QTKit Framework Reference

Audio & Video

2009-02-26

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 QTCaptureDecompressedAudioOutput.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. Two additional classes, which support Core Animation layers for capture and movies, are also provided in the API.

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 QTKit Additions 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 QTKit Additions 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 21)
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
QTAudioContextInsert
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
CIColorTracking
CIVideoDemoGL
QTAudioContextInsert
QTAudioExtractionPanel
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	/System/Library/Frameworks/QTKit.framework
Declared in	QTKit/QTCaptureAudioPreviewOutput.h
Availability	Available in Mac OS X v10.5 and later; QuickTime 7.2.1 and later.
Related sample code	QTRecorder

Overview

This class represents an output destination for a `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 *)outputDeviceUniqueID
```

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 v10.4 and later; QuickTime 7.2.1 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 *)uniqueID
```

Parameters
uniqueID
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 v10.4 and later; QuickTime 7.2.1 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 v10.4 and later; QuickTime 7.2.1 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 v10.4 and later; QuickTime 7.2.1 and later.

Declared In
`QTCaptureAudioPreviewOutput.h`

QTCaptureConnection Class Reference

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

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 *)attributeKey
```

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; QuickTime 7.2.1.

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; QuickTime 7.2.1.

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; QuickTime 7.2.1.

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; QuickTime 7.2.1.

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; QuickTime 7.2.1.

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; QuickTime 7.2.1.

Related Sample Code
MyRecorder

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; QuickTime 7.2.1.

Declared In
`QTCaptureConnection.h`

setAttribute:forKey:

Sets a connection attribute for the given key.

```
- (void)setAttribute:(id)attribute
    forKey:(NSString *)key
```

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; QuickTime 7.2.1.

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; QuickTime 7.2.1.

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; QuickTime 7.2.1.

Declared In
`QTCaptureConnection.h`

Constants

Audio Attributes

Applications can use the following constants to display audio level meters for specific connections and to specify the volumes of audio channels. These string values can be used in key paths for key-value coding, key-value observing, and bindings.

```
NSString * const QTCaptureConnectionAudioAveragePowerLevelsAttribute;
NSString * const QTCaptureConnectionAudioPeakHoldLevelsAttribute;
NSString * const QTCaptureConnectionAudioMasterVolumeAttribute;
NSString * const QTCaptureConnectionAudioVolumesAttribute;
NSString * const QTCaptureConnectionEnabledAudioChannelsAttribute;
```

Constants

`QTCaptureConnectionAudioAveragePowerLevelsAttribute`

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

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 appear jumpy on a level meter. This string value can be used in key paths for key-value coding, key-value observing, and bindings.

Available in Mac OS X v10.5 and later.

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.

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

Available in Mac OS X v10.5 and later.

Declared in `QTCaptureConnection.h`.

`QTCaptureConnectionAudioMasterVolumeAttribute`

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

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

Available in Mac OS X v10.5 and later.

Declared in `QTCaptureConnection.h`.

`QTCaptureConnectionAudioVolumesAttribute`

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

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

Available in Mac OS X v10.5 and later.

Declared in `QTCaptureConnection.h`.

`QTCaptureConnectionEnabledAudioChannelsAttribute`

An `NSIndexSet` that specifies which audio channels should be sent through the connection. 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 through 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. This string value can be used in key paths for key-value coding, key-value observing, and bindings.

Available in Mac OS X v10.5 and later.

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`

QTCaptureDecompressedAudioOutput Class Reference

Inherits from	QTCaptureOutput : NSObject
Conforms to	NSObject (NSObject)
Framework	/System/Library/Frameworks/QTKit.framework
Declared in	QTKit/QTCaptureDecompressedAudioOutput.h
Availability	Available in QuickTime 7.6.3 and later.

Overview

This class represents an output destination for a `QTCaptureSession` object that can be used to process audio sample buffers from the audio being captured. Instances of `QTCaptureDecompressedAudioOutput` produce audio sample buffers suitable for custom high-quality realtime processing. Applications can access the audio sample buffers via the `captureOutput:didOutputAudioSampleBuffer:fromConnection:` (page 37) delegate method. Clients can also create subclasses of `QTCaptureDecompressedAudioOutput` to add custom capturing behavior.

Tasks

Decompressing Audio Output

- `delegate` (page 36)
  Returns the receiver's delegate.
- `setDelegate:` (page 36)
  Sets the receiver's delegate.
- `outputAudioSampleBuffer:fromConnection:` (page 36)
  Called whenever the receiver outputs a new audio sample buffer.
- `captureOutput:didOutputAudioSampleBuffer:fromConnection:` (page 37) *delegate method*
  Called whenever the audio data output outputs a new audio sample buffer.

Instance Methods

delegate

Returns the receiver's delegate.

