVideoOutput.h QTCaptureDevice.h QTCaptureDeviceInput.h QTCaptureFileOutput.h QTCaptureInput.h QTCaptureLayer.h QTCaptureOutput.h QTCaptureSession.h QTCaptureVideoPreviewOutput.h QTCaptureView.h QTCompressionOptions.h QTDataReference.h QTError.h QTFormatDescription.h QTMedia.h QTMovie.h QTMovieLayer.h QTMovieView.h QTSampleBuffer.h QTTime.h QTTimeRange.h QTTrack.h QTUtilities.h

Introduction

The QuickTime Kit is a Objective-C framework (`QTKit.framework`) with a robust and evolving API for manipulating time-based media. Introduced in Mac OS X v10.4, the QuickTime Kit provides a set of Objective-C classes and methods designed for the basic manipulation of media, including movie playback, editing, import and export to standard media formats, among other capabilities. With the release of Mac OS X v10.5 and the latest iteration of QuickTime 7, the reach and capability of the framework have been extended. The QuickTime Kit framework now includes the addition of 15 new classes, all designed to support professional-level video and audio capture, as well as pro-grade recording of media.

Developers who work with the Cocoa Application Kit classes `NSMovie` and `NSMovieView` should move their applications to the QuickTime Kit framework in order to take advantage of the power and enhanced functionality of this API.

Note: The QuickTime Kit framework supports applications running in Mac OS X v10.3. Applications running in Mac OS X v10.3 require QuickTime 7 or later, however.

Important: The issue of thread-safety has been addressed for developers in the release of the QuickTime Kit framework available in Mac OS X v10.5. Five new methods belonging to the `QTMovie` class have been added. These include the following class and instance methods that deal specifically with handling and managing thread-safety operations of movie objects: `enterQTKitOnThread`, `enterQTKitOnThreadDisablingThreadSafetyProtection`, `exitQTKitOnThread`, `attachToCurrentThread`, and `detachFromCurrentThread`. For more information, refer to the *QTMovie Class Reference*.

The new QTKit capture classes introduced in Mac OS X v10.5 generally have good thread-safety characteristics. In particular, these classes can be used from any thread, except for `QTCaptureView`, which inherits from `NSView`. Note, however, that although capture sessions and their inputs and outputs can be created, run, and monitored from any thread, any method calls that mutate these objects or access mutable information should be serialized, using locks or other synchronization mechanisms.

See Also

The following documents provide additional information about the QuickTime Kit framework:

- *QuickTime 7 Update Guide*
- *QuickTime 7.1 Update Guide*
- *QuickTime 7.1 Update Reference*
- *QuickTime 7.2.1 Update Guide*
- *QuickTime Movie Creation Guide*

Classes

NSCoder_QTKitAdditions Reference

Inherits from	NSObject
Conforms to	NSObject (NSObject)
Framework	/System/Library/Frameworks/QTKit.framework
Declared in	QTKit/QTTime.h QTKit/QTTimeRange.h
Availability	Available in Mac OS X v10.4 and later.

Overview

The QuickTime Kit supports categories on the `NSCoder` class that allow you to encode and decode structures of type `QTTime` and `QTTimeRange`, in addition to structures of type SMPTETime in Mac OS X v10.5.

Tasks

Encoding Time and Time Ranges

- `encodeQTTime:forKey:` (page 16)
  Encodes a QTTime structure.
- `encodeQTTimeRange:forKey:` (page 17)
  Encodes a QTTimeRange structure range.
- `encodeSMPTETime:forKey:` (page 17)
  Encodes an SMPTETime for the given key.

Decoding Time and Time Ranges

- `decodeQTTimeForKey:` (page 16)
  Decodes a QTTime structure.
- `decodeQTTimeRangeForKey:` (page 16)
  Decodes a QTTimeRange structure.
- `decodeSMPTETimeForKey:` (page 16)
  Decodes an SMPTETime structure encoded by the receiver for the given key.

Instance Methods

decodeQTTimeForKey:

Decodes a QTTime structure.

```
- (QTTime)decodeQTTimeForKey:(NSString *)key
```

Discussion
This method matches an encode `QTTime` message used during encoding.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTTime.h`

decodeQTTimeRangeForKey:

Decodes a QTTimeRange structure.

```
- (QTTimeRange)decodeQTTimeRangeForKey:(NSString *)key
```

Discussion
This method matches an encode `QTTimeRange` message used during encoding.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTTimeRange.h`

decodeSMPTETimeForKey:

Decodes an SMPTETime structure encoded by the receiver for the given key.

```
- (SMPTETime)decodeSMPTETimeForKey:(NSString *)key
```

Availability
Mac OS X v10.5 and later.

Declared In
`QTTime.h`

encodeQTTime:forKey:

Encodes a QTTime structure.

```
- (void)encodeQTTime:(QTTime)timeforKey
    :(NSString *)key
```

Discussion
This method must be matched by a decode `QTTime` message.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTTime.h`

encodeQTTimeRange:forKey:

Encodes a QTTimeRange structure range.

```
- (void)encodeQTTimeRange:(QTTimeRange)rangeforKey
    :(NSString *)key
```

Discussion
This method must be matched by a decode `QTTimeRange` message.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTTimeRange.h`

encodeSMPTETime:forKey:

Encodes an SMPTETime for the given key.

```
- (void)encodeSMPTETime:(SMPTETime)time
    forKey:(NSString *)key
```

Availability
Mac OS X v10.5 and later.

Declared In
`QTTime.h`

NSValue_QTKitAdditions Reference

Inherits from	NSObject
Conforms to	NSObject (NSObject)
Framework	/System/Library/Frameworks/QTKit.framework
Declared in	QTKit/QTTime.h QTKit/QTTimeRange.h
Availability	Available in Mac OS X v10.4 and later.

Overview

The QuickTime Kit supports categories in the Foundation framework's `NSValue` class that allow you to get `QTTime` and `QTTimeRange` structures as objects of type `NSValue`. In Mac OS X v10.5, QTKit defines extra operations on the `SMPTETime` type. `SMPTETime` is defined in `CoreAudio/CoreAudioTypes.h`.

Tasks

Wrapping Time and Time Range Structures

`+ valueWithQTTime:` (page 20)
Creates an NSValue object that wraps the specified `QTTime` structure.

`+ valueWithQTTimeRange:` (page 20)
Creates an NSValue object that wraps the specified `QTTimeRange` structure.

`+ valueWithSMPTETime:` (page 20)
Returns a new NSValue object containing an `SMPTETime`.

`- QTTimeValue` (page 21)
Returns a QTTime structure that contains the time in an NSValue object.

`- SMPTETimeValue` (page 21)
Returns a SMPTETime structure contained in an NSValue.

`- QTTimeRangeValue` (page 20)
Returns a QTTimeRange structure that contains the range in an NSValue object.

Class Methods

valueWithQTTime:

Creates an NSValue object that wraps the specified `QTTime` structure.

```
+ (NSValue *)valueWithQTTime:(QTTime)time
```

Availability
Available in Mac OS X v10.3 and later.

Related Sample Code
QTAudioExtractionPanel
QTKitMovieShuffler

Declared In
`QTTime.h`

valueWithQTTimeRange:

Creates an NSValue object that wraps the specified `QTTimeRange` structure.

```
+ (NSValue *)valueWithQTTimeRange:(QTTimeRange)range
```

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTTimeRange.h`

valueWithSMPTETime:

Returns a new NSValue object containing an `SMPTETime`.

```
+ (NSValue *)valueWithSMPTETime:(SMPTETime)time
```

Availability
Mac OS X v10.5 and later.

Declared In
`QTTime.h`

Instance Methods

QTTimeRangeValue

Returns a QTTimeRange structure that contains the range in an NSValue object.

```
- (QTTimeRange)QTTimeRangeValue
```

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTTimeRange.h`

QTTimeValue

Returns a QTTime structure that contains the time in an NSValue object.

```
- (QTTime)QTTimeValue
```

Availability
Available in Mac OS X v10.3 and later.

Related Sample Code
QTKitMovieShuffler

Declared In
`QTTime.h`

SMPTETimeValue

Returns a SMPTETime structure contained in an NSValue.

```
- (SMPTETime)SMPTETimeValue
```

Availability
Mac OS X v10.5 and later.

Declared In
`QTTime.h`

QTCaptureAudioPreviewOutput Class Reference

Inherits from	QTCaptureOutput : NSObject
Conforms to	NSObject (NSObject)
Framework	Library/Frameworks/QTKit.framework
Declared in	QTKit/QTCaptureAudioPreviewOutput.h
Availability	Available in QuickTime 7.2.1 and later.
Related sample code	QTRecorder

Overview

This class represents an output destination for `QTCaptureSession` that can be used to preview the audio being captured. Instances of `QTCaptureAudioPreviewOutput` have an associated Core Audio output device that can be used to play audio being captured by the capture session. Note that the unique ID of a Core Audio device can be obtained from its `kAudioDevicePropertyDeviceUID` property. For more information about Core Audio, refer to the *Apple Core Audio Format Specification 1.0*.

Tasks

Getting and Setting Core Audio Output Devices

- `outputDeviceUniqueID` (page 24)
  Returns the unique ID of the Core Audio output device being used to play preview audio.
- `setOutputDeviceUniqueID:` (page 24)
  Sets the unique ID of the Core Audio output device being used to play preview audio.
- `setVolume:` (page 24)
  Sets the preview volume of the output.
- `volume` (page 25)
  Returns the preview volume of the output.

Instance Methods

outputDeviceUniqueID

Returns the unique ID of the Core Audio output device being used to play preview audio.

```
- (NSString *)deviceUniqueID
```

Return Value
The unique ID of the Core Audio device used for preview, or `NIL` if the default system output device is being used.

Availability
Mac OS X v10.5 and later.

Declared In
`QTCaptureAudioPreviewOutput.h`

setOutputDeviceUniqueID:

Sets the unique ID of the Core Audio output device being used to play preview audio.

```
- (void)setOutputDeviceUniqueID:(NSString *)deviceUniqueID
```

Parameters
deviceUniqueID
The unique ID of the Core Audio device to be used for output, or `NIL` if the default system output should be used.

Availability
Mac OS X v10.5 and later.

Declared In
`QTCaptureAudioPreviewOutput.h`

setVolume:

Sets the preview volume of the output.

```
- (void)setVolume:(float)volume
```

Parameters
volume
The preview volume of the receiver, where 1.0 is the maximum volume and 0.0 is muted.

Availability
Mac OS X v10.5 and later.

Declared In
`QTCaptureAudioPreviewOutput.h`

volume

Returns the preview volume of the output.

```
- (float)volume
```

Return Value
The preview volume of the receiver, where 1.0 is the maximum volume and 0.0 is muted.

Availability
Mac OS X v10.5 and later.

Declared In
`QTCaptureAudioPreviewOutput.h`

QTCaptureConnection Reference

Inherits from	NSObject
Conforms to	NSObject (NSObject)
Framework	Library/Frameworks/QTKit.framework
Declared in	QTKit/QTCaptureConnection.h
Availability	Available in QuickTime 7.2.1 and later.
Related sample code	QTRecorder

Overview

This class represents a connection over which a single stream of media data is sent from a `QTCaptureInput` to a `QTCaptureSession` and from a `QTCaptureSession` to a `QTCaptureOutput`.

Instances of `QTCaptureConnection` wrap individual media streams that can be provided by `QTCaptureInput` objects and received by `QTCaptureOutput` objects. Connections can have a QuickTime media type, such as `QTMediaTypeVideo` and `QTMediaTypeSound`, and a format description that describes the media sent or received across the connection. Individual connections belonging to an input can be enabled or disabled to restrict what media enters a capture session, while connections belonging to an output can be enabled or disabled to restrict what media enters the output from the capture session. In addition, if a `QTCaptureConnection` wraps a stream of audio media, it provides a number of attributes to control the volume, mix, and enabled channels of the audio passing through it.

`QTCaptureConnection` objects can have extended attributes that applications can read using the `attributeForKey:` and `connectionAttributes` methods. Some attributes, for which the `attributeIsReadOnly:` method returns `NO`, can be edited using the `setAttribute:forKey:` and `setConnectionAttributes:` methods. In addition to these explicit methods, applications can use key-value coding to get and set extended attributes. For an object that supports a given attribute, `valueForKey:` will be functionally identical to `attributeForKey:`, and `setValue:forKey:` will be identical to `setAttribute:forKey:`. Applications wishing to observe changes for a given attribute can add a key-value observer where the key path is the attribute key.

Tasks

Getting and Setting Connection Attributes

- `attributeForKey:` (page 28)
  Returns the current value of the connection attribute for key.
- `attributeIsReadOnly:` (page 29)
  Returns a Boolean value indicating whether the given attribute for the connection cannot be modified.
- `connectionAttributes` (page 29)
  Returns a dictionary of all attributes set for the receiver.
- `formatDescription` (page 29)
  Returns the format description of the receiver.
- `isEnabled` (page 30)
  Returns a Boolean value indicating whether the receiver is enabled.
- `mediaType` (page 30)
  Returns the QuickTime media type of the receiver.
- `owner` (page 30)
  Returns the `QTCaptureInput` or `QTCaptureOutput` object that owns the receiver.
- `setAttribute:forKey:` (page 31)
  Sets a connection attribute for the given key.
- `setConnectionAttributes:` (page 31)
  Sets the connection's attributes from the key-value pairs specified in the given dictionary.
- `setEnabled:` (page 31)
  Sets whether the receiver is enabled.

Instance Methods

attributeForKey:

Returns the current value of the connection attribute for key.

```
- (id)attributeForKey:(NSString *)key
```

Discussion
Use this method to get attributes of a connection. The keys that can be used with this method are described in the Constants section. Applications using key-value coding can also get an attribute for a given key by passing that key to the `NSObject valueForKey:` method.

Availability
Mac OS X v10.5 and later.

Related Sample Code
QTRecorder

Declared In
QTCaptureConnection.h

attributeIsReadOnly:

Returns a Boolean value indicating whether the given attribute for the connection cannot be modified.

```
- (BOOL)attributeIsReadOnly:(NSString *)attributeKey
```

Return Value
Returns YES if the attribute cannot be modified; otherwise, NO.

Availability
Mac OS X v10.5 and later.

Declared In
QTCaptureConnection.h

connectionAttributes

Returns a dictionary of all attributes set for the receiver.

```
- (NSDictionary *)connectionAttributes
```

Discussion
Applications can use this method to determine what attributes a specific connection supports.

Availability
Mac OS X v10.5 and later.

Declared In
QTCaptureConnection.h

formatDescription

Returns the format description of the receiver.

```
- (QTFormatDescription *)formatDescription
```

Discussion
This method returns the format description of the connection, allowing applications to monitor various attributes of the media being sent or received by the connection (the display size of video media, for example). Applications can be notified of changes to the connection's format by registering to receive QTCaptureConnectionFormatDescriptionWillChangeNotification and QTCaptureConnectionFormatDescriptionDidChangeNotification notifications or by adding a key-value observer to the connection for the key @"formatDescription".

Availability
Mac OS X v10.5 and later.

Related Sample Code
QTRecorder

Declared In
QTCaptureConnection.h

isEnabled

Returns a Boolean value indicating whether the receiver is enabled.

```
- (BOOL)isEnabled
```

Discussion
This method returns a Boolean indicating whether the receiver is enabled to send or receive media data. Individual connections can be enabled or disabled using the `setEnabled:` method.

Availability
Mac OS X v10.5 and later.

Declared In
QTCaptureConnection.h

mediaType

Returns the QuickTime media type of the receiver.

```
- (NSString *)mediaType
```

Return Value
A QuickTime media type, as defined in `QTMedia.h`.

Discussion
This method returns the QuickTime media type, such as `QTMediaTypeVideo` and `QTMediaTypeSound`, of the receiver.

Availability
Mac OS X v10.5 and later.

Declared In
QTCaptureConnection.h

owner

Returns the `QTCaptureInput` or `QTCaptureOutput` object that owns the receiver.

```
- (id)owner
```

Return Value
A `QTCaptureInput` or `QTCaptureOutput` object that uses the receiver as a media connection.

Discussion
This method returns the input or output to which the receiver belongs. The returned input or output uses the receiver as a connection for sending or receiving a media stream.

Availability
Mac OS X v10.5 and later.

Declared In
QTCaptureConnection.h

setAttribute:forKey:

Sets a connection attribute for the given key.

```
- (void)setAttribute:(id)property
    forKey:(NSString *)attributeKey
```

Discussion
Use this method to set attributes of a capture connection. The keys that can be used with this method are described in the Constants section. This method raises an NSInvalidArgumentException if the attribute is read-only or not supported by the receiver. Applications using key-value coding can also set an attribute for a given key by passing that key to the `NSObject setValue:forKey:` method.

Availability
Mac OS X v10.5 and later.

Declared In
QTCaptureConnection.h

setConnectionAttributes:

Sets the connection's attributes from the key-value pairs specified in the given dictionary.

```
- (void)setConnectionAttributes:(NSDictionary *)connectionAttributes
```

Discussion
This method allows application to set multiple attributes on a connection at once. This method raises an NSInvalidArgumentException if any of the attributes in the dictionary are read-only or not supported by the receiver. Applications using key-value coding can also set multiple attributes using the `NSObject setValuesForKeysWithDictionary:` method using attribute keys as keys in the dictionary.

Availability
Mac OS X v10.5 and later.

Declared In
QTCaptureConnection.h

setEnabled:

Sets whether the receiver is enabled.

```
- (void)setEnabled:(BOOL)enabled
```

Discussion
This method sets whether the receiver is enabled to send or receive media data.

Availability
Mac OS X v10.5 and later.

Declared In
`QTCaptureConnection.h`

Constants

QTCaptureConnectionAudioAveragePowerLevelsAttribute

An NSArray of NSNumbers that correspond to the average power, in decibels, of each audio stream sent through the connection.

```
QTCaptureConnectionAudioAveragePowerLevelsAttribute
@"audioAveragePowerLevels"
```

Constants

`QTCaptureConnectionAudioAveragePowerLevelsAttribute`

Applications that wish to display audio level meters for a specific connection can periodically check the value of this attribute. Average power levels change quickly and aqppear jumpy on a level meter.

Discussion
This string value can be used in key paths for key-value coding, key-value observing, and bindings.

Declared In
`QTCaptureConnection.h`

QTCaptureConnectionAudioMasterVolumeAttribute

An NSNumber that specifies the master volume of all audio channels sent through the connection.

```
QTCaptureConnectionAudioMasterVolumeAttribute
@"audioMasterVolume"
```

Constants

`QTCaptureConnectionAudioMasterVolumeAttribute`

The values are between 0.0 and 1.0 for normal volume, or greater than 1.0 for boosting the audio gain.This attribute determines the master volumes of all audio channels sent through the connection. Applications that need to set the volumes of individual channels can set the `QTCaptureConnectionAudioVolumesAttribute` attribute.

Discussion
This string value can be used in key paths for key-value coding, key-value observing, and bindings.

Declared In
`QTCaptureConnection.h`

QTCaptureConnectionAudioPeakHoldLevelsAttribute

An NSArray of NSNumbers that correspond to the peak hold level, in decibels, of each audio channel sent through the connection.

```
QTCaptureConnectionAudioPeakHoldLevelsAttribute
@"audioPeakHoldLevels"
```

Constants

`QTCaptureConnectionAudioPeakHoldLevelsAttribute`

Applications that wish to display audio level meters for a specific connection can periodically check the value of this attribute. Peak hold levels remain at the maximum volume for about a second, and are often useful for displaying audio clipping.

Discussion

This string value can be used in key paths for key-value coding, key-value observing, and bindings.

Declared In

`QTCaptureConnection.h`

QTCaptureConnectionAudioVolumesAttribute

An NSArray of NSNumbers that specify the volumes of audio channels sent through the connection.

```
QTCaptureConnectionAudioVolumesAttribute
@"audioVolumes"
```

Constants

`QTCaptureConnectionAudioVolumesAttribute`

The values are between 0.0 and 1.0 for normal volume, or greater than 1.0 for boosting the audio gain. This attribute determines the individual volumes of audio channels sent through the connection. Applications that need to set the master volume of all channels can set the `QTCaptureConnectionAudioMasterVolumeAttribute` attribute.

Discussion

This string value can be used in key paths for key-value coding, key-value observing, and bindings.

Declared In

`HeaderFile`

QTCaptureConnectionEnabledAudioChannelsAttribute

An NSIndexSet that specifies which audio channels should be sent through the connection.

```
QTCaptureConnectionEnabledAudioChannelsAttribute
@"enabledAudioChannels"
```

Constants

`QTCaptureConnectionEnabledAudioChannelsAttribute`

The indices in the set should be between 0 and the number of volumes in `QTCaptureConnectionAudioVolumesAttribute`. This attribute allows applications to selectively disable certain audio channels from being sent rhough the connection. The value of this attribute should be an NSIndexSet that contains only the channels that should be used. By default, all audio channels are sent though a connection.

Discussion

This string value can be used in key paths for key-value coding, key-value observing, and bindings.

Declared In

`QTCaptureConnection.h`

Notifications

The following are notifications enabling you to change attributes, keys, and format descriptions.

QTCaptureConnectionAttributeDidChangeNotification

Posted when one of the connection's attributes has changed.

The notification's user info dictionary will contain the attribute key of the changed attribute for the key `QTCaptureConnectionChangedAttributeKey`.

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

Declared In
`QTCaptureConnection.h`

QTCaptureConnectionAttributeWillChangeNotification

Posted when one of the connection's attributes is about to change.

The notification's user info dictionary will contain the attribute key of the changed attribute for the key `QTCaptureConnectionChangedAttributeKey`.

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

Declared In
`QTCaptureConnection.h`

QTCaptureConnectionChangedAttributeKey

Used as a key in the user info dictionary passed to `QTCaptureConnectionAttributeWillChangeNotification`, and `QTCaptureConnectionAttributeDidChangeNotification` to indicate the key of that attribute that changed.

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

Declared In
`QTCaptureConnection.h`

QTCaptureConnectionFormatDescriptionDidChangeNotification

Posted when the format description of a connection has changed.

Applications can be notified of changes to a connection's format by registering to receive this notification.

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

Declared In
`QTCaptureConnection.h`

QTCaptureConnectionFormatDescriptionWillChangeNotification

Posted when the format description of a connection is about to change.

Applications can be notified of changes to a connection's format by registering to receive this notification.

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

Declared In
`QTCaptureConnection.h`

QTCaptureDecompressedVideoOutput Reference

Inherits from	QTCaptureOutput : NSObject
Conforms to	NSObject (NSObject)
Framework	Library/Frameworks/QTKit.framework
Declared in	QTKit/QTCaptureDecompressedVideoOutput.h
Availability	Available in QuickTime 7.2.1 and later.

Overview

This class represents an output destination for a `QTCaptureSession` object that can be used to process decompressed frames from the video being captured. Instances of `QTCaptureDecompressedVideoOutput` produce decompressed video frames suitable for high-quality processing. Because instances maintain maximum frame quality and avoid dropping frames, using this output may result in reduced performance while capturing. Applications that need to process decompressed frames but can tolerate dropped frames or drops in decompression quality should use `QTCaptureVideoPreviewOutput` instead. Applications can access the decompressed frames via the `captureOutput:didOutputVideoFrame:withSampleBuffer:fromConnection:` delegate method. Clients can also create subclasses of `QTCaptureDecompressedVideoOutput` to add custom capturing behavior.

Tasks

Decompressing Video Output

- `delegate` (page 38)
  Returns the receiver's delegate.
- `setDelegate:` (page 40)
  Sets the receiver's delegate.
- `outputVideoFrame:withSampleBuffer:fromConnection:` (page 38)
  Called whenever the receiver outputs a new video frame.
- `pixelBufferAttributes` (page 39)
  Returns the Core Video pixel buffer attributes previously set by `setPixelBufferAttributes:` that determine what kind of pixel buffers are output by the receiver.

`- setPixelBufferAttributes:` (page 40)
Sets the CoreVideo pixel buffer attributes that determine what kind of pixel buffers are output by the receiver.

`- captureOutput:didOutputVideoFrame:withSampleBuffer:fromConnection:` (page 41) *delegate method*
Called whenever the video preview output outputs a new video frame.

Instance Methods

delegate

Returns the receiver's delegate.

```
- (id)delegate
```

Availability
Mac OS X v10.5 and later.

Declared In
`QTCaptureDecompressedVideoOutput.h`

outputVideoFrame:withSampleBuffer:fromConnection:

Called whenever the receiver outputs a new video frame.

```
- (void)outputVideoFrame:(CVImageBufferRef)videoFrame
    withSampleBuffer:(QTSampleBuffer *)sampleBuffer
    fromConnection:(QTCaptureConnection *)connection
```

Parameters

videoFrame
A Core Video buffer containing the decompressed frame.

sampleBuffer
A sample buffer containing additional information about the frame, such as its presentation time.

connection
The connection from which the video was received.

Discussion
This method should not be invoked directly. Subclasses can override this method to provide custom processing behavior for each frame. The default implementation calls the delegate's `captureOutput:didOutputVideoFrame:withSampleBuffer:fromConnection:` method.

Warning: Subclasses should not assume that this method will be called on the main thread. In addition, this method is called periodically, so it must be efficient to prevent capture performance problems.

Special Considerations

In order to promptly reclaim memory resources, after this method returns, the sample data contained within the `QTSampleBuffer` object will be released using its `decrementSampleUseCount` method. Clients that reference the sample buffer and are interested in the sample data that it contains after this method returns should call `incrementSampleUseCount` on the sample buffer within this method to ensure that the data remains valid until they no longer need it (at which time they should call `decrementSampleUseCount`). Clients that reference the sample buffer after this method returns, but only need access to its metadata, such as duration, presentation time, and other attributes, need not call `incrementSampleUseCount`. Note that to maintain optimal performance, some sample buffers directly reference pools of memory that may need to be reused by the device system and other capture inputs. This is frequently the case for uncompressed device native capture where memory blocks are copied as little as possible. If multiple sample buffers reference such pools of memory for too long, inputs will no longer be able to copy new samples into memory and those samples will be dropped. If your application is causing samples to be dropped by holding on to sample data for too long using `incrementSampleUseCount`, but it needs access to the sample data for a long period of time, consider copying the data into a new buffer and then calling `decrementSampleUseCount` on the sample buffer so that the memory it references can be reused.

Availability

Mac OS X v10.5 and later.

Not available to 64-bit applications.

Declared In

`QTCaptureDecompressedVideoOutput.h`

pixelBufferAttributes

Returns the Core Video pixel buffer attributes previously set by `setPixelBufferAttributes:` that determine what kind of pixel buffers are output by the receiver.

```
- (NSDictionary *)pixelBufferAttributes
```

Return Value

A dictionary containing pixel buffer attributes for buffers output by the reciever. The keys in the dictionary are described in `CoreVideo/CVPixelBuffer.h`. If the return value is `NIL`, then the receiver outputs buffers using the fastest possible pixel buffer attributes.

Discussion

This method returns the pixel buffer attributes set by `setPixelBufferAttributes:` that clients can use to customize the size and pixel format of the video frames output by the receiver. When the dictionary is non-nil, the receiver will attempt to output pixel buffers using the attributes specified in the dictionary. A non-nil dictionary also guarantees that the output `CVImageBuffer` is a `CVPixelBuffer`. When the value for `kCVPixelBufferPixelFormatTypeKey` is set to an NSNumber, all image buffers output by the receiver will be in that format. When the value is an NSArray, image buffers output by the receiver will be in the most optimal format specified in that array. If the captured images are not in the one of the specified pixel formats, then a format conversion will be performed. If the dictionary is `NIL` or there is no value for the `kCVPixelBufferPixelFormatTypeKey`, then the receiver will output images in the most efficient possible format given the input. For example, if the source is an iSight producing component Y'CbCr 8-bit 4:2:2 video then Y'CbCr 8-bit 4:2:2 will be used as the output format in order to avoid any conversions. The default value for the returned dictionary is `NIL`.

Availability

Available in Mac OS X v10.5 and later.

Declared In
`QTCaptureDecompressedVideoOutput.h`

setDelegate:

Sets the receiver's delegate.

```
- (void)setDelegate:(id)delegate
```

Availability
Mac OS X v10.5 and later.

Declared In
`QTCaptureDecompressedVideoOutput.h`

setPixelBufferAttributes:

Sets the CoreVideo pixel buffer attributes that determine what kind of pixel buffers are output by the receiver.

```
- (void)setPixelBufferAttributes:(NSDictionary *)pixelBufferAttributes
```

Parameters

`pixelBufferAttributes`

A dictionary containing pixel buffer attributes for buffers that will be output by the reciever. The keys in the dictionary are described in `CoreVideo/CVPixelBuffer.h`. If the dictionary is `NIL`, then the receiver outputs buffers using the fastest possible pixel buffer attributes.

Discussion
This method sets the pixel buffer attributes that clients can use to customize the size and pixel format of the video frames output by the receiver. When the dictionary is non-nil, the receiver will attempt to output pixel buffers using the attributes specified in the dictionary. A non-nil dictionary also guarantees that the output `CVImageBuffer` is a `CVPixelBuffer`. When the value for `kCVPixelBufferPixelFormatTypeKey` is set to an NSNumber, all image buffers output by the receiver will be in that format. When the value is an NSArray, image buffers output by the receiver will be in the most optimal format specified in that array. If the captured images are not in the one of the specified pixel formats, then a format conversion will be performed. If the dictionary is `NIL` or there is no value for the `kCVPixelBufferPixelFormatTypeKey`, then the receiver will output images in the most efficient possible format given the input. For example, if the source is an iSight producing component Y'CbCr 8-bit 4:2:2 video then Y'CbCr 8-bit 4:2:2 will be used as the output format in order to avoid any conversions.

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

Declared In
`QTCaptureDecompressedVideoOutput.h`

Delegate Methods

captureOutput:didOutputVideoFrame:withSampleBuffer:fromConnection:

Called whenever the video preview output outputs a new video frame.

```
- (void)captureOutput:(QTCaptureOutput *)captureOutput
    didOutputVideoFrame:(CVImageBufferRef)videoFrame
    withSampleBuffer:(QTSampleBuffer *)sampleBuffer
    fromConnection:(QTCaptureConnection *)connection
```

Parameters

captureOutput
The `QTCaptureDecompressedVideoOutput` instance that output the frame.

videoFrame
A Core Video image buffer containing the decompressed frame.

sampleBuffer
A sample buffer containing additional information about the frame, such as its presentation time.

connection
The connection from which the video was received.

Discussion

Delegates receive this message whenever the output decompresses and outputs a new video frame. Delegates can use the provided video frame for a custom preview or for further image processing.

Warning: Delegates should not assume that this method will be called on the main thread. In addition, this method is called periodically, so it must be efficient to prevent capture performance problems.

Special Considerations

In order to promptly reclaim memory resources, after this method returns, the sample data contained within the `QTSampleBuffer` object will be released using its `decrementSampleUseCount` method. Clients that reference the sample buffer and are interested in the sample data that it contains after this method returns should call `incrementSampleUseCount` on the sample buffer within this method to ensure that the data remains valid until they no longer need it (at which time they should call `decrementSampleUseCount`). Clients that reference the sample buffer after this method returns, but only need access to its metadata, such as duration, presentation time, and other attributes, need not call `incrementSampleUseCount`. Note that to maintain optimal performance, some sample buffers directly reference pools of memory that may need to be reused by the device system and other capture inputs. This is frequently the case for uncompressed device native capture where memory blocks are copied as little as possible. If multiple sample buffers reference such pools of memory for too long, inputs will no longer be able to copy new samples into memory and those samples will be dropped. If your application is causing samples to be dropped by holding on to sample data for too long using `incrementSampleUseCount`, but it needs access to the sample data for a long period of time, consider copying the data into a new buffer and then calling `decrementSampleUseCount` on the sample buffer so that the memory it references can be reused.

Availability

Mac OS X v10.5 and later.

Declared In

`QTCaptureDecompressedVideoOutput.h`

QTCaptureDevice Class Reference

Inherits from	NSObject
Conforms to	NSCoding NSObject (NSObject)
Framework	Library/Frameworks/QTKit.framework
Declared in	QTKit/QTCaptureDevice.h
Availability	Available in QuickTime 7.2.1 and later.
Related sample code	LiveVideoMixer3 QT Capture Widget QTRecorder

Overview

This class represents an available capture device. Each instance of `QTCaptureDevice` corresponds to a capture device that is connected or has been previously connected to the user's computer during the lifetime of the application. Instances of `QTCaptureDevice` cannot be created directly. A single unique instance is created automatically whenever a device is connected to the computer and can be accessed using the `deviceWithUniqueID:` class method. An array of all currently connected devices can also be obtained using the `inputDevices:` class method.

Devices can provide one or more stream of a given media type. Applications can search for devices that provide media of a specific type using the `inputDevicesWithMediaType:` and `defaultInputDeviceWithMediaType` class methods. Table 6-1 details the media types supported by QTCaptureDevice and examples of devices that support them:

Table 6-1 Media types supported by `QTCaptureDevice`

Media Type	Description	Example Devices
`QTMediaTypeVideo`	Media that only contains video frames.	iSight cameras (external and built-in); USB and FireWire webcams
`QTMediaTypeMuxed`	Multiplexed media that may contain audio, video, and other data in a single stream.	DV cameras

Media Type	Description	Example Devices
`QTMediaTypeSound`	Media that only contains audio samples.	Built-in microphones and line-in jacks; the microphone built-in to the external iSight; USB microphones and headsets; any other device supported by Core Audio.

`QTCaptureDevice` objects can have extended attributes that applications can read using the `attributeForKey:` and `deviceAttributes` methods. Some attributes, for which the `attributeIsReadOnly:` method returns NO, can be edited using the `setAttribute:forKey:` and `setDeviceAttributes:` methods. In addition to these explicit methods, applications can use key-value coding to get and set extended attributes. For an object that supports a given attribute, `valueForKey:` will be functionally identical to `attributeForKey:`, and `setValue:forKey:` will be identical to `setAttribute:forKey:`. Applications wishing to observe changes for a given attribute can add a key-value observer where the key path is the attribute key.

Tasks

Finding Devices

`+ defaultInputDeviceWithMediaType:` (page 45)
Returns a `QTCaptureDevice` instance for the default device connected to the user's system of the given media type.

`+ deviceWithUniqueID:` (page 46)
Returns a `QTCaptureDevice` instance with the identifier device UID.

`+ inputDevices` (page 46)
Returns an array of devices currently connected to the computer that can be used as input sources.

`+ inputDevicesWithMediaType:` (page 47)
Returns an array of input devices currently connected to the computer that send a stream with the given media type.

Using a Device

`- close` (page 48)
Releases application control over the device acquired in the `open:` method.

`- isConnected` (page 50)
Returns YES if the device is connected to the computer.

`- isInUseByAnotherApplication` (page 50)
Returns YES is the device is connected, but being exclusively used by another application.

`- open:` (page 51)
Attempts to give the application control over the device so that it can be used for capture.

`- isOpen` (page 50)
Returns `YES` if the device is open in the current application.

Getting Information About a Device

- `attributeForKey:` (page 47)
  Returns a device attribute for the given key.
- `attributeIsReadOnly:` (page 48)
  Returns whether the given attribute for the device cannot be modified.
- `deviceAttributes` (page 48)
  Returns a dictionary of the device's current attirbutes.
- `formatDescriptions` (page 49)
  Returns an array of stream formats currently in use by the device.
- `hasMediaType:` (page 49)
  Returns whether the receiver sends a stream with the given media type.
- `setAttribute:forKey:` (page 52)
  Sets a device attribute for the given key.
- `setDeviceAttributes:` (page 52)
  Sets attributes on the device from the key-value pairs in the given dictionary.
- `localizedDisplayName` (page 51)
  Returns a localized human-readable name for the receiver's device.
- `modelUniqueID` (page 51)
  Returns the unique ID of the model of the receiver's device.
- `uniqueID` (page 53)
  Returns the unique ID of the receiver's device.

Class Methods

defaultInputDeviceWithMediaType:

Returns a `QTCaptureDevice` instance for the default device connected to the user's system of the given media type.

```
+ (QTCaptureDevice *)defaultInputDeviceWithMediaType:(NSString *)mediaType
```

Parameters

mediaType
  The media type, such as `QTMediaTypeVideo`, `QTMediaTypeSound`, or `QTMediaTypeMuxed`, supported by the returned device.

Return Value
The default device with the given media type on the user's system, or `NIL` if no device with that media type exists.

Discussion
This method returns the default device of the given media type connected to the user's system. For example, for `QTMediaTypeSound`, this method will return the default sound input device selected in the Sound Preference Pane. If there is no device for the given media type, this method will return nil.

Availability
Mac OS X v10.5 and later.

Related Sample Code
QT Capture Widget

Declared In
`QTCaptureDevice.h`

deviceWithUniqueID:

Returns a `QTCaptureDevice` instance with the identifier device UID.

```
+ (QTCaptureDevice *)deviceWithUniqueID:(NSString *)deviceUID
```

Parameters

`deviceUID`
The unique identifier of the device instance to be returned.

Return Value
If a device with unique identifier `deviceUID` was connected to the computer at some point during the lifetime of the application, this method returns a `QTCaptureDevice` instance for that identifier. Otherwise, this method returns `NIL`.

Discussion
Every capture device available to the computer is assigned a unique identifier that persists on one computer across device connections and disconnections, as well as across reboots of the computer. This method can be used to recall or track the status of a specific device, even if it has been disconnected.

Availability
Mac OS X v10.5 and later.

Declared In
`QTCaptureDevice.h`

inputDevices

Returns an array of devices currently connected to the computer that can be used as input sources.

```
+ (NSArray *)inputDevices
```

Return Value
An NSArray of `QTCaptureDevice` instances for each connected device. If there are no available devices, the returned array will be empty.

Discussion
This method queries the device system and builds an array of `QTCaptureDevice` instances for input devices currently connected and available for capture. The returned array contains all devices that are available when the method is called. Applications should observe `QTCaptureDeviceWasConnectedNotification` and `QTCaptureDeviceWasDisconnectedNotification` to be notified when the list of available devices has changed.

Availability
Mac OS X v10.5 and later.

Related Sample Code
LiveVideoMixer3

Declared In
`QTCaptureDevice.h`

inputDevicesWithMediaType:

Returns an array of input devices currently connected to the computer that send a stream with the given media type.

```
+ (NSArray *)inputDevicesWithMediaType:(NSString *)mediaType
```

Parameters

mediaType

The media type, such as `QTMediaTypeVideo`, `QTMediaTypeSound`, or `QTMediaTypeMuxed`, supported by each returned device.

Return Value
An array of `QTCaptureDevice` instances for each connected device with the given media type. If there are no available devices, the returned array will be empty.

Discussion
This method queries the device system and builds an array of `QTCaptureDevice` instances for input devices that are currently connected and output streams of the given media type.

Availability
Mac OS X v10.5 and later.

Related Sample Code
QTRecorder

Declared In
`QTCaptureDevice.h`

Instance Methods

attributeForKey:

Returns a device attribute for the given key.

```
- (id)attributeForKey:(NSString *)attributeKey
```

Discussion
Use this method to get attributes of a device. The keys that can be used with this method are described in the Constants section. Applications using key-value coding can also get an attribute for a given key by passing that key to the NSObject `valueForKey:` method.

Availability
Mac OS X v10.5 and later.

Related Sample Code
LiveVideoMixer3
QTRecorder

Declared In
`QTCaptureDevice.h`

attributeIsReadOnly:

Returns whether the given attribute for the device cannot be modified.

```
- (BOOL)attributeIsReadOnly:(NSString *)attributeKey
```

Return Value
Returns YES if the attribute cannot be modified; otherwise, NO.

Availability
Mac OS X v10.5 and later.

Declared In
`QTCaptureDevice.h`

close

Releases application control over the device acquired in the `open:` method.

```
- (void)close
```

Discussion
This method should be called to match each invocation of `open:` when an application no longer needs to use a device for capture. If a device is disconnected or turned off while it is open it will be closed automatically. Applications should check if a device has not been closed automatically by registering to receive `QTCaptureDeviceWasDisconnectedNotification` or by checking `isOpen` before manually closing the device using this method.

Availability
Mac OS X v10.5 and later.

Related Sample Code
QTRecorder

Declared In
`QTCaptureDevice.h`

deviceAttributes

Returns a dictionary of the device's current attirbutes.

```
- (NSDictionary *)deviceAttributes
```

Return Value
An dictionary of attributes supported by the device.

Discussion
Applications can use this method to determine what attributes a specific device supports.

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

Declared In
`QTCaptureDevice.h`

formatDescriptions

Returns an array of stream formats currently in use by the device.

```
- (NSArray *)formatDescriptions
```

Return Value
An array of `QTFormatDescription` objects describing the current stream formats of the device.

Discussion
Applications can use this method to determine what kind of media the receiver outputs. Applications can be notified of format changes by registering to receive `QTCaptureDeviceFormatDescriptionsWillChangeNotification` and `QTCaptureDeviceFormatDescriptionsDidChangeNotification` notifications or by adding a key-value observer for the key `@"formatDescriptions"`.

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

Declared In
`QTCaptureDevice.h`

hasMediaType:

Returns whether the receiver sends a stream with the given media type.

```
- (BOOL)hasMediaType:(NSString *)mediaType
```

Parameters
mediaType
A media type, such as QTMediaTypeVideo, QTMediaTypeSound, or QTMediaTypeMuxed.

Return Value
Returns YES if the device outputs the given media type, NO otherwise.

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

Declared In
`QTCaptureDevice.h`

isConnected

Returns YES if the device is connected to the computer.

```
- (BOOL)isConnected
```

Return Value
Returns `YES` if the device is connected and available to applications; otherwise, `NO`.

Discussion
This method checks whether the receiver's device is currently connected to the computer and available for use by applications.

Availability
Mac OS X v10.5 and later.

Declared In
`QTCaptureDevice.h`

isInUseByAnotherApplication

Returns YES is the device is connected, but being exclusively used by another application.

```
- (BOOL)isInUseByAnotherApplication
```

Return Value
Returns YES if another process has exclusive control over a connected device; otherwise, NO.

Discussion
If the device can only be accessed by one process at a time, this method checks if the process that has exclusive control over the current process.

Availability
Mac OS X v10.5 and later.

Declared In
`QTCaptureDevice.h`

isOpen

Returns `YES` if the device is open in the current application.

```
- (BOOL)isOpen
```

Return Value
Returns `YES` if the device was previously opened by the receiver's `open:` method. Returns `NO` otherwise.

Discussion
The method checks if the device was previously succcessfully opened with the receiver's `open:` method. If this method returns `YES`, the device can be used immediately for capture.

Availability
Mac OS X v10.5 and later.

Declared In
QTCaptureDevice.h

localizedDisplayName

Returns a localized human-readable name for the receiver's device.

```
- (NSString *)localizedDisplayName
```

Return Value
The localized name of the receiver's device.

Discussion
This method can be used when displaying the name of a capture device in the user interface.

Availability
Mac OS X v10.5 and later.

Declared In
QTCaptureDevice.h

modelUniqueID

Returns the unique ID of the model of the receiver's device.

```
- (NSString *)modelUniqueID
```

Return Value
The unique identifier of the model of device corresponding to the recevier.

Discussion
The unique identifier returned by this method is unique to all devices of the same model. The value is persistent across device connections and disconnections, and across different computers.

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

Declared In
QTCaptureDevice.h

open:

Attempts to give the application control over the device so that it can be used for capture.

```
- (BOOL)open:(NSError **)errorPtr
```

Parameters

errorPtr

If not equal to nil, points to an NSError describing why the device could not be opened, or points to nil if the device was opened successfully.

Return Value
Returns YES if the device was opened successfully; otherwise, NO.

Discussion
This method attempts to open the device for control by the current application. If the device is connected and no other processes have exclusive control over it, then the application starts using the device immediately, taking exclusive control of it if necessary. Otherwise, this method returns `NO` and sets errorPtr to point to an error describing why the device could not be opened. Applications that call `open:` should also call the `close` method to relinquish access to the device when it is no longer needed. Multiple calls to this method can be nested. Each call to this method must be matched by a call to `close`. Applications that capture from a device using `QTCaptureDeviceInput` must call this method before creating the `QTCaptureDeviceInput` to be used with the device. If a device is disconnected or turned off while it is open it will be closed automatically.

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

Related Sample Code
QT Capture Widget
QTRecorder

Declared In
`QTCaptureDevice.h`

setAttribute:forKey:

Sets a device attribute for the given key.

```
- (void)setAttribute:(id)property forKey:(NSString *)attributeKey
```

Discussion
Use this method to set attributes of a device. The keys that can be used with this method are described in the Constants section. This method raises an NSInvalidArgumentException if the attribute is read-only or not supported by the receiver. Applications using key-value coding can also set an attribute for a given key by passing that key to the NSObject `setValue:forKey:` method.

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

Related Sample Code
QTRecorder

Declared In
`QTCaptureDevice.h`

setDeviceAttributes:

Sets attributes on the device from the key-value pairs in the given dictionary.

```
- (void)setDeviceAttributes:(NSDictionary *)deviceAttributes
```

Discussion
This method allows application to set multiple attributes on a device at once. This method raises an NSInvalidArgumentException if any of the attributes in the dictionary are read-only or not supported by the receiver. Applications using key-value coding can also set multiple attributes using the NSObject `setValuesForKeysWithDictionary:` method using attribute keys as keys in the dictionary.

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

Declared In
`QTCaptureDevice.h`

uniqueID

Returns the unique ID of the receiver's device.

```
- (NSString *)uniqueID
```

Return Value
The unique identifier of the device corresponding to the receiver.

Discussion
The unique identifier returned by this method is persistent on one computer across device connections and disconnections, as well as across reboots of the computer. It can be passed to the `deviceWithUniqueID:` class method to get the `QTCaptureDevice` instance for the device with that unique identifier.

Availability
Mac OS X v10.5 and later.

Declared In
`QTCaptureDevice.h`

Constants

QTCaptureDeviceChangedAttributeKey

Used as a key in the userInfo dictionary passed to `QTCaptureDeviceAttributeWillChangeNotification`, and `QTCaptureDeviceAttributeDidChangeNotification` to indicate the key of the attribute that changed.

```
QTCaptureDeviceChangedAttributeKey
```

Constants
`QTCaptureDeviceChangedAttributeKey`
Indicates the key of the attribute that changed.

Declared In
`QTCaptureDevice.h`

QTCaptureDeviceAvailableInputSourcesAttribute

For devices with multiple possible input sources, this attribute returns an array of dictionaries describing each available input source.

```
QTCaptureDeviceAvailableInputSourcesAttribute
@"availableInputSources"
```

Constants

`QTCaptureDeviceAvailableInputSourcesAttribute`

Some devices can capture data from one of multiple input sources (different input jacks on the same audio device, for example). The value is an `NSArray` of `NSDictionary` objects. The keys in each dictionary are described in Input Source Dictionary Keys. This string value can be used in key paths for key value coding, key value observing, and bindings.

Declared In

`QTCaptureDevice.h`

QTCaptureDeviceInputSourceIdentifierAttribute

Used to get and set the currently used input source for the device.

```
QTCaptureDeviceInputSourceIdentifierAttribute
@"inputSourceIdentifier"
```

Constants

`QTCaptureDeviceInputSourceIdentifierAttribute`

Some devices can capture data from one of multiple input sources (different input jacks on the same audio device, for example). The value is an object returned by the `QTCaptureDeviceInputSourceIdentifierKey` key in one of the dictionaries returned by `QTCaptureDeviceAvailableInputSourcesAttribute`. This string value can be used in key paths for key value coding, key value observing, and bindings.

Declared In

`QTCaptureDevice.h`

QTCaptureDeviceInputSourceIdentifierKey

An object representing a unique ID for the input source.

```
QTCaptureDeviceInputSourceIdentifierKey
@"identifier"
```

Constants

`QTCaptureDeviceInputSourceIdentifierKey`

This ID is not guaranteed to persist between device connections or changes in device configuration. To set the input source for a device, set `QTCaptureDeviceInputSourceIdentifierAttribute` to the value returned by this key. This string value can be used in key paths for key value coding, key value observing, and bindings.

Special Considerations

This key, along with the `QTCaptureDeviceInputSourceLocalizedDisplayNameKey` key, comprises the NSDictionary objects describing input sources returned by `QTCaptureDeviceAvailableInputSourcesAttribute`.

Declared In

`QTCaptureDevice.h`

QTCaptureDeviceInputSourceLocalizedDisplayNameKey

The localized display name of an input source, suitable for display in a user interface.

```
QTCaptureDeviceInputSourceLocalizedDisplayNameKey
@"localizedDisplayName"
```

Constants

`QTCaptureDeviceInputSourceLocalizedDisplayNameKey`
This string value can be used in key paths for key value coding, key value observing, and bindings.

Special Considerations

This key, along with the `QTCaptureDeviceInputSourceIdentifierKey` key, comprises the NSDictionary objects describing input sources returned by `QTCaptureDeviceAvailableInputSourcesAttribute`.

Declared In

`QTCaptureDevice.h`

QTCaptureDeviceSuspendedAttribute

Specifies other QTCapture device objects that correspond to the same physical device.

```
QTCaptureDeviceSuspendedAttribute
@"suspended"
```

Constants

`QTCaptureDeviceSuspendedAttribute`
For example, some cameras have built in microphones that appear as separate `QTCaptureDevice` instances in the device list.

Discussion

This string value can be used in key paths for key-value coding, key-value observing, and bindings.

Declared In

`QTCaptureDevice.h`

QTCaptureDeviceLinkedDevicesAttribute

An NSArray of `QTCaptureDevice` instances.

```
QTCaptureDeviceLinkedDevicesAttribute
@"linkedDevices"
```

Constants

`QTCaptureDeviceLinkedDevicesAttribute`
The value is an NSArray of `QTCaptureDevice` instances.

Discussion

This string value can be used in key paths for key-value coding, key-value observing, and bindings.

Declared In

`QTCaptureDevice.h`

QTCaptureDeviceLegacySequenceGrabberAttribute

An NSValue interpreted as a ComponentInstance for the legacy sequence grabber component used by the device.

```
QTCaptureDeviceLegacySequenceGrabberAttribute
@"legacySequenceGrabber"
```

Constants

`QTCaptureDeviceLegacySequenceGrabberAttribute`

Some older devices are opened and controlled by legacy Sequence Grabber components. Applications that need to configure legacy devices directly through the Sequence Grabber configuration dialog can access an open component instance with this attribute.

Discussion

This string value can be used in key paths for key-value coding, key-value observing, and bindings.

Warning: If the device is being used in a capture session, do not modify properties of the returned Sequence Grabber component (by displaying the configuration dialog, for example) while the session is running. Doing so will prevent the capture session from capturing more frames.

Declared In

`QTCaptureDevice.h`

QTCaptureDeviceAVCTransportControlsAttribute

For AVC devices that read data from linear media, such as tapes, specifies the mode and speed at which that media is playing.

```
QTCaptureDeviceAVCTransportControlsAttribute
@"AVCTransportControls"
```

Constants

`QTCaptureDeviceAVCTransportControlsAttribute`

The value is an NSDictionary with keys and values described under QTCaptureDevice AVC Transport Controls.

Discussion

This string value can be used in key paths for key-value coding, key-value observing, and bindings.

Declared In

`QTCaptureDevice.h`

QTCaptureDeviceAVCTransportControlsSpeedKey

Specifies the approximate rate at which the device runs through linear media.

```
QTCaptureDeviceAVCTransportControlsSpeedKey
```

Constants

`QTCaptureDeviceAVCTransportControlsSpeedKey`
The value is an NSNumber interpreted as a `QTCaptureDeviceAVCTransportControlsSpeed`. This is one of the keys that comprise the NSDictionary that specifies the linear media playback mode and rate given by the `QTCaptureDeviceAVCTransportControlsAttribute`.

Declared In

`QTCaptureDevice.h`

QTCaptureDeviceAVCTransportControlsPlaybackMode

A value provided with the `QTCaptureDeviceAVCTransportControlsPlaybackModeKey` key that specifies whether the device previews audio and displays video while it is running through linear media.

```
enum {
    QTCaptureDeviceAVCTransportControlsNotPlayingMode       = 0,
    QTCaptureDeviceAVCTransportControlsPlayingMode          = 1
};
```

Constants

`QTCaptureDeviceAVCTransportControlsPlaybackMode`
`QTCaptureDeviceAVCTransportControlsNotPlayingMode` is equivalent to the Play mode on most cameras and tape decks, while `QTCaptureDeviceAVCTransportControlsPlayingMode` is equivalent to Stop on most cameras and tape decks. If the device is connected to a session, the video at the current location on the device's media will only be captured if this attribute is set to `QTCaptureDeviceAVCTransportControlsNotPlayingMode`.

Declared In

`QTCaptureDevice.h`

QTCaptureDeviceAVCTransportControlsSpeed

A value provided with the `QTCaptureDeviceAVCTransportControlsSpeedKey` key that specifies whether the device previews audio and displays video while it is running through linear media.

```
enum {
    QTCaptureDeviceAVCTransportControlsFastestReverseSpeed  = -19000,
    QTCaptureDeviceAVCTransportControlsVeryFastReverseSpeed = -16000,
    QTCaptureDeviceAVCTransportControlsFastReverseSpeed     = -13000,
    QTCaptureDeviceAVCTransportControlsNormalReverseSpeed   = -10000,
    QTCaptureDeviceAVCTransportControlsSlowReverseSpeed     = -7000,
    QTCaptureDeviceAVCTransportControlsVerySlowReverseSpeed = -4000,
    QTCaptureDeviceAVCTransportControlsSlowestReverseSpeed  = -1000,
    QTCaptureDeviceAVCTransportControlsStoppedSpeed         = 0,
    QTCaptureDeviceAVCTransportControlsSlowestForwardSpeed  = 1000,
    QTCaptureDeviceAVCTransportControlsVerySlowForwardSpeed = 4000,
    QTCaptureDeviceAVCTransportControlsSlowForwardSpeed     = 7000,
    QTCaptureDeviceAVCTransportControlsNormalForwardSpeed   = 10000,
    QTCaptureDeviceAVCTransportControlsFastForwardSpeed     = 13000,
    QTCaptureDeviceAVCTransportControlsVeryFastForwardSpeed = 16000,
    QTCaptureDeviceAVCTransportControlsFastestForwardSpeed  = 19000,
};
```

Constants

`QTCaptureDeviceAVCTransportControlsSpeed`

The actual speed at which the media is run for a given value will depend on the manufacturer and model of the device, as well as the value of `QTCaptureDeviceAVCTransportControlsPlaybackModeKey` (in general, when `QTCaptureDeviceAVCTransportControlsPlaybackModeKey` is set to `QTCaptureDeviceAVCTransportControlsNotPlayingMode`, the media will run faster than when it is set to `QTCaptureDeviceAVCTransportControlsPlayingMode`). For most cameras and tape decks, different speeds will affect the media speed, as shown in Table 6-2.

Special Considerations

Table 6-2 Speed variations for media

QTCaptureDeviceAVCTransportControlsPlaybackModeKey	QTCaptureDeviceAVCTransportControlsPlaybackMode is QTCaptureDeviceAVCTransportControlsPlayingMod
QTCaptureDeviceAVCTransportControlsFastestReverseSpeed	Media runs in reverse at greater than normal speed.
QTCaptureDeviceAVCTransportControlsVeryFastReverseSpeed	Media runs in reverse at greater than normal speed.
QTCaptureDeviceAVCTransportControlsFastReverseSpeed	Media runs in reverse at greater than normal speed.
QTCaptureDeviceAVCTransportControlsNormalReverseSpeed	Media runs in reverse at normal speed.
QTCaptureDeviceAVCTransportControlsSlowReverseSpeed	Media runs in reverse at less than normal speed.
QTCaptureDeviceAVCTransportControlsVerySlowReverseSpeed	Media runs in reverse at less than normal speed.
QTCaptureDeviceAVCTransportControlsSlowestReverseSpeed	Media runs in reverse at less than normal speed.
QTCaptureDeviceAVCTransportControlsStoppedSpeed	Media is paused.
QTCaptureDeviceAVCTransportControlsStoppedSpeed	Media runs forward at less than normal speed.
QTCaptureDeviceAVCTransportControlsSlowestForwardSpeed	Media runs forward at less than normal speed.
QTCaptureDeviceAVCTransportControlsSlowForwardSpeed	Media runs forward at less than normal speed.
QTCaptureDeviceAVCTransportControlsNormalForwardSpeed	Media runs forward at normal speed.

QTCaptureDeviceAVCTransportControlsPlaybackModeKey	QTCaptureDeviceAVCTransportControlsPlaybackMode is QTCaptureDeviceAVCTransportControlsPlayingMod
QTCaptureDeviceAVCTransportControlsFastForwardSpeed	Media runs forward at greater than than normal speed.
QTCaptureDeviceAVCTransportControlsVeryFastForwardSpeed	Media runs forward at greater than than normal speed..
QTCaptureDeviceAVCTransportControlsFastestForwardSpeed	Media runs forward at greater than than normal speed.

Declared In
`QTCaptureDevice.h`

Notifications

QTCaptureDeviceWasConnectedNotification

Posted when a device is connected or turned on.

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

Declared In
`QTCaptureDevice.h`

QTCaptureDeviceWasDisconnectedNotification

Posted when a device is disconnected or turned off.

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

Declared In
`QTCaptureDevice.h`

QTCaptureDeviceFormatDescriptionsWillChangeNotification

Posted when the device's formats that are returned by the `formatDescriptions` method are about to change.

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

Declared In
`QTCaptureDevice.h`

QTCaptureDeviceFormatDescriptionsDidChangeNotification

Posted when the device's formats that are returned by the `formatDescriptions` method have just changed.

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

Declared In
`QTCaptureDevice.h`

QTCaptureDeviceAttributeWillChangeNotification

Posted when one of the device's attributes is about to change.

Important: The notification's user info dictionary will contain the attribute key of the changed attribute for the key `QTCaptureDeviceChangedAttributeKey`.

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

Declared In
`QTCaptureDevice.h`

QTCaptureDeviceAttributeDidChangeNotification

Posted when the one of device's attributes has changed.

Important: The notification's user info dictionary will contain the attribute key of the changed attribute for the key `QTCaptureDeviceChangedAttributeKey`.

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

Declared In
`QTCaptureDevice.h`

QTCaptureDeviceInput Class Reference

Inherits from	QTCaptureInput : NSObject
Conforms to	NSObject (NSObject)
Framework	Library/Frameworks/QTKit.framework
Declared in	QTKit/QTCaptureDeviceInput.h
Availability	Available in QuickTime 7.2.1 and later.
Related sample code	LiveVideoMixer3 QT Capture Widget QTRecorder

Overview

This class represents the input source for media devices, such as cameras and microphones. Instances of `QTCaptureDeviceInput` are input sources for `QTCaptureSession` that provide media data from devices connected to the computer. Devices used with `QTCaptureDeviceInput` can be found using the `QTCaptureDevice` class. A `QTCaptureDevice` must be successfully opened using the `open:` method before being used in a `QTCaptureDeviceInput`.

Tasks

Capturing Device Input

`- device` (page 62)
Returns the device associated with the receiver.

`- initWithDevice:` (page 62)
Returns an instance of `QTCaptureDeviceInput` associated with the given device.

`+ deviceInputWithDevice:` (page 62)
Returns an autoreleased instance of `QTCaptureDeviceInput` associated with the given device.

Class Methods

deviceInputWithDevice:

Returns an autoreleased instance of `QTCaptureDeviceInput` associated with the given device.

```
+ (QTCaptureDeviceInput *)deviceInputWithDevice:(QTCaptureDevice *)device
```

Parameters

device

A `QTCaptureDevice` for the device to be associated with the receiver. The device must have been previously opened using the `open:` method or this method will throw an NSInvalidArgumentException.

Return Value

A `QTCaptureDeviceInput` instance associated with the device.

Availability

Mac OS X v10.5 and later.

Related Sample Code

LiveVideoMixer3

Declared In

`QTCaptureDeviceInput.h`

Instance Methods

device

Returns the device associated with the receiver.

```
- (QTCaptureDevice *)device
```

Return Value

If there is a device associated with the receiver, returns a corresponding instance of `QTCaptureDevice`. Otherwise returns `NIL`.

Availability

Mac OS X v10.5 and later.

Declared In

`QTCaptureDeviceInput.h`

initWithDevice:

Returns an instance of `QTCaptureDeviceInput` associated with the given device.

```
- (id)initWithDevice:(QTCaptureDevice *)device
```

Parameters

device

A `QTCaptureDevice` object for the device to be associated with the receiver. The device must have been previously opened using the `open:` method, or else this method will throw an NSInvalidArgumentException.

Return Value

A `QTCaptureDeviceInput` instance associated with the device.

Availability

Mac OS X v10.5 and later.

Declared In

`QTCaptureDeviceInput.h`

QTCaptureFileOutput Reference

Inherits from	QTCaptureOutput : NSObject
Conforms to	NSObject (NSObject)
Framework	Library/Frameworks/QTKit.framework
Declared in	QTKit/QTCaptureFileOutput.h
Availability	Available in QuickTime 7.2.1 and later.
Related sample code	QT Capture Widget QTRecorder

Overview

This is an abstract superclass output destination for `QTCaptureSession` that writes captured media to files. This superclass defines the interface for outputs that record media samples to files. File outputs are designated a recording output file using the `recordToFileURL:` and `recordToFileURL:bufferDestination:` methods. On successive invocations of these methods, the output file can by changed dynamically without losing media samples. A file output can also be set to not record incoming frames (the default behavior when an output is first initialized) by passing `NIL` as the output file URL. Because files are recorded in the background, applications will generally need to set a delegate for a file output so that they can be notified when recorded files are started and finished. The file output delegate can also be used to control recording for exact media samples by implementing the `captureOutput:didOutputSampleBuffer:fromConnection:` method. Currently, the only concrete subclass of this class is `QTCaptureMovieFileOutput`.

Tasks

Recording File Outputs

- `outputFileURL` (page 73)
  Returns the file written to by the receiver.
- `recordToOutputFileURL:` (page 74)
  Sets the file written to by the receiver.
- `recordToOutputFileURL:bufferDestination:` (page 74)
  Sets the file written to by the receiver, specifying where the sample buffer currently in flight should be recorded.

- `recordedDuration` (page 73)
  Returns the duration of the media recorded by the receiver.
- `recordedFileSize` (page 73)
  Returns the size, in bytes, of the data recorded by the receiver to output files.
- `maximumRecordedDuration` (page 72)
  Returns the maximum duration of the media that should be recorded by the receiver.
- `setMaximumRecordedDuration:` (page 76)
  Sets the maximum duration of the media that should be recorded by the receiver.
- `maximumRecordedFileSize` (page 72)
  Returns the maximum file size, in bytes, of the file that should be recorded by the receiver.
- `setMaximumRecordedFileSize:` (page 76)
  Sets the maximum file size, in bytes, of the file that should be recorded by the receiver.
- `compressionOptionsForConnection:` (page 71)
  Returns the options the receiver uses to compress media on the given connection as it is being captured.
- `setCompressionOptions:forConnection:` (page 75)
  Sets the options the receiver uses to compress media on the given connection as it is being captured.
- `delegate` (page 72)
  Returns the receiver's delegate.
- `setDelegate:` (page 75)
  Sets the receiver's delegate.

Methods Implemented by the Delegate

- `captureOutput:didOutputSampleBuffer:fromConnection:` (page 67)
  Gives the delegate the opportunity to inpect samples as they are received by the output and start and stop capturing at exact times.
- `captureOutput:willStartRecordingToOutputFileURL:forConnections:` (page 71)
  Informs the delegate when the output is about to start writing to a file.
- `captureOutput:didStartRecordingToOutputFileURL:forConnections:` (page 68)
  Informs the delegate when the output has started writing to a file.
- `captureOutput:shouldChangeOutputFileAtURL:forConnections:` (page 69)
  Gives the delegate the opportunity to determine what should happen when an output file has reached a soft limit.
- `captureOutput:mustChangeOutputFileAtURL:forConnections:dueToError:` (page 68)
  Informs the delegate when an output file can no longer be written using the incoming media.
- `captureOutput:willFinishRecordingToOutputFileAtURL:forConnections:dueToError:` (page 70)
  Informs the delegate whenthe output will stop writing new samples to a file.
- `captureOutput:didFinishRecordingToOutputFileAtURL:forConnections:dueToError:` (page 67)
  Informs the delegate when an output file is ready to be opened by applications.

Instance Methods

captureOutput:didFinishRecordingToOutputFileAtURL:forConnections:dueToError:

Informs the delegate when an output file is ready to be opened by applications.

```
- (void)captureOutput:(QTCaptureFileOutput *)captureOutput
    didFinishRecordingToOutputFileAtURL:(NSURL *)outputFileURL
    forConnections:(NSArray *)connections
    dueToError:(NSError *)error
```

Parameters

captureOutput

The capture file output that has finished writing the file.

outputURL

The file URL of the file that has been written.

connections

An array of QTCaptureConnection objects owned by the receiver that provided the data that was written to the file.

error

An error describing what caused the file to stop recording, or `NIL` if there was no error.

Discussion

Whenever the receiver's `recordToOutputFileURL:` or `recordToOutputFileURL:bufferDestination:` method is called during recording, they return immediately, finishing any pending file writing in the background. Delegates must implement this method to be informed when those files are finished and ready to be opened by applications.

Warning: Applications should not assume that this method will be called on the main thread.

Availability

Mac OS X v10.5 and later.

captureOutput:didOutputSampleBuffer:fromConnection:

Gives the delegate the opportunity to inpect samples as they are received by the output and start and stop capturing at exact times.

```
- (void)captureOutput:(QTCaptureFileOutput *)captureOutput
    didOutputSampleBuffer:(QTSampleBuffer *)sampleBuffer
    fromConnection:(QTCaptureConnection *)connection
```

Parameters

captureOutput

The capture file output that is receiving the media data.

sampleBuffer

A sample buffer object containing the sample data and additional information about the sample, such as its time code and record date.

connection

The capture connection object owned by the receiver that is receiving the sample data.

Discussion

This method is called whenever the file output receives a single media sample (a single video frame, for example) through the given connection. This gives delegates an opportunity to start and stop capturing or change output files at an exact sample. Calls to the file output's `recordToOutputFileURL:` and `recordToOutputFileURL:bufferDestination:` methods are guaranteed to include the received sample if called from within this method. Delegates can gather information particular to the sample, such as its record time, and whether it marks a scene change, by inspecting the `sampleInfo` object. Sample buffers always contain a single frame of video if called from this method but may also contain multiple packets of audio. For B-frame video formats, this method is always called in presentation order.

Warning: Applications should not assume that this method will be called on the main thread. In addition, this method is called periodically, so it must be efficient to prevent capture performance problems.

Availability

Mac OS X v10.5 and later.

captureOutput:didStartRecordingToOutputFileURL:forConnections:

Informs the delegate when the output has started writing to a file.

```
- (void)captureOutput:(QTCaptureFileOutput *)captureOutput
    didStartRecordingToOutputFileURL:(NSURL *)fileURL
    forConnections:(NSArray *)connections
```

Parameters

captureOutput

The capture file output that started writing the file.

outputURL

The file URL of the file being written.

connections

An array of `QTCaptureConnection` objects owned by the receiver that provided the data that is being written to the file.

Discussion

Applications should not assume that this method will be called on the main thread.

Availability

Mac OS X v10.5 and later.

captureOutput:mustChangeOutputFileAtURL:forConnections:dueToError:

Informs the delegate when an output file can no longer be written using the incoming media.

```
- (void)captureOutput:(QTCaptureFileOutput *)captureOutput
    mustChangeOutputFileAtURL:(NSURL *)outputFileURL
    forConnections:(NSArray *)connections
    dueToError:(NSError *)error
```

Parameters

captureOutput

The capture file output that must finish writing the file.

outputURL

The file URL of the file that is being written.

connections

An array of `QTCaptureConnection` objects owned by the receiver that provided the data that is being written to the file.

error

The error that caused the output to require that a new file be written.

Discussion

This method is called if the existing output file for that connection can no longer be written (this occurs, for example, if the stream format of the samples has changed, the output is receiving invalid samples, or there is insufficient disk space remaining on the output file's disk). Delegates implementing this method can start recording on a new file using `recordToOutputFileURL:` or `recordToOutputFileURL:bufferDestination:` to ensure that incoming data will continue to be recorded. If the delegate does not implement this method or does not set new output files for the given connections, recording stops automatically.

Warning: Applications should not assume that this method will be called on the main thread.

Availability

Mac OS X v10.5 and later.

captureOutput:shouldChangeOutputFileAtURL:forConnections:

Gives the delegate the opportunity to determine what should happen when an output file has reached a soft limit.

```
- (BOOL)captureOutput:(QTCaptureFileOutput *)captureOutput
    shouldChangeOutputFileAtURL:(NSURL *)outputFileURL
    forConnections:(NSArray *)connections
    dueToError:(NSError *)error
```

Parameters

captureOutput

The capture file output that should finish writing the file.

outputURL

The file URL of the file that is being written.

connections

An array of `QTCaptureConnection` objects owned by the receiver that provided the data that is being written to the file.

error

The error that caused the output to suggest that a new file be written.

Return Value

Delegates should return YES if the current file should no longer be written, or NO if the current file should continue to be written.

Discussion
This method is called when the file output encounters a problem, such as dropped media samples (indicated by a `QTErrorMediaDiscontinuity` error), that doesn't require that recording stop but may be a reason for some applications to change files or stop recording. For example, applications concerned with recording every frame of video or every sample of audio may want to treat such problems as error conditions rather than ignoring them. This method is also called when the file output reaches a soft limit, namely one of the limits set using the `setMaximumRecordedDuration:` and `setMaximumRecordedFileSize:` methods. Delegates should check the value of the error parameter to see what kind of error caused this delegate method to be called. If the delegate returns NO, the output will continue writing the same file. If the delegate returns YES and doesn't set a new output file, `captureOutput:mustChangeOutputFileAtURL:forConnections:dueToError:` will be called. If the delegate returns YES and sets a new output file, recording will continue on the new file. If the delegate does not respond to this method, the file output will automatically continue recording when it encounters one of these errors, unless it is a `QTErrorMaximumDurationReached or QTErrorMaximumFileSizeReached` error, in which case the file output will automatically stop recording.


Warning: Applications should not assume that this method will be called on the main thread.

Availability
Mac OS X v10.5 and later.

captureOutput:willFinishRecordingToOutputFileAtURL:forConnections:dueToError:

Informs the delegate whenthe output will stop writing new samples to a file.

```
- (void)captureOutput:(QTCaptureFileOutput *)captureOutput
    willFinishRecordingToOutputFileAtURL:(NSURL *)outputFileURL
    forConnections:(NSArray *)connections
    dueToError:(NSError *)error
```

Parameters

captureOutput
The capture file output that will finish writing the file.

outputURL
The file URL of the file that is being written.

connections
An array of `QTCaptureConnection` objects owned by the receiver that provided the data that is being written to the file.

error
An error describing what caused the file to stop recording, or nil if there was no error.

Discussion
This method is called when the file output will stop recording new samples to the file at `outputFileURL`, either because `recordToFile:` or `recordToFile:bufferDestination:` was called, or because an error, described by the error parameter, occurred (if no error occurred, the error parameter will be `NIL`). Delegates should also implement `captureOutput:didFinishRecordingToOutputFileAtURL:forConnections:dueToError:` to be notified when the file is ready to be opened by applications.

Warning: Applications should not assume that this method will be called on the main thread.

Availability
Mac OS X v10.5 and later.

captureOutput:willStartRecordingToOutputFileURL:forConnections:

Informs the delegate when the output is about to start writing to a file.

```
- (void)captureOutput:(QTCaptureFileOutput *)captureOutput
    willStartRecordingToOutputFileURL:(NSURL *)fileURL
    forConnections:(NSArray *)connections
```

Parameters

captureOutput
The capture file output that will start writing the file.

outputURL
The file URL of the file that will be written.

connections
An array of `QTCaptureConnection` objects owned by the receiver that provided the data that will be written to the file.

Discussion
Applications should not assume that this method will be called on the main thread.

Availability
Mac OS X v10.5 and later.

compressionOptionsForConnection:

Returns the options the receiver uses to compress media on the given connection as it is being captured.

```
- (QTCompressionOptions *)compressionOptionsForConnection:(QTCaptureConnection
    *)connection
```

Parameters

connection
The connection containing the media to be compressed.

Return Value
A `QTCompressionOptions` object detailing the options being used to compress captured media on the given connection, or `NIL` if the media will not be recompressed.

Discussion
This method returns the options for compressing media set with the `setCompressionOptions:forConnection:` method. If the receiver should not recompress the output media, this method returns `NIL`. The default value is `NIL`.

Availability
Mac OS X v10.5 and later.

Declared In
`QTCaptureFileOutput.h`

delegate

Returns the receiver's delegate.

```
- (id)delegate
```

Availability
Mac OS X v10.5 and later.

Declared In
`QTCaptureFileOutput.h`

maximumRecordedDuration

Returns the maximum duration of the media that should be recorded by the receiver.

```
- (QTTime)maximumRecordedDuration
```

Return Value
The maximum time to be recorded, or `QTZeroTime` if there is no limit set.

Discussion
This method returns a soft limit on the duration of recorded files set by `setMaximumRecordedDuration:`. Delegates can determine what to do when the limit is reached by implementing the `captureOutput:shouldChangeOutputFileAtURL:forConnections:dueToError:` method. By default, the current output file is set to `NIL` when the limit is reached.

Availability
Mac OS X v10.5 and later.

Declared In
`QTCaptureFileOutput.h`

maximumRecordedFileSize

Returns the maximum file size, in bytes, of the file that should be recorded by the receiver.

```
- (UInt64)maximumRecordedFileSize
```

Return Value
The maximum file size, in bytes, to be recorded, or 0 if there is no limit set.

Discussion
This method returns a soft limit on the duration of recorded files set by `setMaximumRecordedFileSize:`. Delegates can determine what to do when the limit is reached by implementing the `captureOutput:shouldChangeOutputFileAtURL:forConnections:dueToError:` method. By default, the current output file is set to `NIL` when the limit is reached.

Availability
Mac OS X v10.5 and later.

Declared In
`QTCaptureFileOutput.h`

outputFileURL

Returns the file written to by the receiver.

```
- (NSURL *)outputFileURL
```

Return Value
An `NSURL` object containing the file URL of the file currently being written by the receiver. Returns `NIL` if the reciever is not recording to any file.

Availability
Mac OS X v10.5 and later.

Declared In
`QTCaptureFileOutput.h`

recordedDuration

Returns the duration of the media recorded by the receiver.

```
- (QTTime)recordedDuration
```

Return Value
The recorded time.

Discussion
If recording is in progess, this method returns the total time recorded so far. Otherwise, this method returns the time recorded in the most recent recording.

Availability
Mac OS X v10.5 and later.

Declared In
`QTCaptureFileOutput.h`

recordedFileSize

Returns the size, in bytes, of the data recorded by the receiver to output files.

```
- (UInt64)recordedFileSize
```

Return Value
The recorded size, in bytes.

Discussion
If a recording is in progess, this method returns the size in bytes of the data recorded so far. Otherwise, this method returns the size in the most recent recording.

Availability
Mac OS X v10.5 and later.

Declared In
QTCaptureFileOutput.h

recordToOutputFileURL:

Sets the file written to by the receiver.

```
- (void)recordToOutputFileURL:(NSURL *)outputURL
```

Parameters

outputURL

An NSURL object containing the URL of the output file, or NIL if the receiver should not record to any file. This method throws an NSInvalidArgumentException if the URL is not a valid file URL.

Discussion
The method sets the file URL to which the receiver is currently writing output media. If a file at the given URL already exists when capturing starts, the existing file is overwritten. If NIL is passed as the file URL, the receiver will stop recording to any file. If this method is invoked while an existing output file was already being recorded, no media samples are discarded between the old file and the new file. The sample buffer currently in flight when this method is called will always be written to the new file. Applications can specify where the sample buffer currently in flight will be recorded using the recordToOutputFileURL:bufferDestination: method. When the new file is set, applications cannot open the old file until it has finished recording in the background. Delegates should implement the captureOutput:didFinishRecordingToOutputFileAtURL:forConnections:dueToError: to be notified when the file is ready to be opened.

Availability
Mac OS X v10.5 and later.

Declared In
QTCaptureFileOutput.h

recordToOutputFileURL:bufferDestination:

Sets the file written to by the receiver, specifying where the sample buffer currently in flight should be recorded.

```
- (void)recordToOutputFileURL:(NSURL *)url
    bufferDestination:(QTCaptureFileOutputBufferDestination)bufferDestination
```

Parameters

outputURL

An NSURL object containing the URL of the output file, or NIL if the receiver should not record to any file. This method throws an NSInvalidArgumentException if the URL is not a valid file URL.

bufferDestination

A buffer destination specifying which file should contain the buffer currently in flight.

Discussion
The method sets the file URL to which the receiver is currently writing output media. If a file at the given URL already exists when capturing starts, the existing file will be overwritten. If `NIL` is passed as the file URL, the receiver will stop recording to any file. If this method is invoked while an existing output file was already being recorded, no media samples will be discarded between the old file and the new file. Applications can specify where the sample buffer currently in flight will be recorded using the `bufferDestination` argument. When the new file is set, applications will not be able to open the old file until it has finished recording in the background. Delegates should implement the `captureOutput:didFinishRecordingToOutputFileAtURL:forConnections:dueToError:` method to be notified when the file is ready to be opened.

Availability
Mac OS X v10.5 and later.

Declared In
`QTCaptureFileOutput.h`

setCompressionOptions:forConnection:

Sets the options the receiver uses to compress media on the given connection as it is being captured.

```
- (void)setCompressionOptions:(QTCompressionOptions *)compressionOptions
    forConnection:(QTCaptureConnection *)connection
```

Parameters

compressionOptionscompressionOptions
A `QTCompressionOptions` object detailing the options being used to compress captured media, or `NIL` if the media should not be recompressed.

connection
The connection containing the media to be compressed.

Discussion
This method sets the options for compressing media as it is being captured. If compression cannot be performed in real time, the receiver will drop frames in order to remain synchronized with the session. If the receiver does not recompress the output media, this method should be passed `NIL`. The default value is `NIL`.

Availability
Mac OS X v10.5 and later.

Declared In
`QTCaptureFileOutput.h`

setDelegate:

Sets the receiver's delegate.

```
- (void)setDelegate:(id)delegate
```

Availability
Mac OS X v10.5 and later.

Declared In
`QTCaptureFileOutput.h`

setMaximumRecordedDuration:

Sets the maximum duration of the media that should be recorded by the receiver.

```
- (void)setMaximumRecordedDuration:(QTTime)maximumRecordedDuration
```

Parameters

maximumRecordedDuration

The maximum time to be recorded, or `QTZeroTime` if there should be no limit.

Discussion

This method sets a soft limit on the duration of recorded files. Delegates can determine what to do when the limit is reached by implementing the `captureOutput:shouldChangeOutputFileAtURL:forConnections:dueToError:` method. By default, the current output file is set to `NIL` when the limit is reached.

Availability

Mac OS X v10.5 and later.

Declared In

`QTCaptureFileOutput.h`

setMaximumRecordedFileSize:

Sets the maximum file size, in bytes, of the file that should be recorded by the receiver.

```
- (void)setMaximumRecordedFileSize:(UInt64)maximumRecordedFileSize
```

Parameters

maximumRecordedFileSize

The maximum size, in bytes, to be recorded, or 0 is there should be no limit.

Discussion

This method sets a soft limit on the size of recorded files. Delegates can determine what to do when the limit is reached by implementing the `captureOutput:shouldChangeOutputFileAtURL:forConnections:dueToError:` method. By default, the current output file is set to `NIL` when the limit is reached.

Availability

Mac OS X v10.5 and later.

Declared In

`QTCaptureFileOutput.h`

Constants

QTCaptureFileOutputBufferDestination

Specifies where the media sample buffer currently in flight should be written when changing output files.

```
enum {
    QTCaptureFileOutputBufferDestinationNewFile = 0,
    QTCaptureFileOutputBufferDestinationOldFile = 1
};
```

Constants

`QTCaptureFileOutputBufferDestination`

`QTCaptureFileOutputBufferDestinationNewFile` tells the output to include the buffer currently in flight in the old file. `QTCaptureFileOutputBufferDestinationOldFile` tells the output to include the buffer currently in flight in the new file.

Declared In

`QTCaptureFileOutput.h`

QTCaptureInput Class Reference

Inherits from	NSObject
Conforms to	NSObject (NSObject)
Framework	Library/Frameworks/QTKit.framework
Declared in	QTKit/QTCaptureInput.h
Availability	Available in QuickTime 7.2.1 and later.

Overview

This class provides input source connections for a `QTCaptureSession`. `QTCaptureInput` is an abstract class that provides an interface for connecting capture input sources, such as cameras, to a `QTCaptureSession`. An input source can have multiple connections. For instance, many cameras output both audio and video streams. Each connection owned by a `QTCaptureInput` instance is described by a `QTCaptureConnection`.

Tasks

Capturing Input

`- connections` (page 79)
Returns an array of connections owned by the receiver.

Instance Methods

connections

Returns an array of connections owned by the receiver.

```
- (NSArray *)connections
```

Return Value
An NSArray of `QTCaptureConnection` instances.

Discussion
For each connection owned by the receiver, this method returns a `QTCaptureConnection` describing the media type, format, and other attributes of the connection.

Availability
Mac OS X v10.5 and later.

Declared In
`QTCaptureInput.h`

QTCaptureLayer Reference

Inherits from	CALayer : NSObject
Conforms to	NSCoding (CALayer) CAMediaTiming (CALayer) NSObject (NSObject)
Framework	Library/Frameworks/QTKit.framework
Declared in	QTKit/QTCaptureLayer.h
Availability	Available in QuickTime 7.2.1 and later.

Overview

This class provides a layer that displays video frames currently being captured from a device attached to the computer, and is intended to provide support for Core Animation, that is, drawing the contents of a capture session into a layer. Note that this class requires rendering using visual contexts.

Tasks

Creating Capture Layers

`+ layerWithSession:` (page 82)
Creates an autoreleased `QTCaptureLayer` associated with the specified `QTCaptureSession` object.

`- initWithSession:` (page 82)
Creates a `QTCaptureLayer` associated with the specified `QTCaptureSession` object.

`- session` (page 82)
Returns the capture session associated with a `QTCaptureLayer` object.

`- setSession:` (page 83)
Sets or resets the capture session associated with a `QTCaptureLayer` object.

Class Methods

layerWithSession:

Creates an autoreleased `QTCaptureLayer` associated with the specified `QTCaptureSession` object.

```
+ (id)layerWithSession:(QTCaptureSession *)session
```

Discussion
By default, the movie starts playing immediately at rate 1.0 from the beginning of the movie. These default characteristics can be modified by setting layer properties or movie properties

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

Declared In
`QTCaptureLayer.h`

Instance Methods

initWithSession:

Creates a `QTCaptureLayer` associated with the specified `QTCaptureSession` object.

```
- (id)initWithSession:(QTCaptureSession *)session
```

Discussion
By default, the movie starts playing immediately at rate 1.0 from the beginning of the movie. These default characteristics can be modified by setting layer properties or movie properties.

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

Declared In
`QTCaptureLayer.h`

session

Returns the capture session associated with a `QTCaptureLayer` object.

```
- (QTCaptureSession *)session
```

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

Declared In
`QTCaptureLayer.h`

setSession:

Sets or resets the capture session associated with a `QTCaptureLayer` object.

```
- (void)setSession:(QTCaptureSession *)session
```

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

Declared In
`QTCaptureLayer.h`

QTCaptureMovieFileOutput Reference

Inherits from	QTCaptureFileOutput : QTCaptureOutput : NSObject
Conforms to	NSObject (NSObject)
Framework	Library/Frameworks/QTKit.framework
Declared in	QTKit/QTCaptureMovieFileOutput.h
Availability	Available in QuickTime 7.2.1 and later.
Related sample code	QT Capture Widget QTRecorder

Overview

This class represents an output destination for `QTCaptureSession` that writes captured media to QuickTime movie files. A `QTCaptureMovieFileOutput` instance writes the media captured by its connected capture session to QuickTime movie files. The methods implemented by this class are described in the *QTCaptureFileOutput Reference*.

QTCaptureOutput Class Reference

Inherits from	NSObject
Conforms to	NSObject (NSObject)
Framework	Library/Frameworks/QTKit.framework
Declared in	QTKit/QTCaptureOutput.h
Availability	Available in QuickTime 7.2.1 and later.

Overview

`QTCaptureOutput` is an abstract class that provides an interface for connecting capture output destinations, such as QuickTime files and video previews, to a `QTCaptureSession`. Like a `QTCaptureInput`, a `QTCaptureOutput` can have multiple connections represented by `QTCaptureConnection` objects, one for each stream of media that it receives. Unlike a QTCaptureInput however, a `QTCaptureOutput` does not have any connections when it is first created. When an output is added to a `QTCaptureSession`, it creates connections as appropriate so that the session has a destination for all of its input media.

Tasks

Capturing Connections

`- connections` (page 87)
Returns an array of connections owned by the receiver that are currently connected to a capture session.

Instance Methods

connections

Returns an array of connections owned by the receiver that are currently connected to a capture session.

```
- (NSArray *)connections
```

Return Value
An array of `QTCaptureConnection` instances owned by the receiver that are currently connected to a capture session.

Discussion
`QTCaptureOutputs` create a new output connection for each input connection of a matching media type connected to the capture session. This method returns an array of connections owned by the receiver that are currently connected to the capture session's input connections.

Availability
Mac OS X v10.5 and later.

Declared In
`QTCaptureOutput.h`

QTCaptureSession Class Reference

Inherits from	NSObject
Conforms to	NSCoding NSObject (NSObject)
Framework	Library/Frameworks/QTKit.framework
Declared in	QTKit/QTCaptureSession.h
Availability	Available in QuickTime 7.2.1 and later.
Related sample code	LiveVideoMixer3 QT Capture Widget QTRecorder

Overview

This class is the primary interface for capturing media streams. A `QTCaptureSession` instance provides an interface for connecting capture input sources, subclasses `QTCaptureInput` to output destinations and subclasses of `QTCaptureOutput`. In addition to managing the connections between inputs and outputs, instances of `QTCaptureSession` also manage when a capture is running.

Tasks

Controlling Receiver Capture

- `isRunning` (page 92)
  Returns whether the receiver is running.
- `startRunning` (page 93)
  Tells the receiver to start capturing data from its inputs and sending data to its outputs.
- `stopRunning` (page 93)
  Tells the receiver to stop capturing data from its inputs and sending data to its outputs.

Working with Receiver Inputs and Outputs

- `addInput:error:` (page 90)
  Adds an input to the receiver.

- `addOutput:error:` (page 91)
  Adds an output to the receiver.
- `inputs` (page 91)
  Returns an array of inputs connected to the receiver.
- `outputs` (page 92)
  Returns an array of outputs connected to the receiver.
- `removeInput:` (page 92)
  Removes an input from the receiver.
- `removeOutput:` (page 93)
  Removes an output from the receiver.

Instance Methods

addInput:error:

Adds an input to the receiver.

```
- (BOOL)addInput:(QTCaptureInput *)input
    error:(NSError **)errorPtr
```

Parameters

input
  The capture input to be connected to the receiver.

errorPtr
  After the method returns, if this parameter is not equal to `NIL`, it points to an error describing why the input could not be added, or points to `NIL` if the input was added successfully.

Return Value
Returns YES if the input was added successfully, or has already been added to the receiver. Returns NO if the input could not be added.

Discussion
This method adds a `QTCaptureInput` to the receiver's list of inputs, adding each of its connections to the capture session as media sources. If there are any outputs already added to the receiver after an input is successfully added, each output creates an additional `QTCaptureConnection` for each stream of media that it can read from the session and adds it to the list returned by its `connections` method. If an input is added successfully, it is retained by the receiver and this method returns `YES`. If an input is added more than once, this method does nothing and returns `YES`. If an input cannot be added, this method returns `NO` and returns an NSError in the location pointed to by errorPtr. The same input cannot be added to more than one capture session. If a client tries to add an input that has already been added to another session, the method throws an NSInvalidArgumentException.

Availability
Mac OS X v10.5 and later.

Declared In
`QTCaptureSession.h`

addOutput:error:

Adds an output to the receiver.

```
- (BOOL)addOutput:(QTCaptureOutput *)output
    error:(NSError **)errorPtr
```

Parameters

output

The `QTCaptureOutput` instance connection to be connected to the receiver.

errorPtr

If not equal to `NIL`, points to an error describing why the output could not be added, or points to `NIL` if the output was added successfully.

Return Value
Returns `YES` if the output was added successfully, or has already been added to the receiver. Returns `NO` if the output could not be added.

Discussion
This method adds a `QTCaptureOutput` to the receiver's list of outputs. After an output is successfully added to a session, it creates one `QTCaptureConnection` for each stream of media that it can read from the session and adds it to the list returned by its `connections` method. If an input is added successfully, it is retained by the receiver and this method returns `YES`. If an output is added more than once, this method does nothing and returns `YES`. If an output cannot be added, this method returns `NO` and returns an NSError in the location pointed to by errorPtr. The same output cannot be added to more than one capture session. If a client tries to add an output that has already been added to another session, the method throws an NSInvalidArgumentException.

Availability
Mac OS X v10.5 and later.

Declared In
`QTCaptureSession.h`

inputs

Returns an array of inputs connected to the receiver.

```
- (NSArray *)inputs
```

Return Value
An array of `QTCaptureInput` instances.

Discussion
A capture session can have one or more input sources, which are instances of `QTCaptureInput`.

Availability
Mac OS X v10.5 and later.

Declared In
`QTCaptureSession.h`

isRunning

Returns whether the receiver is running.

```
- (BOOL)isRunning;
```

Return Value
Returns YES if the receiver is running. NO otherwise.

Discussion
When a `QTCaptureSession` is running, it continuously reads media from its inputs and sends it to those outputs currently accepting data. When data does not need to be sent to file outputs, previews, and other outputs, capture sessions should not be running so that the overhead from capturing not affect application performance. By default, capture sessions are not running.

Availability
Mac OS X v10.5 and later.

Declared In
`QTCaptureSession.h`

outputs

Returns an array of outputs connected to the receiver.

```
- (NSArray *)outputs
```

Return Value
An array of `QTCaptureOutput` instances.

Discussion
A capture session can have one or more output destinations, which are instances of `QTCaptureOutput`.

Availability
Mac OS X v10.5 and later.

Declared In
`QTCaptureSession.h`

removeInput:

Removes an input from the receiver.

```
- (void)removeInput:(QTCaptureInput *)input
```

Parameters
input
The `QTCaptureInput` to be removed from the receiver.

Discussion
This method removes a `QTCaptureInput` added with `addInput:error:` and releases it.

Availability
Mac OS X v10.5 and later.

Declared In
QTCaptureSession.h

removeOutput:

Removes an output from the receiver.

```
- (void)removeOutput:(QTCaptureOutput *)output
```

Parameters

output
The QTCaptureOutput instance to be disconnected from the receiver.

Discussion
This method removes a QTCaptureOutput instance previously added using addOutput:error: and releases it.

Availability
Mac OS X v10.5 and later.

Declared In
QTCaptureSession.h

startRunning

Tells the receiver to start capturing data from its inputs and sending data to its outputs.

```
- (void)startRunning;
```

Discussion
When a QTCaptureSession is running, it continuously reads media from its inputs and sends it to those outputs currently accepting data. When data does not need to be sent to file outputs, previews, and other outputs, the capture session should not be running so that the overhead from capturing does not affect application performance. By default, capture sessions are not running.

Availability
Mac OS X v10.5 and later.

Declared In
QTCaptureSession.h

stopRunning

Tells the receiver to stop capturing data from its inputs and sending data to its outputs.

```
- (void)stopRunning;
```

Discussion
When a QTCaptureSession is running, it continuously reads media from its inputs and sends it to those outputs currently accepting data. When data does not need to be sent to file outputs, previews, and other outputs, the capture session should not be running so that the overhead from capturing does not affect application performance. By default, capture sessions are not running.

Availability
Mac OS X v10.5 and later.

Declared In
`QTCaptureSession.h`

Constants

Constant	Description
`QTCaptureSessionErrorKey`	Used as a notification key in the user info dictionary passed to `QTCaptureSessionRuntimeErrorNotification` to indicate the error responsible for the notification. The value is an NSError.

Notifications

QTCaptureSessionRuntimeErrorNotification

Posted when an error occurs that while a capture session is running prevents input media from being previewed or captured. The notification user info dictionary `QTCaptureSessionErrorKey` entry contains an NSError object that describes the error that prevented the session from running properly. Normally, such errors are caused by an invalid configuration of inputs and outputs.

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

Declared In
`QTCaptureSession.h`

QTCaptureVideoPreviewOutput Class Reference

Inherits from	QTCaptureOutput : NSObject
Conforms to	NSObject (NSObject)
Framework	Library/Frameworks/QTKit.framework
Declared in	QTKit/QTCaptureVideoPreviewOutput.h
Availability	Available in QuickTime 7.2.1 and later.
Related sample code	LiveVideoMixer3

Overview

This class represents an output destination for `QTCaptureSession` that can be used to preview the video being captured. Instances of `QTCaptureVideoPreviewOutput` produce decompressed video frames suitable for preview. Because the output video is intended for preview only, instances may drop frames or reduce output quality in order to improve overall performance of the capture session. Applications that need to process full-quality frames without dropping them should use `QTCaptureDecompressedVideoOutput` instead. Applications can access the decompressed frames from a QuickTime visual context for each output connection, or via the `captureOutput:didOutputVideoFrame:withSampleBuffer:fromConnection:` delegate method. In addition, clients can create subclasses of `QTCaptureVideoPreviewOutput` to add custom capturing behavior. Application Kit clients wishing to preview video do not normally need to use `QTCaptureVideoPreviewOutput` instances directly, as they are created and managed by instances of `QTCaptureView`. Clients should use `QTCaptureVideoPreviewOutput` directly only when they require preview functionality not provided by `QTCaptureView` or when they need to process decompressed frames directly.

Tasks

Previewing Output

- `delegate` (page 96)
  Returns the receiver's delegate.
- `visualContextForConnection:` (page 97)
  Returns the QuickTime visual context used to preview the video for the given connection.
- `outputVideoFrame:withSampleBuffer:fromConnection:` (page 96)
  Called whenever the receiver outputs a new video frame.

- setDelegate: (page 97)
  Sets the receiver's delegate.
- setVisualContext:forConnection: (page 97)
  Sets the QuickTime visual context used to preview the video for the described connection.

Capturing Output

- captureOutput:didOutputVideoFrame:withSampleBuffer:fromConnection: (page 98) *delegate method*
  Called whenever the video preview output outputs a new video frame.

Instance Methods

delegate

Returns the receiver's delegate.

```
- (id)delegate
```

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

Declared In
QTCaptureVideoPreviewOutput.h

outputVideoFrame:withSampleBuffer:fromConnection:

Called whenever the receiver outputs a new video frame.

```
- (void)outputVideoFrame:(CVImageBufferRef)videoFrame
    withSampleBuffer:(QTSampleBuffer *)sampleBuffer
    fromConnection:(QTCaptureConnection *)connection
```

Parameters

videoFrame
  A buffer containing the decompressed frame.

sampleBuffer
  A sample buffer containing additional information about the frame, such as its presentation time.

connection
  The connection from which the video was received.

Discussion
This method should not be invoked directly. Subclasses can override this method to provide custom processing behavior for each frame. The default implementation calls the delegate's captureOutput:didOutputVideoFrame:withSampleBuffer:fromConnection: method. Subclasses should not assume that this method will be called on the main thread. In addition, this method is called periodically, so it must be efficient to prevent capture performance problems.

Availability
Mac OS X v10.5 and later.
Not available to 64-bit applications.

Declared In
QTCaptureVideoPreviewOutput.h

setDelegate:

Sets the receiver's delegate.

```
- (void)setDelegate:(id)delegate
```

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

Declared In
QTCaptureVideoPreviewOutput.h

setVisualContext:forConnection:

Sets the QuickTime visual context used to preview the video for the described connection.

```
- (void)setVisualContext:(QTVisualContextRef)visualContext
    forConnection:(QTCaptureConnection *)connection
```

Parameters

visualContext
A QTVisualContextRef to be used for the preview of the given connection.

connection
The connection to be previewed by the given visual context.

Discussion
If the application has an existing visual context being used to display video, this method can be used to set the visual context for the preview.

Availability
Mac OS X v10.5 and later.
Not available to 64-bit applications.

Declared In
QTCaptureVideoPreviewOutput.h

visualContextForConnection:

Returns the QuickTime visual context used to preview the video for the given connection.

```
- (QTVisualContextRef)visualContextForConnection:(QTCaptureConnection *)connection
```

Parameters

connection

The connection previewed by the returned visual context.

Return Value

A `QTVisualContextRef` that provides access to a video preview for the given connection.

Discussion

The returned visual context can be used to obtain frames that can be used to display a video preview of the capture session. By default this method returns `NULL`, until a visual context is set using `setVisualContext:forConnection:`.

Availability

Mac OS X v10.5 and later.

Not available to 64-bit applications.

Declared In

`QTCaptureVideoPreviewOutput.h`

Delegate Methods

captureOutput:didOutputVideoFrame:withSampleBuffer:fromConnection:

Called whenever the video preview output outputs a new video frame.

```
- (void)captureOutput:(QTCaptureOutput *)captureOutput
    didOutputVideoFrame:(CVImageBufferRef)videoFrame
    withSampleBuffer:(QTSampleBuffer *)sampleBuffer
    fromConnection:(QTCaptureConnection *)connection
```

Parameters

captureOutput

The `QTCaptureVideoPreviewOutput` instance that output the frame.

videoFrame

A `CVImageBufferRef` containing the decompressed frame.

sampleBuffer

A `QTSampleBuffer` object containing additional information about the frame, such as its presentation time.

connection

The connection from which the video was received.

Discussion

Delegates receive this method whenever the output decompresses and outputs a new video frame. Delegates ca use the provided video frame for a custom preview or for further image processing. Delegates should not assume that this method will be called on the main thread. In addition, this method is called periodically, so it must be efficient to prevent capture performance problems.

Availability

Mac OS X v10.5 and later.

Declared In
`QTCaptureDecompressedVideoOutput.h`

QTCaptureView Class Reference

Inherits from	NSView : NSResponder : NSObject
Conforms to	NSAnimatablePropertyContainer (NSView) NSCoding (NSResponder) NSObject (NSObject)
Framework	Library/Frameworks/QTKit.framework
Declared in	QTKit/QTCaptureView.h
Availability	Available in QuickTime 7.2.1 and later.
Related sample code	QT Capture Widget QTRecorder

Overview

This is a subclass of `NSView` that displays a video preview of a capture session. A `QTCaptureView` previews the video being processed by an instance of `QTCaptureSession`. This class creates and maintains its own `QTCaptureVideoPreviewOutput` as necessary to gather preview video from the capture session.

Tasks

Associating a View with a Capture Session

- `availableVideoPreviewConnections` (page 102)
  Returns an array of output video connections that can be previewed.
- `captureSession` (page 103)
  Returns the capture session being previewed by the receiver.
- `setCaptureSession:` (page 104)
  Sets the capture session to be previewed by the receiver.
- `setVideoPreviewConnection:` (page 105)
  Sets the output connection to be previewed by the receiver.
- `videoPreviewConnection` (page 106)
  Returns the output connection being previewed by the receiver.

Controlling View Appearance

- `fillColor` (page 103)
  Returns the fill color drawn in the area of the view not covered by the video preview.
- `preservesAspectRatio` (page 103)
  Returns whether the receiver preserves the aspect ratio of the video preview when drawing it.
- `previewBounds` (page 104)
  Returns the rectangle occupied by the video preview in the view.
- `setFillColor:` (page 105)
  Sets the fill color drawn in the area of the view not covered by the video preview.
- `setPreservesAspectRatio:` (page 105)
  Sets whether the receiver preserves the aspect ratio of the video preview when drawing it.

Getting and Setting a Delegate

- `delegate` (page 103)
  Returns the receiver's delegate.
- `setDelegate:` (page 104)
  Sets the receiver's delegate.

Methods Implemented by the Delegate

- `view:willDisplayImage:` (page 106) *delegate method*
  Delegates of `QTCaptureView` can implement this method to modify the image that is to be drawn into a `QTCaptureView`.

Instance Methods

availableVideoPreviewConnections

Returns an array of output video connections that can be previewed.

```
- (NSArray *)availableVideoPreviewConnections
```

Return Value
An array of `QTCaptureConnection` instances for connections available to be previewed.

Discussion
This method returns an array of connections that can be previewed with the receiver. The returned connections can be used with the `setVideoPreviewConnection:` method to set the connection being previewed by the receiver.

Availability
Mac OS X v10.5 and later.

Declared In
QTCaptureView.h

captureSession

Returns the capture session being previewed by the receiver.

```
- (QTCaptureSession *)captureSession
```

Return Value
A QTCaptureSession instance used for the preview.

Availability
Mac OS X v10.5 and later.

Declared In
QTCaptureView.h

delegate

Returns the receiver's delegate.

```
- (id)delegate
```

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

Declared In
QTCaptureView.h

fillColor

Returns the fill color drawn in the area of the view not covered by the video preview.

```
- (NSColor *)fillColor
```

Return Value
An NSColor of the receiver's fill color.

Availability
Mac OS X v10.5 and later.

Declared In
QTCaptureView.h

preservesAspectRatio

Returns whether the receiver preserves the aspect ratio of the video preview when drawing it.

```
- (BOOL)preservesAspectRatio
```

Return Value
Returns YES if the video preview aspect ratio is preserved; otherwise, NO.

Availability
Mac OS X v10.5 and later.

Declared In
`QTCaptureView.h`

previewBounds

Returns the rectangle occupied by the video preview in the view.

```
- (NSRect)previewBounds
```

Return Value
The rectangle occupied by the video preview in the view.

Discussion
The default implementation of this method returns a video rectangle based on the value returned by `preservesAspectRatio`. Subclasses can override this method to change the rectangle occupied by the video preview.

Availability
Mac OS X v10.5 and later.

Declared In
`QTCaptureView.h`

setCaptureSession:

Sets the capture session to be previewed by the receiver.

```
- (void)setCaptureSession:(QTCaptureSession *)captureSession
```

Parameters
captureSession
A `QTCaptureSession` instance to be used for the preview.

Availability
Mac OS X v10.5 and later.

Declared In
`QTCaptureView.h`

setDelegate:

Sets the receiver's delegate.

```
- (void)setDelegate:(id)delegate
```

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

Declared In
QTCaptureView.h

setFillColor:

Sets the fill color drawn in the area of the view not covered by the video preview.

```
- (void)setFillColor:(NSColor *)fillColor
```

Parameters

fillColor
An NSColor to be used for the receiver's fill color.

Availability
Mac OS X v10.5 and later.

Declared In
QTCaptureView.h

setPreservesAspectRatio:

Sets whether the receiver preserves the aspect ratio of the video preview when drawing it.

```
- (void)setPreservesAspectRatio:(BOOL)preservesAspectRatio
```

Parameters

preservesAspectRatio
If YES, preserves the aspect ratio; otherwise, NO.

Availability
Mac OS X v10.5 and later.

Declared In
QTCaptureView.h

setVideoPreviewConnection:

Sets the output connection to be previewed by the receiver.

```
- (void)setVideoPreviewConnection:(QTCaptureConnection *)connection
```

Parameters

connection
A QTCaptureConnection instance for the connection to be previewed.

Discussion
A QTCaptureView can only preview one video connection at a time. This method sets the output connection to be previewed by the receiver. The given connection must be one of the connections returned by availableVideoPreviewConnections or this method throws an NSInvalidArgumentException.

Availability
Mac OS X v10.5 and later.

Declared In
QTCaptureView.h

videoPreviewConnection

Returns the output connection being previewed by the receiver.

```
- (QTCaptureConnection *)videoPreviewConnection
```

Return Value
A QTCaptureConnection instance for the previewed connection.

Discussion
A QTCaptureView can preview only one video connection at a time. This method returns the output connection currently being previewed by the receiver.

Availability
Mac OS X v10.5 and later.

Declared In
QTCaptureView.h

Delegate Methods

view:willDisplayImage:

Delegates of QTCaptureView can implement this method to modify the image that is to be drawn into a QTCaptureView.

```
- (CIImage *)view:(QTCaptureView *)view willDisplayImage
    :(CIImage *)image
```

Parameters

view
A QTCaptureView object that identifies the view which is about to draw.

image
A CIImage object that represents the frame that will otherwise be drawn to the QTCaptureView.

Return Value
Delegates should return a CIImage object to be drawn by the capture view, or NIL if the capture view should draw the original image.

Discussion
The image parameter is a CIImage representing the captured frame that is about to be drawn into a QTCaptureView. The delegate can return another image that modifies the source image (by applying a CIFilter, for example). The returned image will then be drawn into the capture view instead of the source image. The delegate can also return nil or the original image to leave the drawn image unmodified.

Availability
Mac OS X v10.5 and later.

Declared In
`QTCaptureView.h`

QTCompressionOptions Reference

Inherits from	NSObject
Conforms to	NSObject (NSObject)
Framework	Library/Frameworks/QTKit.framework
Declared in	QTKit/QTCompressionOptions.h
Availability	Available in QuickTime 7.2.1 and later.

Overview

This class represents a set of compression options for a particular type of media. `QTCompressionOptions` objects are used to describe compression options for different kinds of media. Compression options are created from presets keyed by a named identifier. Preset identifiers are described in the Constants section that describes the Compression Options Identifiers.

Tasks

Creating and Configuring Compression Options

`+ compressionOptionsIdentifiersForMediaType:` (page 110)
An array of identifiers that can be used to create compression options on the user's computer.

`+ compressionOptionsWithIdentifier:` (page 110)
A `QTCompressionOptions` instance configured with the options for the given identifier.

Receiving Compression Options

`- mediaType` (page 111)
The media type on which the receiver's compression options should be used.

`- localizedDisplayName` (page 111)
A short localized name describing the receiver's compression options.

`- localizedCompressionOptionsSummary` (page 111)
A localized summary of the receiver's compression options.

Class Methods

compressionOptionsIdentifiersForMediaType:

An array of identifiers that can be used to create compression options on the user's computer.

```
+ (NSArray *)compressionOptionsIdentifiersForMediaType:(NSString *)mediaType
```

Return Value
An array of strings that can be used to create compression options with `compressionOptionsWithIdentifier:`.

Availability
Mac OS X v10.5 and later.

Declared In
`QTCompressionOptions.h`

compressionOptionsWithIdentifier:

A `QTCompressionOptions` instance configured with the options for the given identifier.

```
+ (id)compressionOptionsWithIdentifier:(NSString *)identifier
```

Return Value
A compression options object with the appropriate compression options.

Availability
Mac OS X v10.5 and later.

Declared In
`QTCompressionOptions.h`

Instance Methods

isEqualToCompressionOptions:

Returns whether the receiver contains options identical to those in the given compression options object.

```
- (BOOL)isEqualToCompressionOptions:(QTCompressionOptions *)compressionOptions
```

Availability
Mac OS X v10.5 and later.

Declared In
`QTCompressionOptions.h`

localizedCompressionOptionsSummary

A localized summary of the receiver's compression options.

```
- (NSString *)localizedCompressionOptionsSummary
```

Return Value
A localized string summarizing the receiver's compression options.

Availability
Mac OS X v10.5 and later.

Declared In
`QTCompressionOptions.h`

localizedDisplayName

A short localized name describing the receiver's compression options.

```
- (NSString *)localizedDisplayName
```

Return Value
A localized string appropriate for display in the user interface (in a list of compression options, for example).

Availability
Mac OS X v10.5 and later.

Declared In
`QTCompressionOptions.h`

mediaType

The media type on which the receiver's compression options should be used.

```
- (NSString *)mediaType
```

Return Value
A QuickTime media type, such as `QTMediaTypeVideo` or `QTMediaTypeSound`.

Availability
Mac OS X v10.5 and later.

Declared In
`QTCompressionOptions.h`

Constants

The following are compression options identifiers. These identifiers can be passed to the `compressionOptionsWithIdentifier:` class method to get an instance configured with the compression options for that identifier.

QTCompressionOptionsLosslessAppleIntermediateVideo

Compresses video using the Apple Intermediate codec at lossless quality.

```
@"QTCompressionOptionsLosslessAppleIntermediateVideo"
```

Constants

`@"QTCompressionOptionsLosslessAppleIntermediateVideo"`
This is appropriate for an intermediate format for media that requires further processing.

Declared In

`QTCompressionOptions.h`

QTCompressionOptionsLosslessAnimationVideo

Compresses video using the Animation codec at highest quality and color depth.

```
@"QTCompressionOptionsLosslessAnimationVideo"
```

Constants

`@"QTCompressionOptionsLosslessAnimationVideo"`
This is appropriate for an intermediate format for media that requires further processing.

Declared In

`QTCompressionOptions.h`

QTCompressionOptions120SizeH264Video

Compresses video using the H.264 codec using medium bit-rate settings with dimensions no larger than 160x120.

```
@"QTCompressionOptions120SizeH264Video"
```

Constants

`@"QTCompressionOptions120SizeH264Video"`
This is appropriate for delivery to low-bandwidth and low-capacity destinations.

Declared In

`QTCompressionOptions.h`

QTCompressionOptions240SizeH264Video

Compresses video using the H.264 codec using medium bit-rate settings with dimensions no larger than 320x240.

```
@"QTCompressionOptions240SizeH264Video"
```

Constants

`@"QTCompressionOptions240SizeH264Video"`
This is appropriate for delivery to medium-bandwidth and medium-capacity destinations.

Declared In

`QTCompressionOptions.h`

QTCompressionOptionsSD480SizeH264Video

Compresses video using the H.264 codec using medium bit-rate settings with dimensions no larger than 720x480.

```
@"QTCompressionOptionsSD480SizeH264Video"
```

Constants

`@"QTCompressionOptionsSD480SizeH264Video"`
This is appropriate for delivery to medium and high-bandwidth and medium- and high-capacity destinations.

Declared In

`QTCompressionOptions.h`

QTCompressionOptions120SizeMPEG4Video

Compresses video using the MPEG-4 codec using medium bit-rate settings with dimensions no larger than 160x120.

```
@"QTCompressionOptions120SizeMPEG4Video"
```

Constants

`@"QTCompressionOptions120SizeMPEG4Video"`
This is appropriate for delivery to low-bandwidth and low-capacity destinations.

Declared In

`QTCompressionOptions.h`

QTCompressionOptions240SizeMPEG4Video

Compresses video using the MPEG-4 codec using medium bit-rate settings with dimensions no larger than 320x240.

```
@"QTCompressionOptions240SizeMPEG4Video"
```

Constants

`@"QTCompressionOptions240SizeMPEG4Video"`
This is appropriate for delivery to medium-bandwidth and medium-capacity destinations.

Declared In

`QTCompressionOptions.h`

QTCompressionOptionsSD480SizeMPEG4Video

Compresses video using the MPEG-4 codec using medium bit-rate settings with dimensions no larger than 720x480.

```
@"QTCompressionOptionsSD480SizeMPEG4Video"
```

Constants

`@"QTCompressionOptionsSD480SizeMPEG4Video"`
This is appropriate for delivery to medium and high-bandwidth and medium- and high-capacity destinations.

Declared In

`QTCompressionOptions.h`

QTCompressionOptionsLosslessALACAudio

Compresses audio using the Apple Lossless codec.

```
@"QTCompressionOptionsLosslessALACAudio"
```

Constants

`@"QTCompressionOptionsLosslessALACAudio"`
This is appropriate for an intermediate format for media that requires further processing.

Declared In

`QTCompressionOptions.h`

QTCompressionOptionsHighQualityAACAudio

Compresses audio using the AAC codec at 64 kbps per channel.

```
@"QTCompressionOptionsHighQualityAACAudio"
```

Constants

`@"QTCompressionOptionsHighQualityAACAudio"`
This is appropriate for delivery of high-quality music and other audio.

Declared In

`QTCompressionOptions.h`

QTCompressionOptionsVoiceQualityAACAudio

Compresses audio using the AAC codec at 32 kbps per channel.

```
@"QTCompressionOptionsVoiceQualityAACAudio"
```

Constants

`@"QTCompressionOptionsVoiceQualityAACAudio"`
This is appropriate for delivery of voice recordings.

Declared In

`QTCompressionOptions.h`

QTDataReference Class Reference

Inherits from	NSObject
Conforms to	NSCoding NSObject (NSObject)
Framework	QTKit.framework
Declared in	QTKit/QTDataReference.h
Availability	Available in Mac OS X v10.4 and later.

Overview

A QTDataReference object is a representation of a QuickTime data reference which specifies the location of a QuickTime movie or some media data. You can create QTDataReference objects that refer to data stored in files accessed using file names or URLs, or in memory accessed using handles, pointers, or NSData objects.

Tasks

Creating a QTDataReference

`+ dataReferenceWithDataRef:type:` (page 117)

`+ dataReferenceWithDataRefData:type:` (page 117)

`+ dataReferenceWithReferenceToFile:` (page 118)

`+ dataReferenceWithReferenceToURL:` (page 118)

`+ dataReferenceWithReferenceToData:` (page 117)

`+ dataReferenceWithReferenceToData:name:MIMEType:` (page 117)

Initializing a QTDataReference

- `initWithDataRef:type:` (page 119)
- `initWithDataRefData:type:` (page 119)
- `initWithReferenceToFile:` (page 120)
- `initWithReferenceToURL:` (page 121)
- `initWithReferenceToData:` (page 120)
- `initWithReferenceToData:name:MIMEType:` (page 120)

Getting and Setting Data Reference Information

- `dataRef` (page 118)
- `dataRefData` (page 119)
- `dataRefType` (page 119)
- `referenceFile` (page 122)
- `referenceURL` (page 122)
- `referenceData` (page 121)
- `name` (page 121)
- `MIMEType` (page 121)
- `setDataRef:` (page 122)
- `setDataRefType:` (page 122)

Class Methods

dataReferenceWithDataRef:type:

```
+ (id)dataReferenceWithDataRef:(Handle)dataRef type:(NSString *)type
```

Discussion
Creates a QTDataReference object of type *type* initialized with data from *dataRef*. You can use this call to convert an existing QuickTime data reference (stored as a handle) into a QTDataReference.

Availability
Available in Mac OS X v10.3 and later.

Declared In
QTDataReference.h

dataReferenceWithDataRefData:type:

```
+ (id)dataReferenceWithDataRefData:(NSData *)dataRefData type:(NSString *)type
```

Discussion
Creates a QTDataReference object of type *type* initialized with data from *dataRefData*.

Availability
Available in Mac OS X v10.3 and later.

Declared In
QTDataReference.h

dataReferenceWithReferenceToData:

```
+ (id)dataReferenceWithReferenceToData:(NSData *)data
```

Discussion
Creates a QTDataReference object for the data block *data*.

Availability
Available in Mac OS X v10.3 and later.

Declared In
QTDataReference.h

dataReferenceWithReferenceToData:name:MIMEType:

```
+ (id)dataReferenceWithReferenceToData:(NSData *)data name:(NSString *)name
    MIMEType:(NSString *)MIMEType
```

Discussion
Creates a QTDataReference object for the data block *data*; this data reference has two data reference extensions, a filenaming extension and a MIME type extension.

Availability
Available in Mac OS X v10.3 and later.

Declared In
QTDataReference.h

dataReferenceWithReferenceToFile:

```
+ (id)dataReferenceWithReferenceToFile:(NSString *)fileName
```

Discussion
Creates a QTDataReference object for the file *fileName*. The *fileName* is assumed to be a full path name for a file.

Availability
Available in Mac OS X v10.3 and later.

Declared In
QTDataReference.h

dataReferenceWithReferenceToURL:

```
+ (id)dataReferenceWithReferenceToURL:(NSURL *)url
```

Discussion
Creates a QTDataReference object for the URL *url*.

Availability
Available in Mac OS X v10.3 and later.

Declared In
QTDataReference.h

Instance Methods

dataRef

```
- (Handle)dataRef
```

Discussion
Returns the QuickTime data reference associated with a QTDataReference object.

Availability
Available in Mac OS X v10.3 and later.

Declared In
QTDataReference.h

dataRefData

```
- (NSData *)dataRefData
```

Discussion
Returns the QuickTime data reference data associated with a QTDataReference object, stored in an NSData object.

Availability
Available in Mac OS X v10.3 and later.

Declared In
QTDataReference.h

dataRefType

```
- (NSString *)dataRefType
```

Discussion
Returns the type of the data reference associated with a QTDataReference object.

Availability
Available in Mac OS X v10.3 and later.

Declared In
QTDataReference.h

initWithDataRef:type:

```
- (id)initWithDataRef:(Handle)dataRef type:(NSString *)type
```

Discussion
Initializes a newly created QTDataReference object with data from *dataRef*; the QTDataReference is of type *dataRefType*. You can use this call to convert an existing QuickTime data reference (stored as a handle) into a QTDataReference.

Availability
Available in Mac OS X v10.3 and later.

Declared In
QTDataReference.h

initWithDataRefData:type:

```
- (id)initWithDataRefData:(NSData *)dataRefData type:(NSString *)type
```

Discussion
Initializes a newly created QTDataReference object with data from *dataRefData*; the QTDataReference is of type *dataRefType*.

Availability
Available in Mac OS X v10.3 and later.

Declared In
QTDataReference.h

initWithReferenceToData:

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

Discussion
Initializes a newly created QTDataReference object for the data block *data*.

Availability
Available in Mac OS X v10.3 and later.

Declared In
QTDataReference.h

initWithReferenceToData:name:MIMEType:

```
- (id)initWithReferenceToData:(NSData *)data name:(NSString *)name MIMEType:(NSString
    *)MIMEType
```

Discussion
Initializes a newly created QTDataReference object for the data block *data*; this data reference has two data reference extensions, a filenaming extension and a MIME type extension.

Availability
Available in Mac OS X v10.3 and later.

Declared In
QTDataReference.h

initWithReferenceToFile:

```
- (id)initWithReferenceToFile:(NSString *)fileName
```

Discussion
Initializes a newly created QTDataReference object for the file *fileName*. The *fileName* is assumed to be a full path name for a file.

Availability
Available in Mac OS X v10.3 and later.

Declared In
QTDataReference.h

initWithReferenceToURL:

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

Discussion
Initializes a newly created QTDataReference object for the URL *url*.

Availability
Available in Mac OS X v10.3 and later.

Declared In
QTDataReference.h

MIMEType

```
- (NSString *)MIMEType
```

Discussion
Returns the type in a MIME type extension associated with a QTDataReference object.

Availability
Available in Mac OS X v10.3 and later.

Declared In
QTDataReference.h

name

```
- (NSString *)name
```

Discussion
Returns the name in a filenaming extension associated with a QTDataReference object

Availability
Available in Mac OS X v10.3 and later.

Declared In
QTDataReference.h

referenceData

```
- (NSData *)referenceData
```

Discussion
Returns the reference data of a QTDataReference object, that is, the NSData object passed to `initWithReferenceToData` or `initWithReferenceToData:name:MIMEType`. For some QTDataReference objects, this may be `NIL`.

Availability
Available in Mac OS X v10.3 and later.

Declared In
QTDataReference.h

referenceFile

```
- (NSString *)referenceFile
```

Discussion
Returns the file name of the data reference associated with a QTDataReference object. For some QTDataReference objects, this name may be NIL.

Availability
Available in Mac OS X v10.3 and later.

Declared In
QTDataReference.h

referenceURL

```
- (NSURL *)referenceURL
```

Discussion
Returns the URL of the data reference associated with a QTDataReference object. For some QTDataReference objects, this URL may be NIL.

Availability
Available in Mac OS X v10.3 and later.

Declared In
QTDataReference.h

setDataRef:

```
- (void)setDataRef:(Handle)dataRef
```

Discussion
Sets the data reference data of a QTDataReference object to *dataRef*. The previous data reference data is disposed of.

Availability
Available in Mac OS X v10.3 and later.

Declared In
QTDataReference.h

setDataRefType:

```
- (void)setDataRefType:(NSString *)type
```

Discussion
Sets the data reference type of a QTDataReference object to *type*.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTDataReference.h`

Constants

The following constants are Cocoa identifiers for the basic data reference types. One of these types would be returned, for instance, by this method: `- (NString *) dataRefType`.

Constant	Description
`QTDataReferenceTypeFile`	The file type for a QTDataReference object.
`QTDataReferenceTypeHandle`	The handle type for a QTDataReference object.
`QTDataReferenceTypePointer`	The pointer type for a QTDataReference object.
`QTDataReferenceTypeResource`	The resource type for a QTDataReference object.
`QTDataReferenceTypeURL`	The URL type for a QTDataReference object.

QTFormatDescription

Inherits from	NSObject
Conforms to	NSObject (NSObject)
Framework	Library/Frameworks/QTKit.framework
Declared in	QTKit/QTFormatDescription.h
Availability	Available in QuickTime 7.2.1 and later.

Overview

`QTFormatDescription` objects are used to describe the media format of media samples and of media sources, such as devices and capture connections. Format descriptions include basic information about the media, such as media type and format type (or codec type), as well as extended information specific to each media type. The extended information can be accessed via the object's `attributeForKey:` and `formatDescriptionAttributes` methods, using the keys described in the Constants section. In addition to these explicit methods, applications can use key-value coding to get extended attributes. For an object that supports a given attribute, `valueForKey:` will be functionally identical to `attributeForKey:`. Applications wishing to observe changes for a given attribute can add a key-value observer where the key path is the attribute key.

Tasks

Formatting Different Types of Media

- `attributeForKey:` (page 126)
  Returns the current value of the format description attribute for the given key.
- `formatDescriptionAttributes` (page 126)
  Returns a dictionary of all attributes set for the receiver.
- `formatType` (page 126)
  Returns the format type of the described media.
- `isEqualToFormatDescription:` (page 127)
  Returns whether the receiver describes the same format as the given format description.
- `localizedFormatSummary` (page 127)
  Returns a localized summary of the media format.

- `mediaType` (page 127)
  Returns the media type of the described media.
- `quickTimeSampleDescription` (page 128)
  Returns the media's QuickTime SampleDescription.

Instance Methods

attributeForKey:

Returns the current value of the format description attribute for the given key.

```
- (id)attributeForKey:(NSString *)key
```

Discussion
Use this method to get attributes of a format description. The keys that can be used with this method are described in the Constants section. Applications using key-value coding can also get an attribute for a given key by passing that key to the NSObject `valueForKey:` method.

Availability
Mac OS X v10.5 and later.

Declared In
`QTFormatDescription.h`

formatDescriptionAttributes

Returns a dictionary of all attributes set for the receiver.

```
- (NSDictionary *)formatDescriptionAttributes
```

Discussion
Applications can use this method to determine what attributes a specific format description supports.

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

Declared In
`QTFormatDescription.h`

formatType

Returns the format type of the described media.

```
- (UInt32)formatType
```

Discussion
This method returns the specific format, or codec, used to represent the media. Video format types are defined in `QuickTime/ImageCompression.h` and audio format types are defined in `CoreAudio/CoreAudioTypes.h`.

Availability
Mac OS X v10.5 and later.

Declared In
`QTFormatDescription.h`

isEqualToFormatDescription:

Returns whether the receiver describes the same format as the given format description.

```
- (BOOL)isEqualToFormatDescription:(QTFormatDescription *)formatDescription
```

Availability
Mac OS X v10.5 and later.

Declared In
`QTFormatDescription.h`

localizedFormatSummary

Returns a localized summary of the media format.

```
- (NSString *)localizedFormatSummary
```

Return Value
A localized string summarizing the media format.

Availability
Mac OS X v10.5 and later.

Related Sample Code
QTRecorder

Declared In
`QTFormatDescription.h`

mediaType

Returns the media type of the described media.

```
- (NSString *)mediaType
```

Return Value
A QuickTime media type, such as `QTMediaTypeVideo`, `QTMediaTypeSound`, or `QTMediaTypeMuxed`.

Availability
Mac OS X v10.5 and later.

Declared In
`QTFormatDescription.h`

quickTimeSampleDescription

Returns the media's QuickTime SampleDescription.

```
- (NSData *)quickTimeSampleDescription
```

Return Value
An `NSData` containing the `SampleDescription` for the media.

Discussion
This method returns a QuickTime `SampleDescription` structure, allowing applications to get detailed information on the media format. The `SampleDescription` is returned in the native endian byte order for the system.

Availability
Mac OS X v10.5 and later.
Not available to 64-bit applications.

Declared In
`QTFormatDescription.h`

Constants

QTFormatDescriptionAudioChannelLayoutAttribute

Returns an `NSData` interpreted as a Core Audio `AudioChannelLayout` for audio media.

```
QTFormatDescriptionAudioChannelLayoutAttribute
@"audioChannelLayout"
```

Constants
`QTFormatDescriptionAudioChannelLayoutAttribute`
This string value can be used in key paths for key-value coding, key-value observing, and bindings.

Declared In
`QTFormatDescription.h`

QTFormatDescriptionAudioMagicCookieAttribute

Returns an NSData interpreted as a Core Audio magic cookie for audio media.

```
QTFormatDescriptionAudioMagicCookieAttribute
@"audioMagicCookie"
```

Constants
`QTFormatDescriptionAudioMagicCookieAttribute`
This string value can be used in key paths for key-value coding, key-value observing, and bindings.

Declared In
`QTFormatDescription.h`

QTFormatDescriptionAudioStreamBasicDescriptionAttribute

Returns an `NSValue` interpreted as a Core Audio `AudioStreamBasicDescription` for audio media.

```
QTFormatDescriptionAudioStreamBasicDescriptionAttribute
@"audioStreamBasicDescription"
```

Constants

`QTFormatDescriptionAudioStreamBasicDescriptionAttribute`
This string value can be used in key paths for key-value coding, key-value observing, and bindings.

Declared In

`QTFormatDescription.h`

QTFormatDescriptionVideoCleanApertureDisplaySizeAttribute

Returns an `NSValue` interpreted as an `NSSize` that indicates the size of video media displayed through its clean aperture and scaled by its pixel aspect ratio.

```
QTFormatDescriptionVideoCleanApertureDisplaySizeAttribute
@"videoCleanApertureDisplaySize"
```

Constants

`QTFormatDescriptionVideoCleanApertureDisplaySizeAttribute`
This string value can be used in key paths for key-value coding, key-value observing, and bindings.

Declared In

`QTFormatDescription.h`

QTFormatDescriptionVideoEncodedPixelsSizeAttribute

Returns an `NSValue` interpreted as an `NSSize` that indicates the encoded size of video media.

```
QTFormatDescriptionVideoEncodedPixelsSizeAttribute
@"videoEncodedPixelsSize"
```

Constants

`QTFormatDescriptionVideoEncodedPixelsSizeAttribute`
This string value can be used in key paths for key-value coding, key-value observing, and bindings.

Declared In

`QTFormatDescription.h`

QTFormatDescriptionVideoProductionApertureDisplaySizeAttribute

Returns an `NSValue` interpreted as an `NSSize` that indicates the size of video media scaled by its pixel aspect ratio but not displayed through its clean aperture.

```
QTFormatDescriptionVideoProductionApertureDisplaySizeAttribute
@"videoProductionApertureDisplaySize"
```

Constants

`QTFormatDescriptionVideoProductionApertureDisplaySizeAttribute`
This string value can be used in key paths for key-value coding, key-value observing, and bindings.

Declared In
`QTFormatDescription.h`

QTMedia Class Reference

Inherits from	NSObject
Conforms to	NSObject (NSObject)
Framework	QTKit.framework
Declared in	QTKit/QTMedia.h
Availability	Available in Mac OS X v10.4 and later.
Related sample code	QTKitTimeCode QTMetadataEditor

Overview

The QTMedia class represents a QuickTime media (of type `Media`). QTMedia objects are associated with QTTrack objects and support methods for getting and setting the media properties. If necessary, you can retrieve the media identifier associated with a QTMedia object by calling its `quickTimeMedia:` method. Note that a track has a single media.

Tasks

Creating a QTMedia

`+ mediaWithQuickTimeMedia:error:` (page 132)

Initializing a QTMedia

`- initWithQuickTimeMedia:error:` (page 133)

Getting Media Properties

`- track` (page 135)

Setting Media Properties

Getting QTMedia Primitives

Class Methods

mediaWithQuickTimeMedia:error:

```
+ (id)mediaWithQuickTimeMedia:(Media)media error:(NSError **)errorPtr
```

Discussion
Creates a QTMedia object with data from the QuickTime media *media*. If a QTMedia object cannot be created, an NSError object is returned in the location pointed to by *errorPtr*. Pass NIL if you do not want an NSError object returned.

Availability
Available in Mac OS X v10.3 and later.
Not available to 64-bit applications.

Declared In
QTMedia.h

Instance Methods

attributeForKey:

```
- (id)attributeForKey:(NSString *)attributeKey
```

Discussion
Returns the current value of the media attribute *attributeKey*. A list of supported media attributes and their acceptable values can be found in "Constants" (page 135).

Availability
Available in Mac OS X v10.3 and later.

Related Sample Code
QTMetadataEditor

Declared In
QTMedia.h

hasCharacteristic:

```
- (BOOL)hasCharacteristic:(NSString *)characteristic
```

Discussion
Returns YES if a QTMedia object has the specified characteristic. See the list of constants given in "Constants" (page 135) for the characteristics you can query.

Availability
Available in Mac OS X v10.3 and later.

Declared In
QTMedia.h

initWithQuickTimeMedia:error:

```
- (id)initWithQuickTimeMedia:(Media)media error:(NSError **)errorPtr
```

Discussion
Initializes a newly created QTMedia object with data from the QuickTime media *media*. If a QTMedia object cannot be created, an NSError object is returned in the location pointed to by *errorPtr*. Pass NIL if you do not want an NSError object returned.

Availability
Available in Mac OS X v10.3 and later.
Not available to 64-bit applications.

Declared In
QTMedia.h

mediaAttributes

```
- (NSDictionary *)mediaAttributes
```

Discussion
Returns a dictionary containing the current values of all defined media attributes. A list of supported media attributes and their acceptable values can be found in "Constants" (page 135).

Availability
Available in Mac OS X v10.3 and later.

Declared In
QTMedia.h

quickTimeMedia

```
- (Media)quickTimeMedia
```

Discussion
Returns the QuickTime media associated with a QTMedia object.

Availability
Available in Mac OS X v10.3 and later.
Not available to 64-bit applications.

Declared In
QTMedia.h

setAttribute:forKey:

```
- (void)setAttribute:(id)value forKey:(NSString *)attributeKey
```

Discussion
Set the media attribute *attributeKey* to the value specified by the *value* parameter. A list of supported media attributes and their acceptable values can be found in "Constants" (page 135).

Availability
Available in Mac OS X v10.3 and later.

Declared In
QTMedia.h

setMediaAttributes:

```
- (void)setMediaAttributes:(NSDictionary *)attributes
```

Discussion
Set the media attributes using the key-value pairs specified in the dictionary *attributes*. A list of supported media attributes and their acceptable values can be found in "Constants" (page 135).

Availability
Available in Mac OS X v10.3 and later.

Declared In
QTMedia.h

track

```
- (QTTrack *)track
```

Discussion
Returns the QTTrack that contains a QTMedia object.

Availability
Available in Mac OS X v10.3 and later.

Declared In
QTMedia.h

Constants

The following constants specify the media attributes that you can get and set using the `mediaAttributes` and `setMediaAttributes` methods. To get or set a single attribute, use `attributeForKey:` or `setAttribute:`.

Note: The `QTMediaTypeAttribute` attribute indicates the type of media data contained in a QTMedia object. These constants are used to indicate media types, as shown in the portion of the table beginning with the `QTMediaTypeVideo` constant.

Constant	Description
`QTMediaCreationTimeAttribute`	The creation time of a QTMedia object; the value for this key is of type `NSDate`.
`QTMediaDurationAttribute`	The duration of a QTMedia object; the value for this key is of type `NSValue`, interpreted as a QTTime.
`QTMediaModificationTimeAttribute`	The modification time of a QTMedia object; the value for this key is of type `NSDate`.
`QTMediaSampleCountAttribute`	The media sample count; the value for this key is of type `NSNumber`, interpreted as a `long`.
`QTMediaQualityAttribute`	The media quality; the value for this key is of type `NSNumber`, interpreted as a `short`.
`QTMediaTimeScaleAttribute`	The media time scale; the value for this key is of type `NSNumber`, interpreted as a `long`.
`QTMediaTypeAttribute`	The media type; the value for this key is of type `NSString`; see below for the values this attribute can return.
`QTMediaTypeVideo`	Video media.
`QTMediaTypeSound`	Sound media.
`QTMediaTypeText`	Text media.

Constant	Description
QTMediaTypeBase	Base media.
QTMediaTypeMPEG	MPEG media
QTMediaTypeMusic	Music media
QTMediaTypeTimeCode	Timecode media.
QTMediaTypeSprite	Sprite media.
QTMediaTypeFlash	Flash media.
QTMediaTypeMovie	Movie media.
QTMediaTypeTween	Tween media.
QTMediaType3D	3D media.
QTMediaTypeSkin	Skin media
QTMediaTypeQTVR	QuickTime VR media.
QTMediaTypeHint	Hint media.
QTMediaTypeStream	Stream media.

Note: The constants beginning with `QTMediaCharacteristic` indicate the characteristics of a media that you can query using the `hasCharacteristic` method.

Constant	Description
QTMediaCharacteristicVisual	The media has video data. This is a `BOOL`.
QTMediaCharacteristicAudio	The media has audio data. This is a `BOOL`.
QTMediaCharacteristicCanSendVideo	The media can send visual data to another track. This is a `BOOL`.
QTMediaCharacteristicProvidesActions	The media has actions. This is a `BOOL`.
QTMediaCharacteristicNonLinear	The media is non-linear. This is a `BOOL`.
QTMediaCharacteristicCanStep	The media can step. This is a `BOOL`.
QTMediaCharacteristicHasNoDuration	The media has no duration. This is a `BOOL`.
QTMediaCharacteristicHasSkinData	The media has skin data. This is a `BOOL`.
QTMediaCharacteristicProvidesKeyFocus	Key events can be focused at the media. This is a `BOOL`.
QTMediaCharacteristicHasVideoFrameRate	The media has a video frame rate. This is a `BOOL`.

QTMovie Class Reference

Inherits from	NSObject
Conforms to	NSCoding NSCopying NSObject (NSObject)
Framework	/System/Library/Frameworks/QTKit.framework
Declared in	QTKit/QTMovie.h
Availability	Available in Mac OS X v10.4 and later.
Related sample code	QTAudioExtractionPanel QTKitCreateMovie QTKitPlayer QTKitTimeCode QTMetadataEditor

Overview

The `QTMovie` class represents both a QuickTime movie and a movie controller. A movie is a collection of playable and editable media content. It describes the sources and types of the media in that collection and their spatial and temporal organization. These collections may be used for presentation (such as playback on the screen) or for the organization of media for processing (such as composition and transcoding to a different compression type). The collection may be as simple as a single file that plays at its natural size for its intrinsic duration, or it may be very complex (with multiple sources of content, rich composition rules, interactivity, and a variety of contingencies).

Just as a QuickTime movie contains a set of tracks, each of which defines the type, the segments, and the ordering of the media data it presents, a QTMovie object is associated with instances of the QTTrack class. In turn, a QTTrack object is associated with a single QTMedia object.

A QTMovie object can be initialized from a file, from a resource specified by a URL, from a block of memory, from a pasteboard, or from an existing QuickTime movie.

Once a QTMovie object has been initialized, it will typically be used in combination with a QTMovieView for playback.

An exception, `QTMovieUneditableException`, is raised whenever the client attempts to directly or indirectly edit a QTMovie object that is not currently set as editable (for instance, by calling `appendSelectionFromMovie:` on an uneditable movie).

Tasks

Determining If a Movie Can Be Initialized

+ `canInitWithFile:` (page 144)
  Returns `YES` if the contents of the specified file can be used to initialize a QTMovie object.
+ `canInitWithURL:` (page 145)
  Returns `YES` if the contents of the specified URL can be used to initialize a QTMovie object.
+ `canInitWithPasteboard:` (page 145)
  Returns `YES` if the contents of the specified pasteboard can be used to initialize a QTMovie object.
+ `canInitWithDataReference:` (page 144)
  Returns `YES` if the specified data reference can be used to initialize a QTMovie object.
- `initWithPasteboard:error:` (page 166)
  Initializes a QTMovie object with the contents of the pasteboard specified by *`pasteboard`*.

Getting a List of Supported File Types

+ `movieFileTypes:` (page 147)
  Returns an array of file types that can be opened as QuickTime movies.
+ `movieTypesWithOptions:` (page 148)
  Returns an array of UTIs that QuickTime can open.
+ `movieUnfilteredFileTypes` (page 148)
  Returns an array of file types that can be used to initialize a QTMovie object.
+ `movieUnfilteredPasteboardTypes` (page 149)
  Returns an array of pasteboard types that can be used to initialize a QTMovie object.

Creating a Movie

+ `movie` (page 146)
  Creates an empty QTMovie object.
+ `movieNamed:error:` (page 148)
  Creates a QTMovie object initialized with the data from the QuickTime movie of the specified name in the application's bundle.
+ `movieWithData:error:` (page 151)
  Creates a QTMovie object initialized with the data specified by *`data`*.
+ `movieWithURL:error:` (page 153)
  Creates a QTMovie object initialized with the data in the URL specified by *`url`*.
+ `movieWithPasteboard:error:` (page 152)
  Creates a QTMovie object initialized with the contents of the pasteboard specified by *`pasteboard`*.
+ `movieWithFile:error:` (page 151)
  Creates a QTMovie object initialized with the data in the file specified by the name *`fileName`*.

+ `movieWithDataReference:error:` (page 151)
  Creates a QTMovie object intitalized with the data specified by the data reference *dataReference*.
+ `movieWithQuickTimeMovie:disposeWhenDone:error:` (page 152)
  Creates a QTMovie object initialized with the data from an existing QuickTime movie *movie*.
+ `movieWithAttributes:error:` (page 149)
  Creates a QTMovie object initialized with the attributes specified in *attributes*.

Controlling Movie Playback

- `autoplay` (page 155)
  Sets a movie to start playing when a sufficient amount of media data is available.
- `play` (page 170)
  Plays the movie.
- `stop` (page 177)
  Stops the movie playing.
- `gotoBeginning` (page 160)
  Repositions the play position to the beginning of the movie.
- `gotoEnd` (page 161)
  Repositions the play position to the end of the movie.
- `gotoNextSelectionPoint` (page 161)
  Repositions the movie to the next selection point.
- `gotoPreviousSelectionPoint` (page 161)
  Repositions the movie to the previous selection point.
- `gotoPosterFrame` (page 161)
  Repositions the play position to the movie's poster time.
- `setCurrentTime:` (page 174)
  Sets the movie's current time setting to `time`.
- `stepForward` (page 177)
  Sets the movie forward a single frame.
- `stepBackward` (page 177)
  Sets the movie backward a single frame.

Managing Threaded Operations of Movie Objects

+ `enterQTKitOnThread` (page 145)
  Performs any QuickTime-specific initialization for the current (non-main) thread; must be paired with a call to `exitQTKitOnThread`.
+ `enterQTKitOnThreadDisablingThreadSafetyProtection` (page 146)
  Performs any QuickTime-specific initialization for the current (non-main) thread, allowing non-threadsafe components; must be paired with a call to `exitQTKitOnThread`.
+ `exitQTKitOnThread` (page 146)
  Performs any QuickTime-specific shut-down for the current (non-main) thread; must be paired with a call to `enterQTKitOnThread` or `enterQTKitOnThreadDisablingThreadSafetyProtection`.

- `attachToCurrentThread` (page 155)
  Attaches the receiver to the current thread; returns YES if successful, NO otherwise.
- `detachFromCurrentThread` (page 158)
  Detaches the receiver from the current thread; returns YES if successful, NO otherwise.

Initializing a QTMovie

- `initWithFile:error:` (page 165)
  Initializes a QTMovie object with the data in the file specified by the name *`fileName`*.
- `initWithURL:error:` (page 167)
  Initializes a QTMovie object with the data in the URL specified by *`url`*.
- `initWithData:error:` (page 165)
  Initializes a QTMovie object with the data specified by *`data`*.
- `initWithDataReference:error:` (page 165)
  Initializes a QTMovie object with the data reference setting specified by *`dataReference`*.
- `initWithMovie:timeRange:error:` (page 166)
  Initializes a QTMovie object with some or all of the data from an existing QTMovie object *`movie`*.
- `initWithQuickTimeMovie:disposeWhenDone:error:` (page 167)
  Initializes a QTMovie object with the data from an existing QuickTime movie *`movie`*.
- `initWithAttributes:error:` (page 163)
  Initializes a QTMovie object with the attributes specified in *`attributes`*.

Getting Information About a Movie and Its Chapters

- `hasChapters` (page 162)
  Returns YES if the receiver has chapters, NO otherwise.
- `chapterCount` (page 156)
  Returns the number of chapters in the receiver, or 0 if there are no chapters.
- `chapters` (page 157)
  Returns an NSArray containing information about the chapters in the receiver.
- `addChapters` (page 153)
  Adds chapters to the receiver using the information specified in the chapters array.
- `removeChapters` (page 172)
  Removes any existing chapters from the receiver.
- `startTimeOfChapter:` (page 176)
  Returns a QTTime structure that is the start time of the chapter having the specified 0-based index in the list of chapters.
- `chapterIndexForTime:` (page 157)
  Returns the 0-based index of the chapter that contains the specified movie time.

Inspecting Movie Properties

- `duration` (page 159)
  Returns the duration of a QTMovie object as a structure of type `QTTime`.
- `currentTime` (page 157)
  Returns the current time of a QTMovie object as a structure of type `QTTime`.
- `rate` (page 171)
  Returns the current rate of a QTMovie object.
- `volume` (page 179)
  Returns the movie's volume as a scalar value of type `float`.
- `muted` (page 170)
  Returns the movie's mute setting.
- `movieWithTimeRange:error:` (page 169)
  Returns a QTMovie object whose data is the data in the specified time range.
- `attributeForKey:` (page 155)
  Returns the current value of the movie attribute *`attributeKey`*.
- `movieAttributes` (page 169)
  Returns a dictionary containing the current values of all defined movie attributes.

Managing QTMovie Idling States

- `setIdling:` (page 174)
  Sets the movie to idle `YES` or not to idle `NO`.
- `idling` (page 162)
  Returns the current idling state of a QTMovie object.

Setting QTMovie Properties

- `setRate:` (page 175)
  Sets the movie's rate to *`rate`*.
- `setVolume:` (page 176)
  Sets the movie's volume to *`volume`*.
- `setMuted:` (page 175)
  Sets the movie's mute setting to *`mute`*.

Setting Movie Attributes

- `setAttribute:forKey:` (page 173)
  Set the movie attribute *`attributeKey`* to the value specified by the *`value`* parameter.
- `setMovieAttributes:` (page 175)
  Set the movie attributes using the key-value pairs specified in the dictionary *`attributes`*.

Supporting Aperture Modes

- `generateApertureModeDimensions` (page 160)
  Adds information to a QTMovie needed to support aperture modes for tracks created with applications and/or versions of QuickTime that did not support aperture mode dimensions.
- `removeApertureModeDimensions` (page 172)
  Removes aperture mode dimension information from a movie's tracks.

Getting and Setting Selection Times

- `selectionStart` (page 173)
  Returns the start time of the movie's current selection as a QTTime structure.
- `selectionEnd` (page 173)
  Returns the end point of the movie's current selection as a QTTime structure.
- `selectionDuration` (page 173)
  Returns the duration of the movie's current selection as a QTTime structure.
- `setSelection:` (page 176)
  Sets the movie's selection to *`selection`*.

Getting Movie Tracks

- `tracks` (page 177)
  Returns an array of QTTrack objects associated with the receiver.
- `tracksOfMediaType:` (page 178)
  Returns an array of tracks with the specified media type.

Getting Movie Images

- `posterImage` (page 170)
  Returns an NSImage for the poster frame of a QTMovie.
- `currentFrameImage` (page 157)
  Returns an NSImage for the frame at the current time in a QTMovie.
- `frameImageAtTime:` (page 159)
  Returns an NSImage for the frame at the time *`time`* in a QTMovie.
- `frameImageAtTime:withAttributes:error:` (page 159)
  Returns an NSImage*, CIImage*, CGImageRef, CVPixelBufferRef, or CVOpenGLTextureRef for the movie image at the specified time

Storing Movie Data

- `initToWritableDataReference:error:` (page 163)
  Creates a new storage container at the location specified by `dataReference` and returns a QTMovie object that has that container as its default data reference.

- `initToWritableFile:error:` (page 163)
  Useful for directly passing filenames and data objects. The QTMovie returned by this method is editable.
- `initToWritableData:error:` (page 162)
  Useful for directly passing filenames and data objects. The QTMovie returned by this method is editable.
- `movieFormatRepresentation` (page 169)
  Returns the movie's data in an NSData object.
- `writeToFile:withAttributes:` (page 179)
  Returns YES if the movie file was successfully created and NO otherwise.
- `writeToFile:withAttributes:error:` (page 179)
  Returns an NSError object if an error occurs and if errorPtr is non-NULL.

Editing a Movie

- `replaceSelectionWithSelectionFromMovie:` (page 172)
  Replaces the current selection in a QTMovie with the current selection in *`movie`*.
- `appendSelectionFromMovie:` (page 154)
  Appends to a QTMovie the current selection in *`movie`*.
- `insertSegmentOfMovie:timeRange:atTime:` (page 168)
  Inserts into a QTMovie at time *`time`* the selection in `movie` delimited by the time range *`range`*.
- `insertSegmentOfMovie:fromRange:scaledToRange:` (page 168)
  Inserts the specified segment from the movie into the receiver, scaled to the range `dstRange`.
- `insertEmptySegmentAt:` (page 168)
  inserts into a QTMovie an empty segment delimited by the range *`range`*.
- `deleteSegment:` (page 158)
  Deletes from a QTMovie the segment delimited by *`segment`*.
- `scaleSegment:newDuration:` (page 172)
  Scales the QTMovie segment delimited by the segment *`segment`* so that it will have the new duration *`newDuration`*.
- `addImage:forDuration:withAttributes:` (page 154)
  Adds an image for the specified duration to the receiver, using attributes specified in the attributes dictionary.

Saving a Movie

- `canUpdateMovieFile` (page 156)
  Indicates whether a movie file can be updated with changes made to the movie object.
- `updateMovieFile` (page 178)
  Updates the movie file of a QTMovie.

Getting QTMovie Primitives

- `quickTimeMovie` (page 170)
  Returns the QuickTime movie associated with a QTMovie object.
- `quickTimeMovieController` (page 171)
  Returns the QuickTime movie controller associated with a QTMovie object.

Getting and Setting QTMovie Delegates

- `delegate` (page 158)
  Returns the delegate of a QTMovie object.
- `setDelegate:` (page 174)
  Sets the movie's delegate to *`delegate`*.
- `externalMovie:` (page 180) *delegate method*
  This method is called, if implemented by a QTMovie delegate object, when an external movie needs to be found (usually for a wired action targeted at an external movie).
- `movieShouldTask:` (page 181) *delegate method*
  If a QTMovie object has a delegate and that delegate implements this method, that method will be called before QTKit performs the standard idle processing on a movie.
- `movie:shouldContinueOperation:withPhase:atPercent:withAttributes:` (page 181) *delegate method*
  If implemented, this method is called periodically during lengthy operations (such as exporting a movie).
- `movie:linkToURL:` (page 180) *delegate method*
  Called to handle the mcAction `mcActionLinkToURL`.

Class Methods

canInitWithDataReference:

Returns `YES` if the specified data reference can be used to initialize a QTMovie object.

```
+ (BOOL)canInitWithDataReference:(QTDataReference*)dataReference
```

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTMovie.h`

canInitWithFile:

Returns `YES` if the contents of the specified file can be used to initialize a QTMovie object.

```
+ (BOOL)canInitWithFile:(NSString *)fileName
```

Availability
Available in Mac OS X v10.3 and later.

Related Sample Code
QTKitAdvancedDocument
QTKitCreateMovie
QTKitFrameStepper
QTKitImport
QTKitPlayer

Declared In
`QTMovie.h`

canInitWithPasteboard:

Returns `YES` if the contents of the specified pasteboard can be used to initialize a QTMovie object.

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

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTMovie.h`

canInitWithURL:

Returns `YES` if the contents of the specified URL can be used to initialize a QTMovie object.

```
+ (BOOL)canInitWithURL:(NSURL *)url
```

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTMovie.h`

enterQTKitOnThread

Performs any QuickTime-specific initialization for the current (non-main) thread; must be paired with a call to `exitQTKitOnThread`.

```
+ (void)enterQTKitOnThread
```

Availability
Mac OS X v10.5 and later.

Declared In
`QTMovie.h`

enterQTKitOnThreadDisablingThreadSafetyProtection

Performs any QuickTime-specific initialization for the current (non-main) thread, allowing non-threadsafe components; must be paired with a call to `exitQTKitOnThread`.

```
+ (void)enterQTKitOnThreadDisablingThreadSafetyProtection
```

Availability
Mac OS X v10.5 and later.

Related Sample Code
QTKitThreadedExport

Declared In
`QTMovie.h`

exitQTKitOnThread

Performs any QuickTime-specific shut-down for the current (non-main) thread; must be paired with a call to `enterQTKitOnThread` or `enterQTKitOnThreadDisablingThreadSafetyProtection`.

```
+ (void)exitQTKitOnThread
```

Availability
Mac OS X v10.5 and later.

Related Sample Code
QTKitThreadedExport

Declared In
`QTMovie.h`

movie

Creates an empty QTMovie object.

```
+ (id)movie
```

Availability
Available in Mac OS X v10.3 and later.

Related Sample Code
QTAudioExtractionPanel
QTKitImport
QTKitMovieShuffler
QTKitPlayer

Declared In
`QTMovie.h`

movieFileTypes:

Returns an array of file types that can be opened as QuickTime movies.

```
+ (NSArray *)movieFileTypes:(QTMovieTypeOptions)types
```

Discussion
Passing zero as the options parameter returns an array of all the common file types that QuickTime can open in place on the current system. This array includes the file type `.mov` and `.mqv`, and any files types that can be opened using a movie importer that does not need to write data into a new file while performing the import. This array excludes any file types for still images and any file types that require an aggressive movie importer (for instance, the movie importer for text files). The following values can be used to include some or all of the file types that are normally excluded:

```
enum {
    QTIncludeStillImageTypes  =  1 << 0,
    QTIncludeTranslatableTypes =  1 << 1,
    QTIncludeAggressiveTypes =  1 << 2,
    QTIncludeCommonTypes = 0,
    QTIncludeAllTypes = 0xffff
} QTMovieFileTypeOptions;
```

Constants	Description
`QTIncludeStillImageTypes` Available in Mac OS X v10.3 and later.	This value adds to the array all file types for still images that can be opened using a graphics importer.
`QTIncludeTranslatableTypes` Available in Mac OS X v10.3 and later. Declared in `QTMovie.h`.	This value adds to the array all file types for files that can be opened using a movie importer but for which a new file must be created.
`QTIncludeAggressiveTypes` Available in Mac OS X v10.3 and later. Declared in `QTMovie.h`.	This value adds to the array all file types for files that can be opened using a movie importer but that are not commonly used in connection with movies (for instance, text or HTML files).
`QTIncludeCommonTypes` Available in Mac OS X v10.3 and later. Declared in `QTMovie.h`.	This value adds to the array all common file types that QuickTime can open in place on the current system.
`QTIncludeAllTypes` Available in Mac OS X v10.3 and later. Declared in `QTMovie.h`.	This value adds to the array all file types that QuickTime can open on the current system, using any available movie or graphics importer.

Related Sample Code
LiveVideoMixer2
LiveVideoMixer3

QTKitAdvancedDocument

Declared In
QTMovie.h

movieNamed:error:

Creates a QTMovie object initialized with the data from the QuickTime movie of the specified name in the application's bundle.

```
+ (id)movieNamed:(NSString *)name
    error:(NSError **)errorPtr
```

Discussion
If a QTMovie object cannot be created, an NSError object is returned in the location pointed to by *errorPtr*. Pass NIL if you do not want an NSError object returned.

Availability
Available in Mac OS X v10.3 and later.

Related Sample Code
CALayerEssentials

Declared In
QTMovie.h

movieTypesWithOptions:

Returns an array of UTIs that QuickTime can open.

```
+ (NSArray *)movieTypesWithOptions:(QTMovieFileTypeOptions)types
```

Discussion
This method gets an array of NSString objects that specify the uniform type identifiers (UTIs) for types of files that QuickTime can open. The types parameter is interpreted just like the types parameter to `+ (NSArray *)movieFileTypes:(QTMovieFileTypeOptions)types`.

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

Declared In
QTMovie.h

movieUnfilteredFileTypes

Returns an array of file types that can be used to initialize a QTMovie object.

```
+ (NSArray *)movieUnfilteredFileTypes
```

Availability
Available in Mac OS X v10.3 and later.

Related Sample Code
QTCoreImage101
QTCoreVideo103
QTCoreVideo201
QTKitMovieFrameImage
QTKitMovieShuffler

Declared In
`QTMovie.h`

movieUnfilteredPasteboardTypes

Returns an array of pasteboard types that can be used to initialize a QTMovie object.

```
+ (NSArray *)movieUnfilteredPasteboardTypes
```

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTMovie.h`

movieWithAttributes:error:

Creates a QTMovie object initialized with the attributes specified in *attributes*.

```
+ (id)movieWithAttributes:(NSDictionary *)attributes
    error:(NSError **)errorPtr
```

Discussion
If a QTMovie object cannot be created, an NSError object is returned in the location pointed to by *errorPtr*. Pass `NIL` if you do not want an NSError object returned.

A new QTMovie object is created using the specified attributes. There are three types of attributes that can be included in this dictionary:

- Attributes that specify the location of the movie data
- Attributes that specify how the movie is to be instantiated
- Attributes that specify playback characteristics of the movie or other properties of the QTMovie object

The following is a list of the keys that specify the location of the movie data; at least one of these must occur in the dictionary. If more than one occurs, the first one in the dictionary is used.

Attribute	Description
`QTMovieFileNameAttribute`	The file name string of a QTMovie object; the value for this key is of type `NSString`.
`QTMovieURLAttribute`	The URL of a QTMovie object; the value for this key is of type `NSURL`.

Attribute	Description
QTMovieDataReferenceAttribute	The data reference of a QTMovie object; the value for this key is of type QTDataReference.
QTMoviePasteboardAttribute	The pasteboard representation of a QTMovie object; the value for this key is of type NSPasteboard.
QTMovieDataAttribute	The data representation of a QTMovie object; the value for this key is of type NSData.

The following is a list of the keys that specify movie instantiation options; none of these keys is required. If a key is missing, the specified default value is used.

Attribute	Description
QTMovieFileOffset-Attribute	The file offset of a QTMovie. The value for this key is of type NSNumber, which is interpreted as a long long. The default is 0.
QTMovieResolveData-RefsAttribute	The resolved data reference of a QTMovie. The value for this key is of type NSNumber, which is interpreted as a BOOL. Default: YES. If NO, QTMovie makes no attempt to resolve any external data references in a movie file.
QTMovieAskUnresolved-DataRefsAttribute	The asked unresolved data reference setting of a QTMovie. The value for this key is of type NSNumber, which is interpreted as a BOOL. Default: YES. If YES, QTMovie may display a dialog box prompting the user to help resolve any unresolved external data references in a movie file.
QTMovieOpenAsync-OKAttribute	The allowed synchronization opening setting of a QTMovie. The value for this key is of type NSNumber, which is interpreted as a BOOL. Default: YES. If YES, the initialization method returns immediately with a non-nil QTMovie object; however, the movie data might not all be loaded yet, so you may need to check the movie load state before performing certain operations on the movie. If NO, the movie data is loaded synchronously; when the initialization method returns with a non-nil QTMovie object, its data is completely loaded.

The following is a list of the new keys that specify movie playback characteristics or other properties of the QTMovie object; most other existing movie attributes can be included as well.

Attribute	Description
QTMovieAutoAlternatesAttribute	The auto-alternate setting of a QTMovie object. The value for this key is of type NSNumber, interpreted as a BOOL.
QTMovieIsActiveAttribute	The active setting; the value for this key is of type NSNumber, interpreted as a BOOL.
QTMovieDelegateAttribute	The delegate for a QTMovie object. The value for this key is of type NSObject.

Availability
Available in Mac OS X v10.3 and later.

Declared In
QTMovie.h

movieWithData:error:

Creates a QTMovie object initialized with the data specified by *data*.

```
+ (id)movieWithData:(NSData *)data
    error:(NSError **)errorPtr
```

Discussion
If a QTMovie object cannot be created, an NSError object is returned in the location pointed to by *errorPtr*. Pass NIL if you do not want an NSError object returned.

Availability
Available in Mac OS X v10.3 and later.

Related Sample Code
QTKitCreateMovie
QTKitFrameStepper
QTKitImport

Declared In
QTMovie.h

movieWithDataReference:error:

Creates a QTMovie object intitalized with the data specified by the data reference *dataReference*.

```
+ (id)movieWithDataReference:(QTDataReference *)dataReference
    error:(NSError **)errorPtr
```

Discussion
If a QTMovie object cannot be created, an NSError object is returned in the location pointed to by *errorPtr*. Pass NIL if you do not want an NSError object returned.

Availability
Available in Mac OS X v10.3 and later.

Declared In
QTMovie.h

movieWithFile:error:

Creates a QTMovie object initialized with the data in the file specified by the name *fileName*.

```
+ (id)movieWithFile:(NSString *)fileName
    error:(NSError **)errorPtr
```

Discussion
The *fileName* is assumed to be a full path name for a file.

If a QTMovie object cannot be created, an NSError object is returned in the location pointed to by *errorPtr*. Pass NIL if you do not want an NSError object returned.

Availability
Available in Mac OS X v10.3 and later.

Related Sample Code
QTAudioExtractionPanel
QTKitCommandLine
QTKitMovieFrameImage
QTKitPlayer
SillyFrequencyLevels

Declared In
QTMovie.h

movieWithPasteboard:error:

Creates a QTMovie object initialized with the contents of the pasteboard specified by *pasteboard*.

```
+ (id)movieWithPasteboard:(NSPasteboard *)pasteboard
    error:(NSError **)errorPtr
```

Discussion
These contents can be a QuickTime movie (of type Movie), a file path, or data of type QTMoviePasteboardType.

If a QTMovie object cannot be created, an NSError object is returned in the location pointed to by *errorPtr*. Pass NIL if you do not want an NSError object returned.

Availability
Available in Mac OS X v10.3 and later.

Declared In
QTMovie.h

movieWithQuickTimeMovie:disposeWhenDone:error:

Creates a QTMovie object initialized with the data from an existing QuickTime movie *movie*.

```
+ (id)movieWithQuickTimeMovie:(Movie)movie
    disposeWhenDone:(BOOL)dispose
    error:(NSError **)errorPtr
```

Discussion
The dispose parameter (a BOOL) indicates whether the QTKit should call DisposeMovie on the specified movie when the QTMovie object is deallocated. Passing YES effectively transfers "ownership" of the Movie to the QTKit. (Note that most applications will probably want to pass YES; passing NO means that the application wants to call DisposeMovie itself, perhaps so that it can operate on a Movie after it has been disassociated with a QTMovie object.)

If a QTMovie object cannot be created, an NSError object is returned in the location pointed to by *errorPtr*. Pass NIL if you do not want an NSError object returned.

Note that command-line tools that pass NO for the *disposeWhenDone* parameter must make sure to release the active autorelease pool before calling DisposeMovie on the specified QuickTime movie. Failure to do this may result in a crash. Tools that need to call DisposeMovie before releasing the main autorelease pool can create another autorelease pool associated with the movie.

Availability
Available in Mac OS X v10.3 and later.
Not available to 64-bit applications.

Related Sample Code
QTKitCreateMovie

Declared In
QTMovie.h

movieWithURL:error:

Creates a QTMovie object initialized with the data in the URL specified by *url*.

```
+ (id)movieWithURL:(NSURL *)url
    error:(NSError **)errorPtr
```

Discussion
If a QTMovie object cannot be created, an NSError object is returned in the location pointed to by *errorPtr*. Pass NIL if you do not want an NSError object returned.

Availability
Available in Mac OS X v10.3 and later.

Related Sample Code
QTAudioExtractionPanel
QTKitCreateMovie
QTKitFrameStepper
QTKitPlayer
QTMetadataEditor

Declared In
QTMovie.h

Instance Methods

addChapters

Adds chapters to the receiver using the information specified in the chapters array.

```
- (void)addChapters:(NSArray *)chapters
    withAttributes:(NSDictionary *)attributes
    error:(NSError **)errorPtr
```

Discussion
Each array element is an NSDictionary containing key-value pairs. Currently two keys are defined for this dictionary, `QTMovieChapterName` and `QTMovieChapterStartTime`. The value for the `QTMovieChapterName` key is an NSString object that is the chapter name. The value for the `QTMovieChapterStartTime` key is an NSValue object that wraps a `QTTime` structure that indicates the start time of the chapter. The receiving QTMovie object must be editable or an exception will be raised.

The attributes dictionary specifies additional attributes for the chapters. Currently only one key is recognized for this dictionary, `QTMovieChapterTargetTrackAttribute`, which specifies the QTTrack in the receiver that is the target of the chapters; if none is specified, this method uses first video track in movie. If no video track is in the movie, this method uses the first audio track in the movie. If no audio track is in the movie, this method uses the first track in the movie. If an error occurs and errorPtr is non-NULL, then an NSError object is returned in that location.

Availability
Mac OS X v10.5 and later.

addImage:forDuration:withAttributes:

Adds an image for the specified duration to the receiver, using attributes specified in the attributes dictionary.

```
- (void)addImage:(NSImage *)image
    forDuration:(QTTime)duration
    withAttributes:(NSDictionary *)attributes
```

Discussion
Keys in the dictionary can be `QTAddImageCodecType` to select a codec type and `QTAddImageCodecQuality` to select a quality. Qualities are expected to be specified as NSNumbers, using the codec values like `codecNormalQuality`. (See ImageCompression.h for the complete list.) The attributes dictionary can also contain a value for the `QTTrackTimeScaleAttribute` key, which is used as the time scale of the new track, should one need to be created. The default time scale for a new track is 600.

Availability
Available in Mac OS X v10.3 and later.

Related Sample Code
WritableFileDemo

Declared In
`QTMovie.h`

appendSelectionFromMovie:

Appends to a QTMovie the current selection in *movie*.

```
- (void)appendSelectionFromMovie:(id)movie
```

Discussion
If the movie is not editable, this method raises an exception.

Availability
Available in Mac OS X v10.3 and later.

Declared In
QTMovie.h

attachToCurrentThread

Attaches the receiver to the current thread; returns YES if successful, NO otherwise.

```
- (BOOL)attachToCurrentThread
```

Availability
Mac OS X v10.5 and later.

Related Sample Code
QTKitThreadedExport

Declared In
QTMovie.h

attributeForKey:

Returns the current value of the movie attribute *attributeKey*.

```
- (id)attributeForKey:(NSString *)attributeKey
```

Discussion
A list of supported movie attributes and their acceptable values can be found in the "Constants" (page 182) section.

Availability
Available in Mac OS X v10.3 and later.

Related Sample Code
QTCoreVideo201
QTKitAdvancedDocument
QTKitFrameStepper
QTKitMovieShuffler
QTKitTimeCode

Declared In
QTMovie.h

autoplay

Sets a movie to start playing when a sufficient amount of media data is available.

```
- (void)autoplay
```

Discussion
The autoplay method configures a QTMovie object to begin playing as soon as enough data is available that the playback can continue uninterrupted to the end of the movie. This is most useful for movies being loaded from a remote URL or from an extremely slow local device. For movies stored on most local devices, this method has the same effect as the `-[QTMovie play]` method.

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

Declared In
`QTMovie.h`

canUpdateMovieFile

Indicates whether a movie file can be updated with changes made to the movie object.

```
- (BOOL)canUpdateMovieFile
```

Discussion
This method returns `NO` if any of the following conditions are true:

- The movie is not associated with a file.
- The movie is not savable (has `'nsav'` user data set to 1).
- The movie file is not writable.
- The movie file does not contain a movie atom (indicating that the movie was imported from a non-movie format).

Otherwise, the method returns `YES`.

Using this method, an application can check first to see if the movie file can be updated; if not, it can prompt the user for a new name and location of a file in which to save the updated movie.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTMovie.h`

chapterCount

Returns the number of chapters in the receiver, or 0 if there are no chapters.

```
- (NSInteger)chapterCount
```

Availability
Mac OS X v10.5 and later.

Declared In
`QTMovie.h`

chapterIndexForTime:

Returns the 0-based index of the chapter that contains the specified movie time.

```
- (NSInteger)chapterIndexForTime:(QTTime)time
```

Availability
Mac OS X v10.5 and later.

Declared In
`QTMovie.h`

chapters

Returns an NSArray containing information about the chapters in the receiver.

```
- (NSArray *)chapters
```

Discussion
Each array element is an NSDictionary containing key-value pairs. Currently two keys are defined for this dictionary, `QTMovieChapterName` and `QTMovieChapterStartTime`. The value for the `QTMovieChapterName` key is an NSString object that is the chapter name. The value for the `QTMovieChapterStartTime` key is an NSValue object that wraps a `QTTime` structure that indicates the start time of the chapter.

Availability
Mac OS X v10.5 and later.

Declared In
`QTMovie.h`

currentFrameImage

Returns an NSImage for the frame at the current time in a QTMovie.

```
- (NSImage *)currentFrameImage
```

Availability
Available in Mac OS X v10.3 and later.

See Also
- `frameImageAtTime:` (page 159)
- `posterImage` (page 170)

Declared In
`QTMovie.h`

currentTime

Returns the current time of a QTMovie object as a structure of type `QTTime`.

```
- (QTTime)currentTime
```

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTMovie.h`

delegate

Returns the delegate of a QTMovie object.

```
- (id)delegate
```

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTMovie.h`

deleteSegment:

Deletes from a QTMovie the segment delimited by *segment*.

```
- (void)deleteSegment:(QTTimeRange)segment
```

Discussion
If the movie is not editable, this method raises an exception.

Availability
Available in Mac OS X v10.3 and later.

Related Sample Code
QTKitCommandLine

Declared In
`QTMovie.h`

detachFromCurrentThread

Detaches the receiver from the current thread; returns YES if successful, NO otherwise.

```
- (BOOL)detachFromCurrentThread
```

Discussion
These methods allow applications to manage QTMovie objects on non-main threads. Before any QTKit operations can be performed on a secondary thread, either `enterQTKitOnThread` or `enterQTKitOnThreadDisablingThreadSafetyProtection` must be called, and `exitQTKitOnThread` must be called before exiting the thread. A QTMovie object can be migrated from one thread to another by first calling `detachFromCurrentThread` on the first thread and then `attachToCurrentThread` on the second thread.

Availability
Mac OS X v10.5 and later.

Related Sample Code
QTKitThreadedExport

Declared In
`QTMovie.h`

duration

Returns the duration of a QTMovie object as a structure of type `QTTime`.

```
- (QTTime)duration
```

Availability
Available in Mac OS X v10.3 and later.

Related Sample Code
QTKitCreateMovie
QTKitMovieShuffler
QTKitTimeCode

Declared In
`QTMovie.h`

frameImageAtTime:

Returns an NSImage for the frame at the time *time* in a QTMovie.

```
- (NSImage *)frameImageAtTime:(QTTime)time
```

Availability
Available in Mac OS X v10.3 and later.

See Also
- `currentFrameImage` (page 157)
- `posterImage` (page 170)

Declared In
`QTMovie.h`

frameImageAtTime:withAttributes:error:

Returns an NSImage*, CIImage*, CGImageRef, CVPixelBufferRef, or CVOpenGLTextureRef for the movie image at the specified time

```
- (void *)frameImageAtTime:(QTTime)time
    withAttributes:(NSDictionary *)attributes
    error:(NSError **)errorPtr
```

Discussion
if an error occurs and the desired type of image cannot be created, then this returns nil and sets errorPtr to an NSError * describing the error. The dictionary of attributes can contain these keys:

- `QTMovieFrameImageSize`
- `QTMovieFrameImageType`
- `QTMovieFrameImageRepresentationsType`
- `QTMovieFrameImageOpenGLContext`
- `QTMovieFrameImagePixelFormat`
- `QTMovieFrameImageInterlaced`
- `QTMovieFrameImageHighQuality`
- `QTMovieFrameImageSingleField`

Note: All images returned by this method are autoreleased objects and must be retained by the caller if they are to be accessed outside of the current run loop cycle.

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

Declared In
`QTMovie.h`

generateApertureModeDimensions

Adds information to a QTMovie needed to support aperture modes for tracks created with applications and/or versions of QuickTime that did not support aperture mode dimensions.

```
- (void)generateApertureModeDimensions
```

Discussion
If the image descriptions in video tracks lack tags describing clean aperture and pixel aspect ratio information, the media data is scanned to see if the correct values can be divined and attached. Then the aperture mode dimensions are calculated and set. Afterwards, the `QTTrackHasApertureModeDimensionsAttribute` property will be set to `YES` for those tracks. Tracks that do not support aperture modes are not changed.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTMovie.h`

gotoBeginning

Repositions the play position to the beginning of the movie.

```
- (void)gotoBeginning
```

Discussion
If the movie is playing, the movie continues playing from the new position.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTMovie.h`

gotoEnd

Repositions the play position to the end of the movie.

```
- (void)gotoEnd
```

Discussion
If the movie is playing in one of the looping modes, the movie continues playing accordingly; otherwise, play stops.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTMovie.h`

gotoNextSelectionPoint

Repositions the movie to the next selection point.

```
- (void)gotoNextSelectionPoint
```

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTMovie.h`

gotoPosterFrame

Repositions the play position to the movie's poster time.

```
- (void)gotoPosterFrame
```

Discussion
If no poster time is defined, the movie jumps to the beginning. If the movie is playing, the movie continues playing from the new position.

gotoPreviousSelectionPoint

Repositions the movie to the previous selection point.

```
- (void)gotoPreviousSelectionPoint
```

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTMovie.h`

hasChapters

Returns YES if the receiver has chapters, NO otherwise.

```
- (BOOL)hasChapters
```

Availability
Mac OS X v10.5 and later.

Declared In
`QTMovie.h`

idling

Returns the current idling state of a QTMovie object.

```
- (BOOL)idling
```

Discussion
This method allows you to manage the idling state of a QTMovie object, that is, whether it is being tasked. Note that movies attached to a background thread should not be idled; if they are idled, unexpected behavior can result.

initToWritableData:error:

Useful for directly passing filenames and data objects. The QTMovie returned by this method is editable.

```
- (id)initToWritableData:(NSMutableData *)data
    error:(NSError **)errorPtr
```

Discussion
These methods––`initToWritableDataReference:error:`, `initToWritableFile:error:` and `initToWritableData:error:`––create an empty, writable storage container to which media data can be added (for example, using the QTMovie `addImage` method). The methods return QTMovie objects associated with those containers.

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

Declared In
`QTMovie.h`

initToWritableDataReference:error:

Creates a new storage container at the location specified by `dataReference` and returns a QTMovie object that has that container as its default data reference.

```
- (id)initToWritableDataReference:(QTDataReference *)dataReference
    error:(NSError **)errorPtr
```

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

Declared In
`QTMovie.h`

initToWritableFile:error:

Useful for directly passing filenames and data objects. The QTMovie returned by this method is editable.

```
- (id)initToWritableFile:(NSString *)filename
    error:(NSError **)errorPtr
```

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

Related Sample Code
QTKitCreateMovie
WritableFileDemo

Declared In
`QTMovie.h`

initWithAttributes:error:

Initializes a QTMovie object with the attributes specified in *`attributes`*.

```
- (id)initWithAttributes:(NSDictionary *)attributes
    error:(NSError **)errorPtr
```

Discussion
If a QTMovie object cannot be created, an NSError object is returned in the location pointed to by *`errorPtr`*. Pass `NIL` if you do not want an NSError object returned.

A new QTMovie object is created using the specified attributes. There are three types of attributes that can be included in this dictionary:

- Attributes that specify the location of the movie data
- Attributes that specify how the movie is to be instantiated
- Attributes that specify playback characteristics of the movie or other properties of the QTMovie object

The following is a list of the keys that specify the location of the movie data; at least one of these must occur in the dictionary. If more than one occurs, the first one in the dictionary is used.

Attribute	Description
QTMovieFileNameAttribute	The file name string of a QTMovie object; the value for this key is of type NSString.
QTMovieURLAttribute	The URL of a QTMovie object; the value for this key is of type NSURL.
QTMovieDataReferenceAttribute	The data reference of a QTMovie object; the value for this key is of type QTDataReference.
QTMoviePasteboardAttribute	The pasteboard of a QTMovie object; the value for this key is of type NSPasteboard.
QTMovieDataAttribute	The data of a QTMovie object; the value for this key is of type NSData.

The following is a list of the keys that specify movie instantiation options; none of these keys is required. If a key is missing, the specified default value is used.

Attribute	Description
QTMovieFileOffsetAttribute	The file offset of a QTMovie. The value for this key is of type NSNumber, which is interpreted as a long long. The default is 0.
QTMovieResolveData-RefsAttribute	The resolved data reference setting of a QTMovie. The value for this key is of type NSNumber, which is interpreted as a BOOL. Default: YES.
QTMovieAskUnresolved-DataRefsAttribute	The asked unresolved data reference of a QTMovie. The value for this key is of type NSNumber, which is interpreted as a BOOL. Default: YES.
QTMovieOpenAsyncOKAttribute	The opened synchronization of a QTMovie. The value for this key is of type NSNumber, which is interpreted as a BOOL. Default: YES.

The following is a list of the new keys that specify movie playback characteristics or other properties of the QTMovie object; most other existing movie attributes can be included as well.

Attribute	Description
QTMovieAuto-AlternatesAttribute	The auto-alternate of a QTMovie object. The value for this key is of type NSNumber, interpreted as a BOOL.
QTMovieIsActiveAttribute	The active setting; the value for this key is of type NSNumber, interpreted as a BOOL.
QTMovieDontInteract-WithUserAttribute	When set in a dictionary passed to movieWithAttributes or initWithAttributes, this prevents QuickTime from interacting with the user during movie initialization. The value for this key is of type NSNumber, interpreted as a BOOL.
QTMovieDelegateAttribute	The delegate for a QTMovie object. The value for this key is of type NSObject.

Availability
Available in Mac OS X v10.3 and later.

Related Sample Code
QTKitAdvancedDocument

Declared In
QTMovie.h

initWithData:error:

Initializes a QTMovie object with the data specified by *data*.

```
- (id)initWithData:(NSData *)data
    error:(NSError **)errorPtr
```

Discussion
If a QTMovie object cannot be created, an NSError object is returned in the location pointed to by *errorPtr*. Pass NIL if you do not want an NSError object returned.

Availability
Available in Mac OS X v10.3 and later.

Declared In
QTMovie.h

initWithDataReference:error:

Initializes a QTMovie object with the data reference setting specified by *dataReference*.

```
- (id)initWithDataReference:(QTDataReference *)dataReference
    error:(NSError **)errorPtr
```

Discussion
If a QTMovie object cannot be created, an NSError object is returned in the location pointed to by *errorPtr*. Pass NIL if you do not want an NSError object returned.

Availability
Available in Mac OS X v10.3 and later.

Declared In
QTMovie.h

initWithFile:error:

Initializes a QTMovie object with the data in the file specified by the name *fileName*.

```
- (id)initWithFile:(NSString *)fileName
    error:(NSError **)errorPtr
```

Discussion
The *fileName* is assumed to be a full path name for a file. If a QTMovie object cannot be created, an NSError object is returned in the location pointed to by *errorPtr*. Pass NIL if you do not want an NSError object returned.

Note that alias files should not be passed into this method; the client application is responsible for resolving aliases before handing them to QTKit methods.

Availability
Available in Mac OS X v10.3 and later.

Related Sample Code
QTCoreImage101
QTKitButtonTester
QTKitMovieShuffler
QTQuartzPlayer
ViewController

Declared In
QTMovie.h

initWithMovie:timeRange:error:

Initializes a QTMovie object with some or all of the data from an existing QTMovie object *movie*.

```
- (id)initWithMovie:(QTMovie *)movie
    timeRange:(QTTimeRange)range
    error:(NSError **)errorPtr
```

Discussion
The section of data used is delimited by the range *range*. If a QTMovie object cannot be created, an NSError object is returned in the location pointed to by *errorPtr*. Pass NIL if you do not want an NSError object returned.

Availability
Available in Mac OS X v10.3 and later.

Declared In
QTMovie.h

initWithPasteboard:error:

Initializes a QTMovie object with the contents of the pasteboard specified by *pasteboard*.

```
- (id)initWithPasteboard:(NSPasteboard *)pasteboard
    error:(NSError **)errorPtr
```

Discussion
These contents can be a QuickTime movie (of type Movie), a file path, or data of type QTMoviePasteBoardType. If a QTMovie object cannot be created, an NSError object is returned in the location pointed to by *errorPtr*. Pass NIL if you do not want an NSError object returned.

Availability
Available in Mac OS X v10.3 and later.

Declared In
QTMovie.h

initWithQuickTimeMovie:disposeWhenDone:error:

Initializes a QTMovie object with the data from an existing QuickTime movie *movie*.

```
- (id)initWithQuickTimeMovie:(Movie)movie
    disposeWhenDone:(BOOL)dispose
    error:(NSError **)errorPtr
```

Discussion
This is the designated initializer for the QTMovie class. The `dispose` parameter (a `BOOL`) indicates whether the QTKit should call `DisposeMovie` on the specified movie when the QTMovie object is deallocated. Passing `YES` effectively transfers "ownership" of the Movie to the QTKit. (Note that most applications will probably want to pass `YES`; passing `NO` means that the application wants to call `DisposeMovie` itself, perhaps so that it can operate on a Movie after it has been disassociated from a QTMovie object.)

If a QTMovie object cannot be created, an NSError object is returned in the location pointed to by *errorPtr*. Pass `NIL` if you do not want an NSError object returned.

Availability
Available in Mac OS X v10.3 and later.
Not available to 64-bit applications.

Declared In
QTMovie.h

initWithURL:error:

Initializes a QTMovie object with the data in the URL specified by *url*.

```
- (id)initWithURL:(NSURL *)url
    error:(NSError **)errorPtr
```

Discussion
If a QTMovie object cannot be created, an NSError object is returned in the location pointed to by *errorPtr*. Pass `NIL` if you do not want an NSError object returned.

Availability
Available in Mac OS X v10.3 and later.

Related Sample Code
QTKitFrameStepper

Declared In
QTMovie.h

insertEmptySegmentAt:

inserts into a QTMovie an empty segment delimited by the range *range*.

```
- (void)insertEmptySegmentAt:(QTTimeRange)range
```

Discussion
If the movie is not editable, this method raises an exception.

Availability
Available in Mac OS X v10.3 and later.

Declared In
QTMovie.h

insertSegmentOfMovie:fromRange:scaledToRange:

Inserts the specified segment from the movie into the receiver, scaled to the range `dstRange`.

```
- (void)insertSegmentOfMovie:(QTMovie *)movie
    fromRange:(QTTimeRange)srcRange
    scaledToRange:(QTTimeRange)dstRange
```

Discussion
This is essentially an Add Scaled operation on a movie. If the movie is not editable, this method raises an exception.

Availability
Available in Mac OS X v10.3 and later.

Declared In
QTMovie.h

insertSegmentOfMovie:timeRange:atTime:

Inserts into a QTMovie at time *time* the selection in `movie` delimited by the time range *range*.

```
- (void)insertSegmentOfMovie:(QTMovie *)movie
    timeRange:(QTTimeRange)range
    atTime:(QTTime)time
```

Discussion
If the movie is not editable, this method raises an exception.

Availability
Available in Mac OS X v10.3 and later.

Related Sample Code
QTKitMovieShuffler

Declared In
QTMovie.h

movieAttributes

Returns a dictionary containing the current values of all defined movie attributes.

```
- (NSDictionary *)movieAttributes
```

Discussion
A list of supported movie attributes and their acceptable values can be found in the "Constants" (page 182) section.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTMovie.h`

movieFormatRepresentation

Returns the movie's data in an NSData object.

```
- (NSData *)movieFormatRepresentation
```

Availability
Available in Mac OS X v10.3 and later.

See Also
`- writeToFile:withAttributes:` (page 179)

Related Sample Code
QTMetadataEditor

Declared In
`QTMovie.h`

movieWithTimeRange:error:

Returns a QTMovie object whose data is the data in the specified time range.

```
- (id)movieWithTimeRange:(QTTimeRange)range
    error:(NSError **)errorPtr
```

Discussion
If a QTMovie object cannot be created, an NSError object is returned in the location pointed to by *errorPtr*. Pass `NIL` if you do not want an NSError object returned.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTMovie.h`

muted

Returns the movie's mute setting.

```
- (BOOL)muted
```

Availability
Available in Mac OS X v10.3 and later.

Declared In
QTMovie.h

play

Plays the movie.

```
- (void)play
```

Availability
Available in Mac OS X v10.3 and later.

Related Sample Code
TrackFormatDemo
VideoViewer

Declared In
QTMovie.h

posterImage

Returns an NSImage for the poster frame of a QTMovie.

```
- (NSImage *)posterImage
```

Availability
Available in Mac OS X v10.3 and later.

See Also
- currentFrameImage (page 157),
- frameImageAtTime: (page 159)

Related Sample Code
QTKitMovieShuffler

Declared In
QTMovie.h

quickTimeMovie

Returns the QuickTime movie associated with a QTMovie object.

```
- (Movie)quickTimeMovie
```

Availability
Available in Mac OS X v10.3 and later.
Not available to 64-bit applications.

See Also
- quickTimeMovieController (page 171)

Related Sample Code
QTCoreVideo103
QTCoreVideo201
QTCoreVideo202
QTKitTimeCode
VideoViewer

Declared In
QTMovie.h

quickTimeMovieController

Returns the QuickTime movie controller associated with a QTMovie object.

```
- (MovieController)quickTimeMovieController
```

Availability
Available in Mac OS X v10.3 and later.
Not available to 64-bit applications.

See Also
- quickTimeMovie (page 170)

Related Sample Code
QTKitMovieShuffler

Declared In
QTMovie.h

rate

Returns the current rate of a QTMovie object.

```
- (float)rate
```

Availability
Available in Mac OS X v10.3 and later.

Related Sample Code
QTKitMovieShuffler

Declared In
QTMovie.h

removeApertureModeDimensions

Removes aperture mode dimension information from a movie's tracks.

```
- (void)removeApertureModeDimensions
```

Discussion
This method does not attempt to modify sample descriptions, so it may not completely reverse the effects of `generateApertureModeDimensions`. It sets the `QTMovieHasApertureModeDimensionsAttribute` property to `NO`.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTMovie.h`

removeChapters

Removes any existing chapters from the receiver.

```
- (BOOL)removeChapters
```

Discussion
Returns YES if either the receiver had no chapters or the chapters were successfully removed from the receiver. Returns NO if the chapters could not for some reason be removed from the receiver. The receiving QTMovie object must be editable or an exception will be raised.

Availability
Mac OS X v10.5 and later.

Declared In
`QTMovie.h`

replaceSelectionWithSelectionFromMovie:

Replaces the current selection in a QTMovie with the current selection in *`movie`*.

```
- (void)replaceSelectionWithSelectionFromMovie:(id)movie
```

Discussion
If the movie is not editable, this method raises an exception.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTMovie.h`

scaleSegment:newDuration:

Scales the QTMovie segment delimited by the segment *`segment`* so that it will have the new duration *`newDuration`*.

```
- (void)scaleSegment:(QTTimeRange)segment
    newDuration:(QTTime)newDuration
```

Discussion
If the movie is not editable, this method raises an exception.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTMovie.h`

selectionDuration

Returns the duration of the movie's current selection as a QTTime structure.

```
- (QTTime)selectionDuration
```

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTMovie.h`

selectionEnd

Returns the end point of the movie's current selection as a QTTime structure.

```
- (QTTime)selectionEnd
```

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTMovie.h`

selectionStart

Returns the start time of the movie's current selection as a QTTime structure.

```
- (QTTime)selectionStart
```

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTMovie.h`

setAttribute:forKey:

Set the movie attribute *attributeKey* to the value specified by the *value* parameter.

```
- (void)setAttribute:(id)value
    forKey:(NS String *)attributeKey
```

Discussion
A list of supported movie attributes and their acceptable values can be found in the "Constants" (page 182) section.

Availability
Available in Mac OS X v10.3 and later.

Related Sample Code
QTCoreVideo103
QTCoreVideo201
QTKitCommandLine
QTKitMovieShuffler
ViewController

Declared In
`QTMovie.h`

setCurrentTime:

Sets the movie's current time setting to `time`.

```
- (void)setCurrentTime:(QTTime)time
```

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTMovie.h`

setDelegate:

Sets the movie's delegate to *`delegate`*.

```
- (void)setDelegate:(id)delegate
```

Availability
Available in Mac OS X v10.3 and later.

Related Sample Code
QTKitProgressTester

Declared In
`QTMovie.h`

setIdling:

Sets the movie to idle `YES` or not to idle `NO`.

```
- (void)setIdling:(BOOL)state
```

Discussion
This method allows you to manage the idling state of a QTMovie object, that is, whether it is being tasked. Note that movies attached to a background thread should not be idled; if they are idled, unexpected behavior can result.

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

Declared In
QTMovie.h

setMovieAttributes:

Set the movie attributes using the key-value pairs specified in the dictionary *attributes*.

```
- (void)setMovieAttributes:(NSDictionary *)attributes
```

Discussion
A list of supported movie attributes and their acceptable values can be found in the "Constants" (page 182) section.

Availability
Available in Mac OS X v10.3 and later.

Declared In
QTMovie.h

setMuted:

Sets the movie's mute setting to *mute*.

```
- (void)setMuted:(BOOL)mute
```

Discussion
Note that this does not affect the volume.

Availability
Available in Mac OS X v10.3 and later.

Declared In
QTMovie.h

setRate:

Sets the movie's rate to *rate*.

```
- (void)setRate:(float)rate
```

Discussion
For instance, 0.0 is stop, 1.0 is playback at normal speed, 2.0 is twice normal speed, and so on.

Availability
Available in Mac OS X v10.3 and later.

Related Sample Code
QTCoreVideo102
QTCoreVideo103
QTCoreVideo201
QTCoreVideo202
QTCoreVideo301

Declared In
`QTMovie.h`

setSelection:

Sets the movie's selection to *`selection`*.

```
- (void)setSelection:(QTTimeRange)selection
```

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTMovie.h`

setVolume:

Sets the movie's volume to *`volume`*.

```
- (void)setVolume:(float)volume
```

Discussion
Note that this does not affect the movie's stored settings.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTMovie.h`

startTimeOfChapter:

Returns a QTTime structure that is the start time of the chapter having the specified 0-based index in the list of chapters.

```
- (QTTime)startTimeOfChapter:(NSInteger)chapterIndex
```

Availability
Mac OS X v10.5 and later.

Declared In
QTMovie.h

stepBackward

Sets the movie backward a single frame.

```
- (void)stepBackward
```

Availability
Available in Mac OS X v10.3 and later.

Declared In
QTMovie.h

stepForward

Sets the movie forward a single frame.

```
- (void)stepForward
```

Availability
Available in Mac OS X v10.3 and later.

Declared In
QTMovie.h

stop

Stops the movie playing.

```
- (void)stop
```

Availability
Available in Mac OS X v10.3 and later.

Related Sample Code
QTAudioExtractionPanel
QTKitMovieShuffler
QTKitPlayer

Declared In
QTMovie.h

tracks

Returns an array of QTTrack objects associated with the receiver.

```
- (NSArray *)tracks
```

Availability
Available in Mac OS X v10.3 and later.

Related Sample Code
QTMetadataEditor
TrackFormatDemo

Declared In
`QTMovie.h`

tracksOfMediaType:

Returns an array of tracks with the specified media type.

```
- (NSArray *)tracksOfMediaType:(NSString *)type
```

Discussion
The type parameter should be one of the media types defined by constants in `QTMedia.h` beginning with "QTMediaType", for instance, `QTMediaTypeVideo`.

Availability
Available in Mac OS X v10.3 and later.

Related Sample Code
QTKitTimeCode

Declared In
`QTMovie.h`

updateMovieFile

Updates the movie file of a QTMovie.

```
- (BOOL)updateMovieFile
```

Discussion
Returns `YES` if the update succeeds and `NO` otherwise.

Availability
Available in Mac OS X v10.3 and later.

Related Sample Code
QTKitCommandLine
QTMetadataEditor
WritableFileDemo

Declared In
`QTMovie.h`

volume

Returns the movie's volume as a scalar value of type `float`.

```
- (float)volume
```

Discussion
The valid range is 0.0 to 1.0.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTMovie.h`

writeToFile:withAttributes:

Returns YES if the movie file was successfully created and NO otherwise.

```
- (BOOL)writeToFile:(NSString *)fileNamewithAttributes
    :(NSDictionary *)attributes
```

Discussion
This method returns YES if the movie file was successfully created and NO otherwise. NO will also be returned if the load state of the target is less than `QTMovieLoadStateComplete`, in which case no attempt is made to write the QTMovie into a file. If the dictionary *`attributes`* contains an object whose key is `QTMovieFlatten`, then the movie is flattened into the specified file. If the dictionary *`attributes`* contains an object whose key is `QTMovieExport`, then the movie is exported into the specified file using a movie exporter whose type is specified by the value of the key `QTMovieExportType`. The value associated with the `QTMovieExportSettings` key should be an object of type NSData that contains an atom container of movie export settings.

Availability
Available in Mac OS X v10.3 and later.

See Also
`- movieFormatRepresentation` (page 169)

Related Sample Code
QTKitCommandLine
QTKitMovieShuffler
QTKitProgressTester
QTKitThreadedExport

Declared In
`QTMovie.h`

writeToFile:withAttributes:error:

Returns an NSError object if an error occurs and if errorPtr is non-NULL.

```
- (BOOL)writeToFile:(NSString *)fileName
    withAttributes:(NSDictionary *)attributes
    error:(NSError **)errorPtr
```

Discussion
The method operates exactly like the existing `QTMovie writeToFile:withAttributes` method.

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

See Also
`- movieFormatRepresentation` (page 169)

Declared In
`QTMovie.h`

Delegate Methods

externalMovie:

This method is called, if implemented by a QTMovie delegate object, when an external movie needs to be found (usually for a wired action targeted at an external movie).

```
- (QTMovie *)externalMovie:(NSDictionary *)dictionary
```

Discussion
The keys for the dictionary in this delegate method are: *`QTMovieTargetIDNotificationParameter`* and *`QTMovieTargetNameNotificationParameter`*. The *`QTMovieTargetIDNotificationParameter`* key indicates that the delegate should return a QTMovie object that has the specified movie ID. The *`QTMovieTargetNameNotificationParameter`* key indicates that the delegate should return a QTMovie object that has the specified movie name.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTMovie.h`

movie:linkToURL:

Called to handle the mcAction `mcActionLinkToURL`.

```
- (BOOL)movie:(QTMovie *)movielinkToURL
    :(NSURL *)url
```

Discussion
Most applications will not need to install a delegate to handle this.

Availability
Available in Mac OS X v10.3 and later.

Declared In
QTMovie.h

movie:shouldContinueOperation:withPhase:atPercent:withAttributes:

If implemented, this method is called periodically during lengthy operations (such as exporting a movie).

```
- (BOOL)movie:(QTMovie *)movieshouldContinueOperation
    :(NSString *)opwithPhase
    :(QTMovieOperationPhase)phaseatPercent
    :(NSNumber *)percentwithAttributes
    :(NSDictionary *)attributes
```

Discussion
A delegate can implement this method. The `op` string is a localized string that indicates what the operation is. The `phase` indicates whether the operation is just beginning, ending, or is at a certain percentage of completion. If the phase is `QTMovieOperationUpdatePercentPhase`, then the `percent` parameter indicates the percentage of the operation completed. The `attributes` dictionary may be `NIL`; if not `NIL`, it is the same dictionary passed to a QTMovie method that caused the lengthy operation (for example, the `attributes` dictionary passed to `writeToFile`). The constants for this method are defined as follows:

```
typedef enum {
    QTMovieOperationBeginPhase = movieProgressOpen,
    QTMovieOperationUpdatePercentPhase = movieProgressUpdatePercent,
    QTMovieOperationEndPhase = movieProgressClose
}
```

Availability
Available in Mac OS X v10.3 and later.

Declared In
QTMovie.h

movieShouldTask:

If a QTMovie object has a delegate and that delegate implements this method, that method will be called before QTKit performs the standard idle processing on a movie.

```
- (BOOL)movieShouldTask:(id)movie
```

Discussion
The delegate can cancel that normal processing by returning `YES`.

Availability
Available in Mac OS X v10.3 and later.

Declared In
QTMovie.h

Constants

The following constants specify the movie attributes that you can get and set using the `movieAttributes` and `setMovieAttributes` methods. To get or set a single attribute, use `attributeForKey` or `setAttribute`.

Constant	Description
`QTMovieActiveSegmentAttribute`	The active segment of a QTMovie object; the value for this key is of type `NSValue`, interprete `QTTimeRange` structure. (**Deprecated.** This constant is available in Mac OS X 10.4 and later, bu in Mac OS X 10.5.)
`QTMovieApertureModeAttribute`	Sets the aperture mode attribute on a QTMovie object to indicate whether aspect ratio and c correction should be performed. When a movie is in clean, production, or encoded pixels ape each track's dimensions are overridden by special dimensions for that mode. The original trac are preserved and can be restored by setting the movie into classic aperture mode. Aperture saved in movies. The associated value is of type `NSString` and is assumed to be one of the foll `QTMovieApertureModeClassic`. No aspect ratio or clean aperture correction is performed. default aperture mode and provides compatibility with behavior in QuickTime 7.0.x and earlie `-[QTTrack setDimensions]`, the movie is automatically switched to classic mode. `QTMovieApertureModeClean`. An aperture mode for general display. Where possible, video wi at the correct pixel aspect ratio, trimmed to the clean aperture. A movie in clean aperture mo track's dimensions to match the size returned by `-[QTTrack apertureModeDimensionsForMode:QTMovieApertureModeClean]`. `QTMovieApertureModeProduction`. `QTMovieApertureModeProduction`. An aperture mode for modal use in authoring applica possible, video will be displayed at the correct pixel aspect ratio, but without trimming to the so that the edge processing region can be viewed. A movie in production aperture mode sets dimensions to match the size returned by `-[QTTrack apertureModeDimensionsForMode:QTMovieApertureModeProduction]`. `QTMovieApertureModeEncodedPixels`. `QTMovieApertureModeEncodedPixels`. An aperture mode for technical use. Displays all e with no aspect ratio or clean aperture compensation. A movie in encoded pixels aperture mo track's dimensions to match the size returned by `-[QTTrack apertureModeDimensionsForMode:QTMovieApertureModeEncodedPixels]`.
`QTMovieAutoAlternatesAttribute`	The auto-alternate state of a QTMovie object. The value for this key is of type `NSNumber`, inte `BOOL`.
`QTMovieCopyrightAttribute`	The copyright string of a QTMovie object; the value for this key is of type `NSString`.
`QTMovieCreationTimeAttribute`	The creation time of a QTMovie object; the value for this key is of type `NSDate`.
`QTMovieCurrentSizeAttribute`	The current size of a QTMovie object; the value for this key is of type `NSValue`, interpreted as structure.
`QTMovieCurrentTimeAttribute`	The current time of a QTMovie object; the value for this key is of type `NSValue`, interpreted a structure.

Constant	Description
QTMovieDataSize-Attribute	The data size of a QTMovie. The value for this key is of type NSNumber, which is interpreted as
QTMovieDelegate-Attribute	The delegate for a QTMovie object. The value for this key is of type NSObject.
QTMovieDisplay-NameAttribute	The display name of a QTMovie object. A display name is stored as user data in a movie file a differ from the base name of the movie's filename or URL. The value for this key is of type NS
QTMovieDontInteract-WithUserAttribute	When set in a dictionary passed to movieWithAttributes or initWithAttributes, this preve from interacting with the user during movie initialization. The value for this key is of type NSN interpreted as a BOOL.
QTMovieDuration-Attribute	The duration of a QTMovie object; the value for this key is of type NSValue, interpreted as a QTT
QTMovieEditable-Attribute	The editable setting; the value for this key is of type NSNumber, interpreted as a BOOL. This va the movie can be edited.
QTMovieFileName-Attribute	The file name string of a QTMovie object; the value for this key is of type NSString.
QTMovieHasAperture-ModeDimensions-Attribute	The aperture mode dimensions set on any track in this QTMovie object, even if those dimens identical to the classic dimensions (as is the case for content with square pixels and no edge- region). The value for this key is of type NSNumber, interpreted as a BOOL.
QTMovieHasAudio-Attribute	The audio data setting; the value for this key is of type NSNumber, interpreted as a BOOL. This the movie contains audio data.
QTMovieHasDuration-Attribute	The duration setting; the value for this key is of type NSNumber, interpreted as a BOOL. This va the movie has a duration.
QTMovieHasVideo-Attribute	The video data setting; the value for this key is of type NSNumber, interpreted as a BOOL. This the movie contains video data.
QTMovieIsActive-Attribute	The active setting; the value for this key is of type NSNumber, interpreted as a BOOL.
QTMovieIsInteractive-Attribute	The interactive setting; the value for this key is of type NSNumber, interpreted as a BOOL. This the movie is interactive.
QTMovieIsLinear-Attribute	The linear setting; the value for this key is of type NSNumber, interpreted as a BOOL. This valu movie is linear, as opposed to a non-linear QuickTime VR movie.
QTMovieIsSteppable-Attribute	The steppable setting; the value for this key is of type NSNumber, interpreted as a BOOL. This the movie can step from frame to frame.

Constant	Description

QTMovieLoadState-Attribute	The load state value; the value for this key is of type NSNumber, interpreted as a long.
	enum {
	QTMovieLoadStateError = -1L, // an error occurred while loading th
	QTMovieLoadStateLoading = 1000, // the movie is loading
	QTMovieLoadStateLoaded = 2000, // the movie atom has loaded; it's s
	query movie properties
	QTMovieLoadStatePlayable = 10000, // the movie has loaded enough me
	to begin playing
	QTMovieLoadStatePlaythroughOK = 20000, // the movie has loaded enou
	data to play through to the end
	QTMovieLoadStateComplete = 100000L // the movie has loaded complete
	};
	
	The attributeForKey: QTMovieLoadStateAttribute returns an NSNumber that wraps
	the enumerated constants shown above are the possible values of that long integer.
	Mac OS X v10.5 and later.

QTMovieLoops-Attribute	The looping setting; the value for this key is of type NSNumber, interpreted as a BOOL. This valu movie is set to loop.
QTMovieLoopsBackAnd-ForthAttribute	The palindrome looping setting; the value for this key is of type NSNumber, interpreted as a BO is YES if the movie is set to loop back and forth. Note that QTMovieLoopsAttribute and QTMovieLoopsBackAndForthAttribute are independent and indeed exclusive. QTMovieLoo is used to get and set the state of normal looping; QTMovieLoopsBackAndForthAttribute and set the state of palindrome looping.
QTMovieModification-TimeAttribute	The modification time of a QTMovie object; the value for this key is of type NSDate.
QTMovieMuted-Attribute	The mute setting; the value for this key is of type NSNumber, interpreted as a BOOL. This value movie volume is muted.
QTMovieNatural-SizeAttribute	The natural size of a QTMovie object; the value for this key is of type NSValue, interpreted as structure.
QTMoviePlaysAll-FramesAttribute	The play-all-frames setting; the value for this key is of type NSNumber, interpreted as a BOOL. YES if the movie will play all frames.
QTMoviePlays-SelectionOnly-Attribute	The play-selection setting; the value for this key is of type NSNumber, interpreted as a BOOL. Th if the movie will play only the current selection.
QTMoviePosterTime-Attribute	The movie poster time of a QTMovie object; the value for this key is of type NSValue, interprete structure.
QTMoviePreferred-MutedAttribute	The preferred mute setting; the value for this key is of type NSNumber, interpreted as a BOOL. YES if the movie preferred mute setting is muted.
QTMoviePreferred-RateAttribute	The preferred rate; the value for this key is of type NSNumber, interpreted as a float.
QTMoviePreferred-VolumeAttribute	The preferred volume; the value for this key is of type NSNumber, interpreted as a float.

Constant	Description
QTMoviePreview-ModeAttribute	The preview mode setting; the value for this key is of type NSNumber, interpreted as a BOOL. YES if the movie is in preview mode.
QTMoviePreviewRange-Attribute	The preview range of a QTMovie object; the value for this key is of type NSValue, interpreted QTTimeRange structure.
QTMovieRateAttribute	The movie rate; the value for this key is of type NSNumber, interpreted as a float.
QTMovieRateChanges-PreservePitch-Attribute	When the playback rate is not unity, audio must be resampled in order to play at the new rate resampling affects the pitch of the audio (for example, playing at 2x speed raises the pitch by a lowers an octave). If this property is set on the Movie, an alternative algorithm is used, which al without changing the pitch. As this is more computationally expensive, this property may be si on some slow CPUs.
QTMovieSelection-Attribute	The selection range of a QTMovie object; the value for this key is of type NSValue, interprete QTTimeRange structure.
QTMovieTimeScale-Attribute	The movie time scale; the value for this key is of type NSNumber, interpreted as a long. In Mac later, this attribute is gettable and settable. In general, you should set this attribute only on n movies or on movies that have not been edited. Also, you should only increase the time scale should try to use integer multiples of the existing time scale. In earlier versions of Mac OS X, t gettable only.
QTMovieURLAttribute	The URL of a QTMovie object; the value for this key is of type NSURL.
QTMovieVolume-Attribute	The movie volume; the value for this key is of type NSNumber, interpreted as a float.

The following constants specify items in dictionaries passed to QTMovie notifications and delegate methods.

Constant	Description
QTMovieMessageNotification-Parameter	Used as a key in the userInfo dictionary passed to the QTMovieMessageNotification notification to indicate the message. The associated value is an NSString.
QTMovieRateDid-ChangeNotificationParameter	Used as a key in the userInfo dictionary passed to the QTMovieRateDidChangeNotification notification to indicate the new playback rate. The associated value is an NSNumber that holds a float.
QTMovieStatusFlags-NotificationParameter	Used as a key in the userInfo dictionary passed to the QTMovieStatusStringPostedNotification notification to indicate status flags. The associated value is an NSNumber that holds a long.
QTMovieStatusCode-NotificationParameter	Used as a key in the userInfo dictionary passed to the QTMovieStatusStringPostedNotification notification to indicate a status code (or error code). The associated value is an NSNumber that holds an int.

Constant	Description
`QTMovieStatusString-NotificationParameter`	Used as a key in the userInfo dictionary passed to the `QTMovieStatusStringPostedNotification` notification to indicate a status string.
`QTMovieTargetIDNotification-Parameter`	Used as a key in the dictionary passed to the `externalMovie:` delegate method to indicate that the delegate should return a QTMovie object that has the movie ID specified by the key's value.
`QTMovieTargetName-NotificationParameter`	Used as a key in the dictionary passed to the `externalMovie:` delegate method to indicate that the delegate should return a QTMovie object that has the movie name specified by the key's value.

The following constants are dictionary keys that you can use to specify movie attributes, using the `writeToFile` method.

Constant	Description
`QTMovieExport`	The movie export setting; the value for this key is of type `NSNumber`, interpreted as a `BOOL`.
`QTMovieExportType`	The movie export type; the value for this key is of type `NSNumber`, interpreted as a `long`.
`QTMovieFlatten`	The movie flatten setting; the value for this key is of type `NSNumber`, interpreted as a `BOOL`.
`QTMovieExportSettings`	Information to come.
`QTMovieExportManufacturer`	The export manufacturer value; the value for this key is of type `NSNumber`, interpreted as a `long`.

The following constants are dictionary keys that you can use to specify movie attributes, using the `addImage` method.

Constant	Description
`QTAddImageCodecType`	The image codec string; the value for this key is of type `NSString`.
`QTAddImageCodecQuality`	The image codec value; the value for this key is of type `NSNumber`.

The following is a dictionary of attributes can contain these keys, using the `frameImageAtTime:withAttributes:error:` method.

Constant	Description
`QTMovieFrameImageSize`	Size of the image. Value is an NSValue containing an NSSize record. The default image size is the current movie size.
`QTMovieFrameImageType`	Type of the image. Value is an NSString. The default image type is NSImage.

Constant	Description
`QTMovieFrameImage-RepresentationsType`	For NSImage, the image representations in the image. Value is an NSArray of NSString; strings are, for example, NSBitmapImageRep class description. The default is NSBitmapImageRep.
`QTMovieFrameImage-OpenGLContext`	For CVOpenGLTextureRef, the OpenGL context to use. Value is an NSValue (CGLContextObj).
`QTMovieFrameImagePixelFormat`	For CVOpenGLTextureRef, the pixel format to use. Value is an NSValue (CGLPixelFormatObj).
`QTMovieFrameImageInterlaced`	Image is interlaced. Value is an NSNumber (BOOL) (default = NO).
`QTMovieFrameImageHighQuality`	Image is high quality. Value is an NSNumber (BOOL) (default = YES).
`QTMovieFrameImageSingleField`	Image is single field. Value is an NSNumber (BOOL) (default = YES). The returned object is an autorelease object.

The following constants are data locators that you can use to specify movie attributes, using the `movieWithAttributes` and `initWithAttributes` methods.

Constant	Description
`QTMovieDataReferenceAttribute`	The data reference of a QTMovie object.
`QTMoviePasteboardAttribute`	The pasteboard setting of a QTMovie object.
`QTMovieDataAttribute`	The data of a QTMovie object.

The following constants are movie instantiation options that you can use to specify movie attributes, using the `movieWithAttributes` and `initWithAttributes` methods.

Constant	Description
`QTMovieFileOffsetAttribute`	The file offset value; the value for this key is of type `NSNumber`, interpreted as a `long long`.
`QTMovieResolveDataRefAttribute`	The resolved data reference setting; the value for this key is of type `NSNumber`, interpreted as a `BOOL`.
`QTMovieAskUnresolved-DataRefAttribute`	The unresolved data reference setting; the value for this key is of type `NSNumber`, interpreted as a `BOOL`.
`QTMovieOpenAsyncOKAttribute`	The open async setting; the value for this key is of type `NSNumber`, interpreted as a `BOOL`.

These constants allow applications to get information about a movie and its chapters, and to navigate within a movie by chapters. Since chapters are a reasonably common feature of movies and podcasts, QTKit enables developers to create them.

Constant	Description
`QTMovieChapterName`	A key indicating the chapter name in the dictionaries that are array elements in the array returned by `QTMovie chapters` or passed to QTMovie `addChapters: withAttributes:error`.
`QTMovieChapterStartTime`	Aey indicating the chapter start time in the dictionaries that are array elements in the array returned by `QTMovie chapters` or passed to `QTMovie addChapters: withAttributes:error`.
`QTMovieChapterTarget-TrackAttribute`	A key indicating the track in the `QTMovie object` that is the target of the chapter track.

Notifications

QTMovieApertureModeDidChangeNotification

Issued when the aperture mode of the target QTMovie object changes.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTMovie.h`

QTMovieChapterDidChangeNotification

Issued when the chapter associated with QTMovie changes.

This notification contains no information in the userInfo dictionary.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTMovie.h`

QTMovieChapterListDidChangeNotification

Issued when the chapter list associated with QTMovie changes.

This notification contains no information in the userInfo dictionary.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTMovie.h`

QTMovieCloseWindowRequestNotification

Sent when a request is made to close the movie's window.

This notification contains no information in the userInfo dictionary.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTMovie.h`

QTMovieDidEndNotification

Sent when the movie is "done" or at its end.

This notification contains no userInfo parameters. It is equivalent to the standard player controller's `mcActionMovieFinished` action.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTMovie.h`

QTMovieEditabilityDidChangeNotification

Sent when the editable state of a movie has changed.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTMovie.h`

QTMovieEditedNotification

Sent when a movie has been edited.

This notification contains no userInfo dictionary.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTMovie.h`

QTMovieEnterFullScreenRequestNotification

Sent when a request is made to play back a movie in full screen mode.

This notification contains no information in the userInfo dictionary.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTMovie.h`

QTMovieExitFullScreenRequestNotification

Sent when a request is made to play back a movie in normal windowed mode.

This notification contains no information in the userInfo dictionary.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTMovie.h`

QTMovieLoadStateDidChangeNotification

Sent when the load state of a movie has changed.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTMovie.h`

QTMovieLoopModeDidChangeNotification

Sent when a change is made in a movie's looping mode.

This notification contains no information in the userInfo dictionary.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTMovie.h`

QTMovieMessageStringPostedNotification

Sent when a movie message has been received by the movie controller.

Movie messages can be sent to an application by wired actions (for instance, a wired sprite) or by code that issues the `mcActionShowMessageString` movie controller action. The userInfo dictionary contains a single entry whose value is of type `NSString`, which is the movie message.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTMovie.h`

QTMovieRateDidChangeNotification

Sent when the rate of a movie has changed.

The userInfo dictionary contains a single entry whose value is of type `NSNumber` that represents a `float`, which is the new rate.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTMovie.h`

QTMovieSelectionDidChangeNotification

Sent when the selection of a movie has changed.

This notification contains no userInfo dictionary.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTMovie.h`

QTMovieSizeDidChangeNotification

Sent when the size of a movie has changed.

This notification contains no userInfo dictionary.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTMovie.h`

QTMovieStatusStringPostedNotification

Status messages can be sent by QuickTime's streaming components or by any code that wants to display a message in the movie controller bar status area.

The userInfo dictionary contains a single entry whose value is of type `NSString`, which is the status message.

The following are keys (notification parameters) for userInfo items for the `QTMovieStatusStringPostedNotification` notification `QTMovieStatusCodeNotificationParameter` and `QTMovieStatusStringNotificationParameter`.

A status string notification can indicate an error (in which case `QTMovieStatusCodeNotificationParameter` will have a value), or it can contain a string (in which case `QTMovieStatusStringNotificationParameter` will have a value). For more information, see `mcActionShowStatusString`.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTMovie.h`

QTMovieTimeDidChangeNotification

Sent when the time in a movie has changed.

The `QTMovieTimeDidChangeNotification` is fired whenever the movie time changes to a time other than what it would be during normal playback. So, for example, this notification is not fired every frame.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTMovie.h`

QTMovieVolumeDidChangeNotification

Sent when the volume of a movie has changed.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTMovie.h`

QTMovieLayer Reference

Inherits from	CALayer : NSObject
Conforms to	NSCoding (CALayer) CAMediaTiming (CALayer) NSObject (NSObject)
Framework	Library/Frameworks/QTKit.framework
Declared in	QTKit/QTMovieLayer.h
Availability	Available in QuickTime 7.2.1 and later.
Related sample code	CALayerEssentials Core Animation QuickTime Layer

Overview

This class provides a layer into which the frames of a `QTMovie` can be drawn, and is intended to provide support for Core Animation, that is, drawing the contents of a movie into a layer. Note that this class requires rendering using visual contexts.

Tasks

Creating Movie Layers

`+ layerWithMovie:` (page 194)
Creates an autoreleased `QTMovieLayer` associated with the specified `QTMovie` object.

`- initWithMovie:` (page 194)
Creates a `QTMovieLayer` associated with the specified `QTMovie` object.

`- movie` (page 194)
Returns the movie associated with a `QTMovieLayer` object.

Class Methods

layerWithMovie:

Creates an autoreleased `QTMovieLayer` associated with the specified `QTMovie` object.

```
+ (id)layerWithMovie:(QTMovie *)movie
```

Discussion
By default, the movie starts playing immediately at rate 1.0 from the beginning of the movie. These default characteristics can be modified by setting layer properties or movie properties.

Availability
Mac OS X v10.5 and later.

Related Sample Code
CALayerEssentials
Core Animation QuickTime Layer

Declared In
`QTMovieLayer.h`

Instance Methods

initWithMovie:

Creates a `QTMovieLayer` associated with the specified `QTMovie` object.

```
- (id)initWithMovie:(QTMovie *)movie
```

Discussion
By default, the movie starts playing immediately at rate 1.0 from the beginning of the movie. These default characteristics can be modified by setting layer properties or movie properties.

Availability
Mac OS X v10.5 and later.

Declared In
`QTMovieLayer.h`

movie

Returns the movie associated with a `QTMovieLayer` object.

```
- (QTMovie *)movie
```

Availability
Mac OS X v10.5 and later.

Declared In
`QTMovieLayer.h`

QTMovieView Class Reference

Inherits from	NSView : NSResponder : NSObject
Conforms to	NSTextInput NSUserInterfaceValidations NSCoding NSAnimatablePropertyContainer (NSView) NSCoding (NSResponder) NSObject (NSObject)
Framework	/System/Library/Frameworks/QTKit.framework
Declared in	QTKit/QTMovieView.h
Availability	Available in Mac OS X v10.4 and later.
Related sample code	QTAudioExtractionPanel QTKitCreateMovie QTKitMovieShuffler QTKitPlayer QTKitTimeCode

Overview

A `QTMovieView` is a subclass of `NSView` that can be used to display and control QuickTime movies. You normally use a QTMovieView in combination with a QTMovie object, which supplies the movie being displayed. A QTMovieView also supports editing operations on the movie.

The movie can be placed within an arbitrary bounding rectangle in the view's coordinate system, and the remainder of the view can be filled with a fill color. The movie controller, if it is visible, can also be placed within an arbitrary bounding rectangle in the view's coordinate system.

Adopted Protocols

NSMenuValidations

```
- validateMenuItem:
```

NSUserInterfaceValidations

```
- validateUserInterfaceItem
```

Tasks

Initializing the View

- `initWithFrame:` (page 204)

Getting View Characteristics

- `movie` (page 206)
- `isControllerVisible` (page 205)
- `isEditable` (page 205)
- `preservesAspectRatio` (page 208)
- `fillColor` (page 203)
- `movieBounds` (page 206)
- `movieControllerBounds` (page 207)
- `controllerBarHeight` (page 201)

Setting View Characteristics

- `setMovie:` (page 210)
- `setControllerVisible:` (page 209)
- `setPreservesAspectRatio:` (page 211)
- `setShowsResizeIndicator:` (page 211)
- `setFillColor:` (page 210)
- `setEditable:` (page 210)
- `selectNone:` (page 209)

Controlling Movie Playback

- `play:` (page 208)
- `pause:` (page 207)
- `gotoBeginning:` (page 203)
- `gotoEnd:` (page 203)
- `gotoNextSelectionPoint:` (page 203)
- `gotoPreviousSelectionPoint:` (page 204)
- `gotoPosterFrame:` (page 204)
- `stepForward:` (page 213)
- `stepBackward:` (page 212)

Editing a Movie

- `cut:` (page 202)
- `copy:` (page 202)
- `paste:` (page 207)
- `selectAll:` (page 208)
- `delete:` (page 202)
- `add:` (page 200)
- `addScaled:` (page 201)
- `replace:` (page 208)
- `trim:` (page 213)

Showing and Hiding Buttons in the Movie Controller Bar

- `setBackButtonVisible:` (page 209)
  Sets the specified controller bar button to be visible or invisible, according to the state parameter.
- `setCustomButtonVisible:` (page 209)
  Sets the specified controller bar button to be visible or invisible, according to the state parameter.
- `setHotSpotButtonVisible:` (page 210)
  Sets the specified controller bar button to be visible or invisible, according to the state parameter.
- `setStepButtonsVisible:` (page 211)
  Sets the specified controller bar button to be visible or invisible, according to the state parameter.
- `setTranslateButtonVisible:` (page 212)
  Sets the specified controller bar button to be visible or invisible, according to the state parameter.
- `setVolumeButtonVisible:` (page 212)
  Sets the specified controller bar button to be visible or invisible, according to the state parameter.
- `setZoomButtonsVisible:` (page 212)
  Sets the specified controller bar button to be visible or invisible, according to the state parameter.
- `isBackButtonVisible` (page 204)
  Returns the current visibility state of the specified controller bar button.
- `isCustomButtonVisible` (page 205)
  Returns the current visibility state of the specified controller bar button.
- `isHotSpotButtonVisible` (page 205)
  Returns the current visibility state of the specified controller bar button.
- `areStepButtonsVisible` (page 201)
  Returns the current visibility state of the specified controller bar button.
- `isTranslateButtonVisible` (page 206)
  Returns the current visibility state of the specified controller bar button.
- `isVolumeButtonVisible` (page 206)
  Returns the current visibility state of the specified controller bar button.
- `areZoomButtonsVisible` (page 201)
  Returns the current visibility state of the specified controller bar button.

Instance Methods

add:

```
- (IBAction)add:(id)sender
```

Discussion
This action method adds the contents of the clipboard to the movie at the current movie time. This action is undoable. If the movie is not editable, this method raises an exception.

Availability
Available in Mac OS X v10.3 and later.

Declared In
QTMovieView.h

addScaled:

```
- (IBAction)addScaled:(id)sender
```

Discussion
This action method adds the contents of the clipboard to the movie, scaled to fit into the current movie selection. This action is undoable. If the movie is not editable, this method raises an exception.

Availability
Available in Mac OS X v10.3 and later.

Declared In
QTMovieView.h

areStepButtonsVisible

Returns the current visibility state of the specified controller bar button.

```
- (BOOL)areStepButtonsVisible
```

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

Declared In
QTMovieView.h

areZoomButtonsVisible

Returns the current visibility state of the specified controller bar button.

```
- (BOOL)areZoomButtonsVisible
```

Discussion
These methods allow applications to hide and show specific buttons in the movie controller bar.

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

Declared In
QTMovieView.h

controllerBarHeight

```
- (float)controllerBarHeight
```

Discussion
Returns the height of the controller bar.

Availability
Available in Mac OS X v10.3 and later.

Declared In
QTMovieView.h

copy:

```
- (IBAction)copy:(id)sender
```

Discussion
This action method copies the current movie selection onto the clipboard. If there is no selection, the current frame is copied. The movie does not need to be editable.

Availability
Available in Mac OS X v10.3 and later.

Declared In
QTMovieView.h

cut:

```
- (IBAction)cut:(id)sender
```

Discussion
This action method deletes the current movie selection from the movie, placing it on the clipboard. If there is no selection, the current frame is deleted. This action is undoable. If the movie is not editable, this method raises an exception.

Availability
Available in Mac OS X v10.3 and later.

Declared In
QTMovieView.h

delete:

```
- (IBAction)delete:(id)sender
```

Discussion
This action method deletes the current movie selection from the movie, placing it on the clipboard. If there is no selection, the current frame is deleted. This action is undoable. If the movie is not editable, this method raises an exception.

Availability
Available in Mac OS X v10.3 and later.

Declared In
QTMovieView.h

fillColor

```
- (NSColor *)fillColor
```

Discussion
Returns the fill color of the QTMovieView.

Availability
Available in Mac OS X v10.3 and later.

Declared In
QTMovieView.h

gotoBeginning:

```
- (IBAction)gotoBeginning:(id)sender
```

Discussion
This action method sets the current movie time to the beginning of the movie. If the movie is playing, the movie continues playing from the new position.

Availability
Available in Mac OS X v10.3 and later.

Declared In
QTMovieView.h

gotoEnd:

```
- (IBAction)gotoEnd:(id)sender
```

Discussion
This action method sets the current movie time to the end of the movie. If the movie is playing in one of the looping modes, the movie continues playing accordingly; otherwise, play stops.

Availability
Available in Mac OS X v10.3 and later.

Declared In
QTMovieView.h

gotoNextSelectionPoint:

```
- (IBAction)gotoNextSelectionPoint:(id)sender
```

Discussion
This action method sets the current movie time to the next selection point.

Availability
Available in Mac OS X v10.3 and later.

Declared In
QTMovieView.h

gotoPosterFrame:

```
- (IBAction)gotoPosterFrame:(id)sender
```

Discussion
This action method sets the current movie time to the movie poster frame.

Availability
Available in Mac OS X v10.3 and later.

Declared In
QTMovieView.h

gotoPreviousSelectionPoint:

```
- (IBAction)gotoPreviousSelectionPoint:(id)sender
```

Discussion
This action method sets the current movie time to the previous selection point.

Availability
Available in Mac OS X v10.3 and later.

Declared In
QTMovieView.h

initWithFrame:

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

Discussion
Initializes a newly allocated QTMovieView with *frame* as its frame rectangle. The new movie view object must be inserted into the view hierarchy of an NSWindow before it can be used. This method is the designated initializer for the `QTMovieView` class.

Availability
Available in Mac OS X v10.3 and later.

Declared In
QTMovieView.h

isBackButtonVisible

Returns the current visibility state of the specified controller bar button.

```
- (BOOL)isBackButtonVisible
```

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

Declared In
`QTMovieView.h`

isControllerVisible

```
- (BOOL)isControllerVisible
```

Discussion
Returns `YES` if the movie controller bar of the QTMovieView object is visible. The default is `YES`.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTMovieView.h`

isCustomButtonVisible

Returns the current visibility state of the specified controller bar button.

```
- (BOOL)isCustomButtonVisible
```

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

Declared In
`QTMovieView.h`

isEditable

```
- (BOOL)isEditable
```

Discussion
Returns `YES` if the QTMovieView object is editable. When editable, a movie can be modified using editing methods and associated key commands. The default is `NO`.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTMovieView.h`

isHotSpotButtonVisible

Returns the current visibility state of the specified controller bar button.

```
- (BOOL)isHotSpotButtonVisible
```

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

Declared In
QTMovieView.h

isTranslateButtonVisible

Returns the current visibility state of the specified controller bar button.

```
- (BOOL)isTranslateButtonVisible
```

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

Declared In
QTMovieView.h

isVolumeButtonVisible

Returns the current visibility state of the specified controller bar button.

```
- (BOOL)isVolumeButtonVisible
```

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

Declared In
QTMovieView.h

movie

```
- (QTMovie *)movie
```

Discussion
Returns the QTMovie object associated with the QTMovieView.

Availability
Available in Mac OS X v10.3 and later.

Related Sample Code
QTKitTimeCode

Declared In
QTMovieView.h

movieBounds

```
- (NSRect)movieBounds
```

Discussion
Returns the rectangle currently occupied by the movie in a QTMovieView. This rectangle does not include the area occupied by the movie controller bar (if it's visible).

Availability
Available in Mac OS X v10.3 and later.

Declared In
QTMovieView.h

movieControllerBounds

```
- (NSRect)movieControllerBounds
```

Discussion
Returns the rectangle currently occupied by the movie controller bar (if it's visible) in a QTMovieView.

Availability
Available in Mac OS X v10.3 and later.

Declared In
QTMovieView.h

paste:

```
- (IBAction)paste:(id)sender
```

Discussion
This action method inserts the contents of the clipboard (if it contains a movie clip) into the movie at the current play position. This action is undoable. If the movie is not editable, this method raises an exception.

Availability
Available in Mac OS X v10.3 and later.

Declared In
QTMovieView.h

pause:

```
- (IBAction)pause:(id)sender
```

Discussion
This action method pauses the movie playback. This method does nothing if the movie is already paused.

Availability
Available in Mac OS X v10.3 and later.

Related Sample Code
MyMovieFilter

Declared In
QTMovieView.h

play:

```
- (IBAction)play:(id)sender
```

Discussion
This action method starts the movie playing at its current location. This method does nothing if the movie is already playing.

Availability
Available in Mac OS X v10.3 and later.

Related Sample Code
MyMovieFilter

Declared In
`QTMovieView.h`

preservesAspectRatio

```
- (BOOL)preservesAspectRatio
```

Discussion
Returns `YES` if the QTMovieView object maintains the aspect ratio of the movie when drawing it in the view. The remainder is filled with `fillColor`.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTMovieView.h`

replace:

```
- (IBAction)replace:(id)sender
```

Discussion
This action method replaces the current movie selection with the contents of the clipboard. If there is no selection, the contents of the clipboard replace the entire movie. This action is undoable. If the movie is not editable, this method raises an exception.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTMovieView.h`

selectAll:

```
- (IBAction)selectAll:(id)sender
```

Discussion
This action method selects the entire movie.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTMovieView.h`

selectNone:

```
- (IBAction)selectNone:(id)sender
```

Discussion
This action method selects nothing. Note that it does not change the movie time.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTMovieView.h`

setBackButtonVisible:

Sets the specified controller bar button to be visible or invisible, according to the state parameter.

```
- (void)setBackButtonVisible:(BOOL)state
```

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

Declared In
`QTMovieView.h`

setControllerVisible:

```
- (void)setControllerVisible:(BOOL)controllerVisible
```

Discussion
Sets the visibility state of the movie controller bar in a QTMovieView to *controllerVisible*.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTMovieView.h`

setCustomButtonVisible:

Sets the specified controller bar button to be visible or invisible, according to the state parameter.

```
- (void)setCustomButtonVisible:(BOOL)state
```

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

Declared In
QTMovieView.h

setEditable:

```
- (void)setEditable:(BOOL)editable
```

Discussion
Sets the edit state of a QTMovieView to *editable*. The default state is NO.

Availability
Available in Mac OS X v10.3 and later.

Declared In
QTMovieView.h

setFillColor:

```
- (void)setFillColor:(NSColor *)fillColor
```

Discussion
Sets the fill color of a QTMovieView to *fillColor*. Note that this may cause a redraw.

Availability
Available in Mac OS X v10.3 and later.

Declared In
QTMovieView.h

setHotSpotButtonVisible:

Sets the specified controller bar button to be visible or invisible, according to the state parameter.

```
- (void)setHotSpotButtonVisible:(BOOL)state
```

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

Declared In
QTMovieView.h

setMovie:

```
- (void)setMovie:(QTMovie *)movie
```

Discussion
Sets the QTMovie object in a QTMovieView to *movie*. The currently set QuickTime movie is disposed of using `DisposeMovie`, unless the QTMovie was created with a call to `initWithQuickTimeMovie` and the `disposeWhenDone` flag was `NO`.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTMovieView.h`

setPreservesAspectRatio:

```
- (void)setPreservesAspectRatio:(BOOL)preservesAspectRatio
```

Discussion
Sets the aspect ratio state of a QTMovieView to *preservesAspectRatio*. If *preservesAspectRatio* is `YES`, the longer side of the movie rectangle is scaled to exactly fit into the view's frame and the other side is centered in the view frame; the remaining area is filled with the view's fill color. Note that the movie view may be redrawn, but not resized.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTMovieView.h`

setShowsResizeIndicator:

```
- (void)setShowsResizeIndicator:(BOOL)show
```

Discussion
Shows or hides the movie controller grow box.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTMovieView.h`

setStepButtonsVisible:

Sets the specified controller bar button to be visible or invisible, according to the state parameter.

```
- (void)setStepButtonsVisible:(BOOL)state
```

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

Declared In
`QTMovieView.h`

setTranslateButtonVisible:

Sets the specified controller bar button to be visible or invisible, according to the state parameter.

```
- (void)setTranslateButtonVisible:(BOOL)state
```

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

Declared In
`QTMovieView.h`

setVolumeButtonVisible:

Sets the specified controller bar button to be visible or invisible, according to the state parameter.

```
- (void)setVolumeButtonVisible:(BOOL)state
```

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

Declared In
`QTMovieView.h`

setZoomButtonsVisible:

Sets the specified controller bar button to be visible or invisible, according to the state parameter.

```
- (void)setZoomButtonsVisible:(BOOL)state
```

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

Declared In
`QTMovieView.h`

stepBackward:

```
- (IBAction)stepBackward:(id)sender
```

Discussion
This action method steps the movie backward one frame.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTMovieView.h`

stepForward:

```
- (IBAction)stepForward:(id)sender
```

Discussion
This action method steps the movie forward one frame.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTMovieView.h`

trim:

```
- (IBAction)trim:(id)sender
```

Discussion
This action method trims the movie to the current movie selection. If there is no selection, the current frame is retained and the remainder of the movie is deleted. This action is undoable. If the movie is not editable, this method raises an exception.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTMovieView.h`

QTSampleBuffer Reference

Inherits from	NSObject
Conforms to	NSObject (NSObject)
Framework	Library/Frameworks/QTKit.framework
Declared in	QTKit/QTSampleBuffer.h
Availability	Available in QuickTime 7.2.1 and later.
Related sample code	QTRecorder

Overview

This class provides format information, timing information, and metadata on media sample buffers. `QTSampleBuffer` objects contain data from media samples as well as metadata about those samples, including format information, timing information, and other attributes. Some extended information can be accessed via a QTSampleBuffer's `attributeForKey:` and `sampleBufferAttributes` methods, using the keys described in the Constants section. In addition to these explicit methods, applications can use key-value coding to get extended attributes. For an object that supports a given attribute, `valueForKey:` will be functionally identical to `attributeForKey:`. Applications wishing to observe changes for a given attribute can add a key-value observer where the key path is the attribute key.

Tasks

Getting Sample Buffer Information

- `attributeForKey:` (page 216)
  Returns a sample buffer attribute for the given key.
- `audioBufferListWithOptions:` (page 217)
  Returns a pointer to a Core Audio `AudioBufferList` containing audio data owned by the receiver.
- `bytesForAllSamples` (page 217)
  Returns a pointer to the bytes of media data contained in the sample buffer.
- `decodeTime` (page 218)
  Returns the decode time of the buffer.

- `decrementSampleUseCount` (page 218)
  Decrements the use count of the sample data owned by the receiver, allowing the sample data to be invalidated after a matching call to `incrementSampleUseCount`.
- `duration` (page 219)
  Returns the duration of the buffer.
- `formatDescription` (page 219)
  Returns the format description of the buffer.
- `getAudioStreamPacketDescriptions:inRange:` (page 219)
  Gets an array of Core Audio AudioStreamPacketDescriptions describing the lengths of samples in variable bit- rate audio buffers.
- `incrementSampleUseCount` (page 220)
  Increments the use count of the sample data owned by the receiver, preventing the sample data from being invalidated until a matching call to `decrementSampleUseCount`.
- `lengthForAllSamples` (page 220)
  Returns the length of the buffer returned by `bytesForAllSamples`.
- `numberOfSamples` (page 221)
  Returns the number of media samples contained in the buffer.
- `presentationTime` (page 221)
  Returns the presentation time of the buffer.
- `sampleBufferAttributes` (page 221)
  Returns a dictionary of the sample buffer's current attirbutes.
- `sampleUseCount` (page 222)
  Returns the use count of the sample data owned by the receiver.

Instance Methods

attributeForKey:

Returns a sample buffer attribute for the given key.

```
- (id)attributeForKey:(NSString *)attributeKey
```

Parameters

attributeKey
  The key of the returned attribute. Attribute keys are described in the Constants section.

Return Value
An object for the given attribute key, or `NIL` if the sample buffer does not have the given attribute.

Discussion
Use this method to get attributes of a sample buffer. The keys that can be used with this method are described in the Constants section. Applications using key-value coding can also get an attribute for a given key by passing that key to the NSObject `valueForKey:` method.

Availability
Mac OS X v10.5 and later.

Declared In
`QTSampleBuffer.h`

audioBufferListWithOptions:

Returns a pointer to a Core Audio `AudioBufferList` containing audio data owned by the receiver.

```
- (AudioBufferList
    *)audioBufferListWithOptions:(QTSampleBufferAudioBufferListOptions)options;
```

Parameters

`options`

A bitfield containing options that determine what kind of audio buffer list will be returned. The options constants, which can be combined using the bitwise or operator, are described as part of the `QTSampleBufferAudioBufferListOptions` type.

Return Value
A pointer to an `AudioBufferList` structure. This pointer and its associated audio buffers will remain valid as long as the receiver is valid and the value returned by `sampleUseCount` is greater then 0.

Discussion
This method returns a pointer to a Core Audio `AudioBufferList` containing all of the audio data in the sample buffer. The AudioBufferList can then be passed to Core Audio APIs for rendering and processing audio. The returned `AudioBufferList` will be valid for as long as the receiver is valid and the value returned by `sampleUseCount` has not been decremented to 0. Clients passing the `AudioBufferList` to an audio unit must include the `QTSampleBufferAudioBufferListOptionAssure16ByteAlignment` flag in the options parameter. This method will throw an NSInternalInconsistencyException if called after `decrementSampleUseCount` has been used to invalidate the media data contained in the sample buffer.

Availability
Mac OS X v10.5 and later.
Not available to 64-bit applications.

Declared In
`QTSampleBuffer.h`

bytesForAllSamples

Returns a pointer to the bytes of media data contained in the sample buffer.

```
- (void *)bytesForAllSamples
```

Return Value
A pointer to a buffer of media data.

Discussion
This method returns a pointer to the data for the media samples contained within the sample buffer. Clients reading bytes from this pointer should check the total length of the buffer using `lengthForAllSamples`. Applications can interpret the media data returned by this method using the infomation from the sample buffer's `formatDescription`. This method will throw an NSInternalInconsistencyException if called after `decrementSampleUseCount` has been used to invalidate the media data contained in the sample buffer.

Availability
Mac OS X v10.5 and later.
Not available to 64-bit applications.

Declared In
`QTSampleBuffer.h`

decodeTime

Returns the decode time of the buffer.

```
- (QTTime)decodeTime
```

Return Value
A `QTTime` representing the decode time of the buffer. For B-frame video media, the decode time may be different from the `presentationTime`.

Availability
Mac OS X v10.5 and later.

Declared In
`QTSampleBuffer.h`

decrementSampleUseCount

Decrements the use count of the sample data owned by the receiver, allowing the sample data to be invalidated after a matching call to `incrementSampleUseCount`.

```
- (void)decrementSampleUseCount
```

Discussion
This method allows clients to control when the potentially large memory buffers owned by the receiver are deallocated. A newly allocated `QTSampleBuffer` has a sample use count of 1. When the sample use count drops to 0, the memory allocated for the samples will be freed and the `bytesForAllSamples`, `lengthForAllSamples`, and`audioBufferListWithOptions:` methods will each throw an NSInternalInconsistencyException when called. This method is analagous to the NSObject release method in that it allows clients to relinquish ownership over data contained within the sample buffer. In particular, clients that have called `incrementSampleUseCount` because they were interested in the sample data of `QTSampleBuffer` objects returned by other APIs in QTKit should call this method when they no longer need that data. It is particularly important that clients using garbage collection ensure that the sample use count is 0 when they no longer require the sample data owned by a `QTSampleBuffer`, so that memory can be deallocated promptly rather than when the object is finalized.

Availability
Mac OS X v10.5 and later.

Declared In
`QTSampleBuffer.h`

duration

Returns the duration of the buffer.

```
- (QTTime)duration
```

Return Value
A `QTTime` representing the duration of the buffer.

Availability
Mac OS X v10.5 and later.

Declared In
`QTSampleBuffer.h`

formatDescription

Returns the format description of the buffer.

```
- (QTFormatDescription *)formatDescription
```

Return Value
A `QTFormatDescription` object describing the media format of the buffer.

Availability
Mac OS X v10.5 and later.

Declared In
`QTSampleBuffer.h`

getAudioStreamPacketDescriptions:inRange:

Gets an array of Core Audio AudioStreamPacketDescriptions describing the lengths of samples in variable bit- rate audio buffers.

```
- (BOOL)getAudioStreamPacketDescriptions:(void *)audioStreamPacketDescriptions
    inRange:(NSRange)range
```

Parameters

audioStreamPacketDescriptions
: An array of Core Audio AudioStreamPacketDescription structures allocated to be large enough to fit the number of packet descriptions indicated by range.

range
: The range of packet descriptions to use when filling the array. If the range falls outside the number of samples returned by `numberOfSamples`, this method raises an NSRangeException.

Return Value
If the buffer contains variable bit-rate audio, this method fills the audioStreamPacketDescriptions with AudioStreamPacketDescription structures and returns YES. If the buffer contains single bit-rate audio, this method returns NO and leaves audioStreamPacketDescriptions untouched.

Discussion
Applications that need to process individual packets of variable bit-rate audio from the buffer should call this method to determine the length of each sample in the buffer. This method raises an NSInternalInconsistencyException if this method is invoked on a `QTSampleBuffer` object that does not describe an audio sample buffer.

Availability
Mac OS X v10.5 and later.

Declared In
`QTSampleBuffer.h`

incrementSampleUseCount

Increments the use count of the sample data owned by the receiver, preventing the sample data from being invalidated until a matching call to `decrementSampleUseCount`.

```
- (void)incrementSampleUseCount
```

Discussion
This method allows clients to control when the potentially large memory buffers owned by the receiver are deallocated. A newly allocated `QTSampleBuffer` has a sample use count of 1. When the sample use count drops to 0, the memory allocated for the samples will be freed and the `bytesForAllSamples`, `lengthForAllSamples`, and `audioBufferListWithOptions:` methods will each throw an NSInternalInconsistencyException when called. This method is analogous to the NSObject retain method in that it allows clients to declare ownership over data contained within the sample buffer. In particular, clients interested in the sample data of `QTSampleBuffer` objects returned by other APIs in QTKit should call this method to ensure that they have acceess to the sample data, and later call `decrementSampleUseCount` when they no longer need that data. It is particularly important that clients using garbage collection ensure that the sample use count is 0 when they no longer require the sample data owned by a `QTSampleBuffer`, so that memory can be deallocated promptly rather than when the object is finalized.

Availability
Mac OS X v10.5 and later.

Declared In
`QTSampleBuffer.h`

lengthForAllSamples

Returns the length of the buffer returned by `bytesForAllSamples`.

```
- (NSUInteger)lengthForAllSamples
```

Return Value
The length, in bytes of the buffer returned by `bytesForAllSamples`.

Discussion
Clients reading bytes from the pointer returned by `bytesForAllSamples` should use this method to check the total length of the buffer. This method will throw an NSInternalInconsistencyException if called after `decrementSampleUseCount` has been used to invalidate the media data contained in the sample buffer.

Availability
Mac OS X v10.5 and later.
Not available to 64-bit applications.

Declared In
`QTSampleBuffer.h`

numberOfSamples

Returns the number of media samples contained in the buffer.

```
- (NSInteger)numberOfSamples
```

Return Value
The number of samples in the buffer.

Discussion
In general, video buffers will always contain one sample (a single frame), while audio buffers may contain multiple samples. Applications that need to interpret variable bit-rate audio can get the individual sample lengths with the `getAudioStreamPacketDescriptions:inRange:` method.

Availability
Mac OS X v10.5 and later.

Declared In
`QTSampleBuffer.h`

presentationTime

Returns the presentation time of the buffer.

```
- (QTTime)presentationTime
```

Return Value
A `QTTime` representing the presentation time of the buffer. For B-frame video media, the presentation time may be different from the `decodeTime`.

Availability
Mac OS X v10.5 and later.

Declared In
`QTSampleBuffer.h`

sampleBufferAttributes

Returns a dictionary of the sample buffer's current attirbutes.

```
- (NSDictionary *)sampleBufferAttributes
```

Return Value
A dictionary of attributes attached to the sample buffer. Attribute keys are described in the Constants section that discusses the attributes.

Discussion
Applications can use this method to determine what attributes a specific sample buffer supports.

Availability
Mac OS X v10.5 and later.

Declared In
`QTSampleBuffer.h`

sampleUseCount

Returns the use count of the sample data owned by the receiver.

```
- (NSUInteger)sampleUseCount
```

Return Value
The use count of the sample data owned by the receiver.

Discussion
This method returns the use count of the data owned by the reciever, as determined buy the number of invocations of `incrementSampleUseCount` and `decrementSampleUseCount`. If the value retunred by this method is 0, then the data owned by the reciever has been invalidated and the `bytesForAllSamples`, `lengthForAllSamples`, and `audioBufferListWithOptions:` methods wil throw an NSInternalInconsistencyException. Clients should rarely need to call this method. It is generally only useful for debugging purposes.

Availability
Mac OS X v10.5 and later.

Declared In
`QTSampleBuffer.h`

Constants

QTSampleBufferHostTimeAttribute

If the buffer is from a real time source, this attribute returns the buffer's host time.

```
QTSampleBufferHostTimeAttribute
@"hostTime"
```

Constants
`QTSampleBufferHostTimeAttribute`
The value returned by this attribute can be compared with the return value of `CVGetCurrentHostTime()` or `AudioGetCurrentHostTime()` to determine whether or not it is too late for the buffer to be processed in real time. Value is an NSNumber interpreted as a UInt64. This string value can be used in key paths for key-value coding, key-value observing, and bindings.

Declared In
`QTSampleBuffer.h`

QTSampleBufferSMPTETimeAttribute

Returns the SMPTE timecode fo the sample buuffer, if it has one.

```
QTSampleBufferSMPTETimeAttribute
@"SMPTETime"
```

Constants

`QTSampleBufferSMPTETimeAttribute`

The value is an `NSValue` interpreted as a `SMPTETime` (defined in `CoreAudio/CoreAudioTypes.h`). This string value can be used in key paths for key-value coding, key-value observing, and bindings.

Declared In

`QTSampleBuffer.h`

QTSampleBufferSceneChangeTypeAttribute

If the buffer marks a scene change in the input content, returns a constant.

```
QTSampleBufferSceneChangeTypeAttribute
@"sceneChangeType"
```

Constants

`QTSampleBufferSceneChangeTypeAttribute`

The returned constant is described in Scene Change Types specifying the type of scene change. This string value can be used in key paths for key-value coding, key-value observing, and bindings.

Declared In

`QTSampleBuffer.h`

QTSampleBufferDateRecordedAttribute

Returns the date on which the media in the buffer was originally recorded.

```
QTSampleBufferDateRecordedAttribute
@"dateRecorded"
```

Constants

`QTSampleBufferDateRecordedAttribute`

The value is an NSDate. This string value can be used in key paths for key-value coding, key-value observing, and bindings.

Declared In

`QTSampleBuffer.h`

QTSampleBufferExplicitSceneChange

Indicates that a scene change was explicitly marked in the sample buffer's metadata.

```
QTSampleBufferExplicitSceneChange
```

Constants

`QTSampleBufferExplicitSceneChange`

This constant is returned by `QTSampleBufferSceneChangeTypeAttribute` specifying what kind of scene change, if any, is marked by a sample buffer.

Declared In
`QTSampleBuffer.h`

QTSampleBufferTimeStampDiscontinuitySceneChange

Indicates that the scene changed due to a discontinuity in time stamps between the current sample buffer and the previous sample buffer.

```
QTSampleBufferTimeStampDiscontinuitySceneChange
```

Constants

`QTSampleBufferTimeStampDiscontinuitySceneChange`

This constant is returned by `QTSampleBufferSceneChangeTypeAttribute` specifying what kind of scene chnage, if any, is marked by a sample buffer.

Declared In
`QTSampleBuffer.h`

QTTrack Class Reference

Inherits from	NSObject
Conforms to	NSObject (NSObject)
Framework	/System/Library/Frameworks/QTKit.framework
Declared in	QTKit/QTTrack.h
Availability	Available in Mac OS X v10.4 and later.
Related sample code	CIVideoDemoGL MoviePlayer - C# QTAudioExtractionPanel QTKitTimeCode QTMetadataEditor

Overview

The QTTrack class represents a QuickTime track (of type `Track`). QTTrack objects are associated with QTMovie objects and support methods for getting and setting the track properties. If necessary, you can retrieve the track identifier associated with a QTTrack object by calling its `quickTimeTrack:` method. Note that a movie can have multiple tracks. A track has a single media.

Tasks

Creating a QTTrack

`+ trackWithQuickTimeTrack:error:` (page 227)
Creates a QTTrack object with data from the QuickTime track *track*.

Initializing a QTTrack

Initializes a newly created QTTrack object with data from the QuickTime track *track*.

`- initWithQuickTimeTrack:error:` (page 229)

Getting Track Properties

- `movie` (page 231)
- `media` (page 231)
- `isEnabled` (page 230)
- `volume` (page 234)
- `attributeForKey:` (page 228)
- `trackAttributes` (page 234)

Setting Track Properties

- `setEnabled:` (page 233)
- `setVolume:` (page 233)
- `setAttribute:forKey:` (page 232)
- `setTrackAttributes:` (page 233)

Editing Track Properties

- `addImage:forDuration:withAttributes:` (page 228)
- `deleteSegment:` (page 229)
- `insertEmptySegmentAt:` (page 230)
- `insertSegmentOfTrack:timeRange:atTime:` (page 230)
- `insertSegmentOfTrack:fromRange:scaledToRange:` (page 230)
- `scaleSegment:newDuration:` (page 232)

Getting QTTrack Primitives

- `quickTimeTrack` (page 231)

Getting and Setting Aperture Mode Dimensions

- `apertureModeDimensionsForMode:` (page 228)
  Returns an NSSize value that indicates the dimensions of the target track for the specified movie aperture mode. For instance, passing a mode of `QTMovieApertureModeClean` would cause `apertureModeDimensionsForMode:` to return the track dimensions to use in clean aperture mode.
- `setApertureModeDimensions:forMode:` (page 232)
  Sets the dimensions of the target track for the specified movie aperture mode.
- `generateApertureModeDimensions` (page 229)
  Adds information to a QTTrack needed to support aperture modes for tracks created with applications and/or versions of QuickTime that did not support aperture mode dimensions. If the image descriptions in the track lack tags describing clean aperture and pixel aspect ratio information, the media data is scanned to see if the correct values can be divined and attached. Then the aperture mode dimensions are calculated and set. Afterwards, the `QTTrackHasApertureModeDimensionsAttribute` property will be set to `YES` for this track. Tracks that do not support aperture modes are not changed.
- `removeApertureModeDimensions` (page 232)
  Removes aperture mode dimension information from the target track. It does not attempt to modify sample descriptions, so it may not completely reverse the effects of `generateApertureModeDimensions`. It sets the `QTTrackHasApertureModeDimensionsAttribute` property to `NO`.

Class Methods

trackWithQuickTimeTrack:error:

Creates a QTTrack object with data from the QuickTime track *track*.

```
+ (id)trackWithQuickTimeTrack:(Track)track error:(NSError **)errorPtr
```

Discussion
If a QTTrack object cannot be created, an NSError object is returned in the location pointed to by *errorPtr*. Pass `NIL` if you do not want an NSError object returned.

Availability
Available in Mac OS X v10.3 and later.
Not available to 64-bit applications.

Declared In
`QTTrack.h`

Instance Methods

addImage:forDuration:withAttributes:

```
- (void)addImage:(NSImage *)image forDuration:(QTTime)duration
    withAttributes:(NSDictionary *)attributes
```

Discussion
Adds an image for the specified duration to the receiver, using attributes specified in the attributes dictionary. Keys in the dictionary can be `QTAddImageCodecType` to select a codec type and `QTAddImageCodecQuality` to select a quality. Qualities are expected to be specified as NSNumbers, using the codec values like `codecNormalQuality`. (See ImageCompression.h for the complete list.)

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTTrack.h`

apertureModeDimensionsForMode:

Returns an NSSize value that indicates the dimensions of the target track for the specified movie aperture mode. For instance, passing a mode of `QTMovieApertureModeClean` would cause `apertureModeDimensionsForMode:` to return the track dimensions to use in clean aperture mode.

```
- (NSSize)apertureModeDimensionsForMode:(NSString *)mode
```

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTTrack.h`

attributeForKey:

```
-(id)attributeForKey:(NSString *)attributeKey
```

Discussion
Returns the current value of the track attribute *`attributeKey`*. A list of supported track attributes and their acceptable values can be found in the "Constants" (page 234) section.

Availability
Available in Mac OS X v10.3 and later.

Related Sample Code
QTKitPlayer
QTMetadataEditor
TrackFormatDemo

Declared In
QTTrack.h

deleteSegment:

```
- (void)deleteSegment:(QTTimeRange)segment
```

Discussion
Deletes from a QTTrack the segment delimited by *segment*. If the track is not editable, this method raises an exception.

Availability
Available in Mac OS X v10.3 and later.

Declared In
QTTrack.h

generateApertureModeDimensions

Adds information to a QTTrack needed to support aperture modes for tracks created with applications and/or versions of QuickTime that did not support aperture mode dimensions. If the image descriptions in the track lack tags describing clean aperture and pixel aspect ratio information, the media data is scanned to see if the correct values can be divined and attached. Then the aperture mode dimensions are calculated and set. Afterwards, the `QTTrackHasApertureModeDimensionsAttribute` property will be set to `YES` for this track. Tracks that do not support aperture modes are not changed.

```
- (void)generateApertureModeDimensions
```

Availability
Available in Mac OS X v10.3 and later.

Declared In
QTTrack.h

initWithQuickTimeTrack:error:

```
- (id)initWithQuickTimeTrack:(Track)track error:(NSError **)errorPtr
```

Discussion
If a QTTrack object cannot be created, an NSError object is returned in the location pointed to by *errorPtr*. Pass `NIL` if you do not want an NSError object returned.

Availability
Available in Mac OS X v10.3 and later.
Not available to 64-bit applications.

Declared In
QTTrack.h

insertEmptySegmentAt:

```
- (void)insertEmptySegmentAt:(QTTimeRange)range
```

Discussion
Inserts into a QTTrack an empty segment delimited by the range *range*. If the track is not editable, this method raises an exception.

Availability
Available in Mac OS X v10.3 and later.

Declared In
QTTrack.h

insertSegmentOfTrack:fromRange:scaledToRange:

```
- (void)insertSegmentOfTrack:(QTTrack *)track fromRange:(QTTimeRange)srcRange
    scaledToRange:(QTTimeRange)dstRange
```

Discussion
Inserts the specified segment from the track into the receiver, scaled to the range *dstRange*. This is essentially an Add Scaled operation on a track. If the track is not editable, this method raises an exception.

Availability
Available in Mac OS X v10.3 and later.

Declared In
QTTrack.h

insertSegmentOfTrack:timeRange:atTime:

```
- (void)insertSegmentOfTrack:(QTTrack *)track timeRange:(QTTimeRange)range
    atTime:(QTTime)time
```

Discussion
Inserts into a QTTrack at time *time* the selection in movie delimited by the time range *range*. If the track is not editable, this method raises an exception.

Availability
Available in Mac OS X v10.3 and later.

Declared In
QTTrack.h

isEnabled

```
- (BOOL)isEnabled
```

Discussion
Returns YES if the QTTrack object is currently enabled, NO otherwise.

Availability
Available in Mac OS X v10.3 and later.

Declared In
QTTrack.h

media

```
- (QTMedia *)media
```

Discussion
Returns the media associated with a QTTrack object.

Availability
Available in Mac OS X v10.3 and later.

Related Sample Code
QTKitTimeCode
QTMetadataEditor

Declared In
QTTrack.h

movie

```
- (QTMovie *)movie
```

Discussion
Returns the movie that contains a QTTrack object.

Availability
Available in Mac OS X v10.3 and later.

Declared In
QTTrack.h

quickTimeTrack

```
-(Track)quickTimeTrack
```

Discussion
Returns the QuickTime track associated with a QTTrack object.

Availability
Available in Mac OS X v10.3 and later.
Not available to 64-bit applications.

Related Sample Code
QTAudioExtractionPanel
QTKitTimeCode

Declared In
QTTrack.h

removeApertureModeDimensions

Removes aperture mode dimension information from the target track. It does not attempt to modify sample descriptions, so it may not completely reverse the effects of `generateApertureModeDimensions`. It sets the `QTTrackHasApertureModeDimensionsAttribute` property to `NO`.

```
- (void)removeApertureModeDimensions
```

Availability
Available in Mac OS X v10.3 and later.

Declared In
QTTrack.h

scaleSegment:newDuration:

```
- (void)scaleSegment:(QTTimeRange)segment newDuration:(QTTime)newDuration
```

Discussion
Scales the QTTrack segment delimited by the segment *segment* so that it will have the new duration *newDuration*. If the track is not editable, this method raises an exception.

Availability
Available in Mac OS X v10.3 and later.

Declared In
QTTrack.h

setApertureModeDimensions:forMode:

Sets the dimensions of the target track for the specified movie aperture mode.

```
- (void)setApertureModeDimensions:(NSSize)dimensions forMode:(NSString *)mode
```

Availability
Available in Mac OS X v10.3 and later.

Declared In
QTTrack.h

setAttribute:forKey:

```
-(void)setAttribute:(id)value forKey:(NSString *)attributeKey
```

Discussion
Set the track attribute *attributeKey* to the value specified by the *value* parameter. A list of supported track attributes and their acceptable values can be found in the "Constants" (page 234) section.

Availability
Available in Mac OS X v10.3 and later.

Declared In
QTTrack.h

setEnabled:

```
- (void)setEnabled:(BOOL)enabled
```

Discussion
Sets the enabled state of a QTTrack to *enabled*.

Availability
Available in Mac OS X v10.3 and later.

Related Sample Code
QTKitTimeCode

Declared In
QTTrack.h

setTrackAttributes:

```
-(void)setTrackAttributes:(NSDictionary *)attributes
```

Discussion
Set the track attributes using the key-value pairs specified in the dictionary *attributes*. A list of supported track attributes and their acceptable values can be found in the "Constants" (page 234) section.

Availability
Available in Mac OS X v10.3 and later.

Declared In
QTTrack.h

setVolume:

```
-(void)setVolume:(float)volume
```

Discussion
Sets the volume of a QTTrack to *volume*. The valid range is 0.0 to 1.0.

Availability
Available in Mac OS X v10.3 and later.

Declared In
QTTrack.h

trackAttributes

```
-(NSDictionary *)trackAttributes
```

Discussion
Returns a dictionary containing the current values of all defined track attributes. A list of supported track attributes and their acceptable values can be found in the "Constants" (page 234) section.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTTrack.h`

volume

```
-(float)volume
```

Discussion
Returns the volume of a QTTrack object. The valid range is 0.0 to 1.0.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTTrack.h`

Constants

The following constants specify the track attributes that you can get and set using the `trackAttributes` and `setTrackAttributes` methods. To get or set a single attribute, use `attributeForKey` or `setAttribute`.

Constant	Description
`QTTrackBoundsAttribute`	The bounding rectangle of a QTTrack object; the value for this key is of type `NSValue`, interpreted as an `NSRect`.
`QTTrackCreationTimeAttribute`	The creation time of a QTTrack object; the value for this key is of type `NSDate`.
`QTTrackDimensionsAttribute`	The dimensions of a QTTrack object; the value for this key is of type `NSValue`, interpreted as an `NSSize`.
`QTTrackDisplayNameAttribute`	The display name of a QTTrack object; the value for this key is of type `NSString`.
`QTTrackEnabledAttribute`	The track enabled state of a QTTrack object; the value for this key is of type `NSNumber`, interpreted as a `BOOL`.

Constant	Description
`QTTrackFormatSummaryAttribute`	An `NSString` that is a localized, human-readable string that summarizes a track's format; for example, "16-bit Integer (Big Endian), Stereo (L R), 48.000 kHz". This attribute is gettable but not settable. Mac OS X v10.5 and later.
`QTTrackHasApertureModeDimensionsAttribute`	The value to determine whether aperture mode dimensions have been set on a track, even if they are all identical to the classic dimensions (as is the case for content with square pixels and no edge-processing region).
`QTTrackIDAttribute`	The track ID of a QTTrack object; the value for this key is of type `NSNumber`, interpreted as a `long`.
`QTTrackLayerAttribute`	The track layer of a QTTrack object; the value for this key is of type `NSNumber`, interpreted as a `short`.
`QTTrackMediaTypeAttribute`	The media type of a QTTrack object; the value for this key is of type `NSString`.
`QTTrackModificationTimeAttribute`	The modification time of a QTTrack object; the value for this key is of type `NSDate`.
`QTTrackRangeAttribute`	The range of time this track occupies; the value for this key is of type `NSValue`, interpreted as a `QTTimeRange`.
`QTTrackTimeScaleAttribute`	The track time scale; the value for this key is of type `NSNumber`, interpreted as a `long`.
`QTTrackUsageInMovieAttribute`	The movie usage setting; the value for this key is of type `NSNumber`, interpreted as a `BOOL`.
`QTTrackUsageInPosterAttribute`	The poster usage setting; the value for this key is of type `NSNumber`, interpreted as a `BOOL`.
`QTTrackUsageInPreviewAttribute`	The preview usage setting; the value for this key is of type `NSNumber`, interpreted as a `BOOL`.
`QTTrackVolumeAttribute`	The volume of a QTTrack object; the value for this key is of type `NSNumber`, interpreted as a `float`.

QuickTime Kit Capture Constants Reference

Inherits from	NSObject
Framework	Library/Frameworks/QTKit.framework
Declared in	QTKit/QTError.h
Availability	Available in QuickTime 7.2.1 and later.

Overview

This describes the constants, error codes and SMPTETime operations. Error codes are returned within QTKitErrorDomain. QTKit defines extra operations on the SMPTETime type. SMPTETime is defined in `CoreAudio/CoreAudioTypes.h`.

Constants

QTErrorCaptureInputKey

An NSError userInfo key corresponding to the QTCaptureInput for which the error occurred.

```
QTErrorCaptureInputKey
```

Constants
`QTErrorCaptureInputKey`

Declared In
`HeaderFile`

QTErrorCaptureOutputKey

An NSError userInfo key corresponding to the `QTCaptureOutput` for which the error occurred.

```
QTErrorCaptureOutputKey
```

Constants
`QTErrorCaptureOutputKey`

Declared In
`HeaderFile`

QTErrorDeviceKey

An NSError userInfo key corresponding to the `QTCaptureDevice` for which the error occurred.

```
QTErrorDeviceKey
```

Constants
`QTErrorDeviceKey`

Declared In
`HeaderFile`

QTErrorExcludingDeviceKey

An NSError userInfo key corresponding to the `QTCaptureDevice` that is excluding the device for which the error occurred.

```
QTErrorExcludingDeviceKey
```

Constants
`QTErrorExcludingDeviceKey`

Declared In
`HeaderFile`

QTKitErrorDomain

The error domain for NSError codes specific to QTKit.

```
QTKitErrorDomain
```

Constants
`QTKitErrorDomain`

Declared In
`HeaderFile`

QTMediaTypeMuxed

A media type referring to media consisting of multiplexed audio and video.

```
QTMediaTypeMuxed
```

Constants
`QTMediaTypeMuxed`

Declared In
`HeaderFile`

QTErrorRecordingSuccesfullyFinishedKey

An NSError userInfo key that returns whether the products of a recording were successfully finished after recording stopped due to an error. Value is an NSNumber interpreted as a BOOL.

```
QTErrorRecordingSuccesfullyFinishedKey
```

Constants
`QTErrorRecordingSuccesfullyFinishedKey`

Declared In
`HeaderFile`

QTKit Error Codes

Error codes returned within QTKitErrorDomain.

```
enum {
    QTErrorUnknown                                  = -1,
    QTErrorIncompatibleInput                        = 1002,
    QTErrorIncompatibleOutput                       = 1003,
    QTErrorInvalidInputsOrOutputs                   = 1100,
    QTErrorDeviceAlreadyUsedbyAnotherSession        = 1101,
    QTErrorNoDataCaptured                           = 1200,
    QTErrorSessionConfigurationChanged              = 1201,
    QTErrorDiskFull                                 = 1202,
    QTErrorDeviceWasDisconnected                    = 1203,
    QTErrorMediaChanged                             = 1204,
    QTErrorMaximumDurationReached                   = 1205,
    QTErrorMaximumFileSizeReached                   = 1206,
    QTErrorMediaDiscontinuity                       = 1207,
    QTErrorDeviceNotConnected                       = 1300,
    QTErrorDeviceInUseByAnotherApplication          = 1301,
    QTErrorDeviceExcludedByAnotherDevice            = 1302,
};
```

Constants

`QTErrorUnknown`

Indicates an unexpected or unknown error.

Check `NSUnderlyingErrorKey` for an NSError representing the internal cause of the error.

Available in Mac OS X v10.5 and later.

Declared in `QTError.h`.

`QTErrorInputAlreadyConnectedToAnotherSession`

The input could not be added to the specified session because it is already connected to another session.

Check `QTErrorCaptureInputKey` for the input experiencing the error.

`QTErrorOutputAlreadyConnectedToAnotherSession`

The output could not be added to the specified session because it is already connected to another session.

Check `QTErrorCaptureOutputKey` for the output experiencing the error.

`QTErrorIncompatibleInput`
: The input could not be added to the specified session because it is incompatible with existing inputs and outputs in the session.

  Check `QTErrorCaptureInputKey` for the input experiencing the error.

  Available in Mac OS X v10.5 and later.

  Declared in `QTError.h`.

`QTErrorIncompatibleOutput`
: The output could not be added to the specified session because it is incompatible with existing inputs and outputs in the session.

  Check `QTErrorCaptureOutputKey` for the output experiencing the error.

  Available in Mac OS X v10.5 and later.

  Declared in `QTError.h`.

`QTErrorInvalidInputOrOutput`
: The input or output could not be added to the specified session because the session experiences a runtime error due to a problem with one of the inputs or outputs.

  Check `NSUnderlyingErrorKey` for an NSError representing the internal cause of the error.

`QTErrorDeviceAlreadyUsedbyAnotherSession`
: The device could not be added to the session because it experiences a runtime error trying to use a device already being used by another session.

  Available in Mac OS X v10.5 and later.

  Declared in `QTError.h`.

`QTErrorNoDataCaptured`
: Returned when no data was successfully captured during a recording or other capture operation.

  Available in Mac OS X v10.5 and later.

  Declared in `QTError.h`.

`QTErrorSessionConfigurationChanged`
: The recording has been automatically stopped because an input or output has been added or removed, or the channels of an input or output have changed.

  Check `QTErrorCaptureSuccesfullyFinishedKey` to determine if the recorded products were successfully completed when recording was stopped.

  Available in Mac OS X v10.5 and later.

  Declared in `QTError.h`.

`QTErrorDiskFull`
: The recording has been automatically stopped because the disk being used for recorded products is full.

  Check `QTErrorCaptureSuccesfullyFinishedKey` to determine if the recorded products were successfully completed when recording was stopped. This error will occur while the destination disk still has sufficient space to avoid system wide warnings about low disk space.

  Available in Mac OS X v10.5 and later.

  Declared in `QTError.h`.

`QTErrorDeviceWasDisconnected`
The recording has been automatically stopped because an input device was disconnected.

Check `QTErrorCaptureSuccesfullyFinishedKey` to determine if the capture products were successfully completed when recording was stopped.

Available in Mac OS X v10.5 and later.

Declared in `QTError.h`.

`QTErrorMediaChanged`
The recording has been automatically stopped because the format of the input media changed or the media samples were invalid.

Check `QTErrorCaptureSuccesfullyFinishedKey` to determine if the capture products were successfully completed when recording was stopped.

Available in Mac OS X v10.5 and later.

Declared in `QTError.h`.

`QTErrorMaximumDurationReached`
Returned when recording has reached the maximum duration specified by the application.

Available in Mac OS X v10.5 and later.

Declared in `QTError.h`.

`QTErrorMaximumFileSizeReached`
Returned when recording has reached the maximum file size specified by the application.

Available in Mac OS X v10.5 and later.

Declared in `QTError.h`.

`QTErrorMediaDiscontinuity`
Returned when there is a discontinuity in captured media, usually because of perfomance problems on the user's system or because of a change in a device's state. This error generally indicates that media samples have been dropped in order to maintain real time capture.

Available in Mac OS X v10.5 and later.

Declared in `QTError.h`.

`QTErrorDeviceNotConnected`
The device is not connected to the computer.

Available in Mac OS X v10.5 and later.

Declared in `QTError.h`.

`QTErrorDeviceInUseByAnotherApplication`
The device is in use by another application.

Available in Mac OS X v10.5 and later.

Declared in `QTError.h`.

`QTErrorDeviceExcludedByAnotherDevice`
The device is excluded by another device.

Check `QTErrorExcludingDeviceKey` to determine the device that needs to be closed to open the device that failed.

Available in Mac OS X v10.5 and later.

Declared in `QTError.h`.

Declared In
`QTError.h`

Functions

QTKit Functions Reference

Framework: QTKit/QTKit.h

Overview

This chapter describes the functions that are available in the QuickTime Kit framework.

Functions by Task

Creating QTTime Structures

The following functions are used to create QTTime structures:

`QTMakeTime` (page 248)
Creates a `QTTime` structure.

`QTMakeTimeScaled` (page 249)
Returns a `QTTime` structure.

`QTTimeFromString` (page 252)
Returns a `QTTime` structure.

`QTMakeTimeWithTimeRecord` (page 250)
Creates a `QTTime` structure.

`QTMakeTimeWithTimeInterval` (page 249)
Creates a `QTTime` structure.

Getting and Setting Times

The following functions are used to get and set times:

`QTGetTimeRecord` (page 247)
Returns the value of a `QTTime` structure expressed as a `TimeRecord`.

`QTGetTimeInterval` (page 247)
Returns the value of a `QTTime` structure expressed as an `NSTimeInterval`.

Comparing QTTime Structures

The following function is used to compare `QTTime` structures:

QTTimeCompare (page 252)
Returns a value of type NSComparisonResult.

Adding and Subtracting Times

The following functions are used to add and subtract times:

QTTimeIncrement (page 253)
Adds two QTTime structures.

QTTimeDecrement (page 252)
Subtracts one QTTime from another.

Getting a Time Description

The following function is used to get a time description:

QTStringFromTime (page 251)
Returns a description of a QTTime structure.

Time Range Functions

QTEqualTimeRanges (page 247)
Returns YES if the specified time ranges are identical.

QTIntersectionTimeRange (page 248)
Returns a QTTimeRange structure that represents the intersection of the two ranges.

QTMakeTimeRange (page 249)
Returns a QTTimeRange structure initialized using the QTTime structures time and duration.

QTStringFromTimeRange (page 251)
Returns a description of a QTTimeRange structure.

QTTimeInTimeRange (page 253)
Returns YES if the specified time time lies in the time range range.

QTTimeRangeEnd (page 254)
Returns a QTTime structure representing the end of the specified time range.

QTTimeRangeFromString (page 254)
Returns a QTTimeRange structure

QTUnionTimeRange (page 254)
Returns a QTTimeRange structure.

QuickTime Helper Functions

QTStringForOSType (page 250)
Returns an NSString representing the specified four-character code type.

QTOSTypeForString (page 250)
Returns a four-character code representing the specified NSString.

Functions

QTEqualTimeRanges

Returns `YES` if the specified time ranges are identical.

```
QTKIT_EXTERN BOOL QTEqualTimeRanges (
   QTTimeRange range,
   QTTimeRange range2
);
```

Discussion
This function returns `YES` if the specified time ranges are identical.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTTimeRange.h`

QTGetTimeInterval

Returns the value of a `QTTime` structure expressed as an `NSTimeInterval`.

```
QTKIT_EXTERN BOOL QTGetTimeInterval (
   QTTime time,
   NSTimeInterval *timeInterval
);
```

Discussion
This function returns, in the location to by *`timeInterval`*, the value of a `QTTime` structure expressed as a `NSTimeInterval`. Returns `YES` if the method succeeded.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTTime.h`

QTGetTimeRecord

Returns the value of a `QTTime` structure expressed as a `TimeRecord`.

```
QTKIT_EXTERN BOOL QTGetTimeRecord (
   QTTime time,
   TimeRecord *timeRecord
);
```

Discussion
This function returns, in the location pointed to by *`timeRecord`*, the value of a `QTTime` structure expressed as a `TimeRecord`. Returns `YES` if the method succeeded.

Availability
Available in Mac OS X v10.3 and later.

Related Sample Code
QTAudioExtractionPanel

Declared In
`QTTime.h`

QTIntersectionTimeRange

Returns a `QTTimeRange` structure that represents the intersection of the two ranges.

```
QTKIT_EXTERN QTTimeRange QTIntersectionTimeRange (
   QTTimeRange range1,
   QTTimeRange range2
);
```

Discussion
This function returns a `QTTimeRange` structure that represents the intersection of the two ranges. The intersection of two ranges is the largest range that includes all times that are in both ranges.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTTimeRange.h`

QTMakeTime

Creates a `QTTime` structure.

```
QTKIT_EXTERN QTTime QTMakeTime (
   long long timeValue,
   long timeScale
);
```

Discussion
This function creates a `QTTime` structure initialized using the scalar value `timeValue` and the time scale *`scale`*.

Availability
Available in Mac OS X v10.3 and later.

Related Sample Code
QTAudioExtractionPanel
QTKitCommandLine
QTKitCreateMovie
QTKitMovieShuffler

Declared In
`QTTime.h`

QTMakeTimeRange

Returns a `QTTimeRange` structure initialized using the QTTime structures `time` and `duration`.

```
QTKIT_EXTERN QTTimeRange QTMakeTimeRange (
   QTTime time,
   QTTime duration
);
```

Discussion
This function returns a `QTTimeRange` structure initialized using the `QTTime` structures time and duration. Those structures may have different time scales. In all cases, the time scale used in the new `QTTimeRange` structure is that of time.

Availability
Available in Mac OS X v10.3 and later.

Related Sample Code
QTKitCommandLine
QTKitMovieShuffler

Declared In
`QTTimeRange.h`

QTMakeTimeScaled

Returns a `QTTime` structure.

```
QTKIT_EXTERN QTTime QTMakeTimeScaled (
   QTTime time,
   long timeScale
);
```

Discussion
This function returns a `QTTime` structure whose time is set to the time of a `QTTime` structure interpreted using the time scale *`scale`*.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTTime.h`

QTMakeTimeWithTimeInterval

Creates a `QTTime` structure.

```
QTKIT_EXTERN QTTime QTMakeTimeWithTimeInterval (
   NSTimeInterval timeInterval
);
```

Discussion
Creates a QTTime structure initialized using the NSTimeInterval value *`timeInterval`*.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTTime.h`

QTMakeTimeWithTimeRecord

Creates a `QTTime` structure.

```
QTKIT_EXTERN QTTime QTMakeTimeWithTimeRecord (
   TimeRecord timeRecord
);
```

Discussion
This function creates a `QTTime` structure initialized using the values in the time record *`timeRecord`*.

Availability
Available in Mac OS X v10.3 and later.

Related Sample Code
QTAudioExtractionPanel

Declared In
`QTTime.h`

QTOSTypeForString

Returns a four-character code representing the specified NSString.

```
QTKIT_EXTERN OSType QTOSTypeForString (
   NSString *string
);
```

Discussion
This function returns a four-character code representing the specified `NSString`.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTUtilities.h`

QTStringForOSType

Returns an NSString representing the specified four-character code type.

```
QTKIT_EXTERN NSString * QTStringForOSType (
   OSType type
);
```

Discussion
This function returns an `NSString` representing the specified four-character code type.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTUtilities.h`

QTStringFromTime

Returns a description of a `QTTime` structure.

```
QTKIT_EXTERN NSString * QTStringFromTime (
   QTTime time
);
```

Discussion
This function returns a description of a `QTTime` structure. The string is in the form "`sign:days:hours:minutes:seconds:timevalue:timescale`", where sign is empty or "-". Note that this is not for user input, but for archiving and debugging purposes.

Availability
Available in Mac OS X v10.3 and later.

Related Sample Code
CIVideoDemoGL
QTAudioExtractionPanel
QTKitPlayer
QTRecorder

Declared In
`QTTime.h`

QTStringFromTimeRange

Returns a description of a `QTTimeRange` structure.

```
QTKIT_EXTERN NSString * QTStringFromTimeRange (
   QTTimeRange range
);
```

Discussion
This function returns a description of a `QTTimeRange` structure. The string is in the form "`hours:minutes:seconds.frames:: hours:minutes:seconds.frames`". Note that this is for archiving and debugging purposes, not for user display.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTTimeRange.h`

QTTimeCompare

Returns a value of type `NSComparisonResult`.

```
QTKIT_EXTERN NSComparisonResult QTTimeCompare (
   QTTime time,
   QTTime otherTime
);
```

Discussion
This function returns a value of type `NSComparisonResult` that indicates the result of comparing a `QTTime` structure with the specified `QTTime` structure *`otherTime`*.

Availability
Available in Mac OS X v10.3 and later.

Related Sample Code
QTAudioExtractionPanel
QTKitMovieShuffler

Declared In
`QTTime.h`

QTTimeDecrement

Subtracts one `QTTime` from another.

```
QTKIT_EXTERN QTTime QTTimeDecrement (
   QTTime time,
   QTTime decrement
);
```

Discussion
This function returns a `QTTime` structure whose time is set to the time of a `QTTime` structure minus that of the structure *`decrement`*.

Availability
Available in Mac OS X v10.3 and later.

Related Sample Code
QTAudioExtractionPanel

Declared In
`QTTime.h`

QTTimeFromString

Returns a `QTTime` structure.

```
QTKIT_EXTERN QTTime QTTimeFromString (
   NSString *string
);
```

Discussion
This function returns a `QTTime` structure whose time is set to the time expressed by the string; the string is assumed to be in the form "`days:hours:minutes:seconds:frames/timescale`".

Availability
Available in Mac OS X v10.3 and later.

Related Sample Code
QTAudioExtractionPanel

Declared In
`QTTime.h`

QTTimeIncrement

Adds two `QTTime` structures.

```
QTKIT_EXTERN QTTime QTTimeIncrement (
   QTTime time,
   QTTime increment
);
```

Discussion
This function returns a `QTTime` structure whose time is set to the time of a `QTTime` structure plus that of the structure *`increment`*.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTTime.h`

QTTimeInTimeRange

Returns `YES` if the specified time time lies in the time range range.

```
QTKIT_EXTERN BOOL QTTimeInTimeRange (
   QTTime time,
   QTTimeRange range
);
```

Discussion
This function returns `YES` if the specified time time lies in the time range `range`.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTTimeRange.h`

QTTimeRangeEnd

Returns a `QTTime` structure representing the end of the specified time range.

```
QTKIT_EXTERN QTTime QTTimeRangeEnd (
   QTTimeRange range
);
```

Discussion
This function returns a `QTTime` structure representing the end of the specified time range.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTTimeRange.h`

QTTimeRangeFromString

Returns a `QTTimeRange` structure

```
QTKIT_EXTERN QTTimeRange QTTimeRangeFromString (
   NSString *string
);
```

Discussion
This function returns a `QTTimeRange` structure whose range is set to the range expressed by string; the string is assumed to be in the form "`days:hours:minutes:seconds.frames/timescale~days:hours:minutes:seconds.frames/timescale`".

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTTimeRange.h`

QTUnionTimeRange

Returns a `QTTimeRange` structure.

```
QTKIT_EXTERN QTTimeRange QTUnionTimeRange (
   QTTimeRange range1,
   QTTimeRange range2
);
```

Discussion
This function returns a `QTTimeRange` structure that represents the union of the two ranges. The union of two ranges is the smallest range that includes all times that are in either range.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTTimeRange.h`

Data Types

QTKit Data Types Reference

Framework: QTKit/QTKit.h

Overview

This chapter describes the data types and constants found in the QuickTime Kit framework.

Data Types

QTTime

Defines the value and time scale of a time.

```
typedef struct {     long     long            timeValue;     long
timeScale;     long               flags; }
```

Discussion
The `QTTime` structure defines the value and time scale of a time. Currently only one flag is defined:

```
enum {
 kQTTimeIsIndefinite = 1 << 0
};
```

If this flag is set in a `QTTime` structure, the other fields should not be used. The QTKit provides a number of functions for converting and comparing `QTTime` structures.

QTTimeRange

Defines a range of time.

```
typedef struct {     QTTime time;     QTTime duration; } QTTimeRange;
```

Discussion
The `QTTimeRange` structure defines a range of time. It is used, for instance, to specify the active segment of a movie or track. The QTKit provides a number of functions for converting and comparing `QTTimeRange` structures.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTTimeRange.h`

Document Revision History

This table describes the changes to *QuickTime Kit Framework Reference*.

Date	Notes
2007-10-31	Added descriptions of two new classes, QTMovieLayer and QTCaptureLayer, and added a reference to the "QuickTime 7.2.1 Update Guide."

Document Revision History

Index

Symbols

A

B

C

D

E

F

G

H

I

L

M

N

O

P

Q

R

S

T

U

V

W