```
- (id)delegate
```

Availability
Mac OS X v10.5 and later; QuickTime 7.6.3.

Declared In
`QTCaptureDecompressedAudioOutput.h`

outputAudioSampleBuffer:fromConnection:

Called whenever the receiver outputs a new audio sample buffer.

```
- (void)outputAudioSampleBuffer:(QTSampleBuffer *)sampleBuffer
    fromConnection:(QTCaptureConnection *)connection
```

Parameters

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

connection
The connection from which the audio was received.

Discussion
This method should not be invoked directly. Subclasses can override this method to provide custom processing behavior for each sample buffer. The default implementation calls the delegate's `captureOutput:didOutputAudioSampleBuffer:fromConnection:` (page 37) 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; QuickTime 7.6.3.

Declared In
`QTCaptureDecompressedAudioOutput.h`

setDelegate:

Sets the receiver's delegate.

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

Availability
Mac OS X v10.5 and later; QuickTime 7.6.3.

Declared In
QTCaptureDecompressedAudioOutput.h

Delegate Methods

captureOutput:didOutputAudioSampleBuffer:fromConnection:

Called whenever the audio data output outputs a new audio sample buffer.

```
- (void)captureOutput:(QTCaptureOutput *)captureOutput
    didOutputAudioSampleBuffer:(QTSampleBuffer *)sampleBuffer
    fromConnection:(QTCaptureConnection *)connection
```

Parameters

captureOutput

The QTCaptureDecompressedAudioOutput instance that output the frame.

sampleBuffer

A sample buffer containing the audio data and additional information about the buffer, such as its presentation time.

connection

The connection from which the audio was received.

Discussion
Delegates receive this message whenever the output produces a new audio sample buffer. Delegates can use the provided sample buffer for custom processing of captured audio.

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; QuickTime 7.6.3.

Declared In
QTCaptureDecompressedAudioOutput.h

QTCaptureDecompressedVideoOutput Class Reference

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

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:` (page 45) delegate method. Clients can also create subclasses of `QTCaptureDecompressedVideoOutput` to add custom capturing behavior.

Tasks

Decompressing Video Output

- `automaticallyDropsLateVideoFrames` (page 40)
  Returns whether the receiver discards video frames that are output before earlier frames have been processed.
- `delegate` (page 41)
  Returns the receiver's delegate.
- `setDelegate:` (page 44)
  Sets the receiver's delegate.
- `setMinimumVideoFrameInterval:` (page 44)
  Sets the minimum time interval between which the receiver should output consecutive video frames.

- `outputVideoFrame:withSampleBuffer:fromConnection:` (page 42)
  Called whenever the receiver outputs a new video frame.
- `minimumVideoFrameInterval` (page 41)
  Returns the minimum time interval between which the receiver will output consecutive video frames.
- `pixelBufferAttributes` (page 43)
  Returns the Core Video pixel buffer attributes previously set by `setPixelBufferAttributes:` that determine what kind of pixel buffers are output by the receiver.
- `setAutomaticallyDropsLateVideoFrames:` (page 43)
  Sets whether the receiver discards video frames that are output before earlier frames have been processed.
- `setPixelBufferAttributes:` (page 44)
  Sets the CoreVideo pixel buffer attributes that determine what kind of pixel buffers are output by the receiver.
- `captureOutput:didDropVideoFrameWithSampleBuffer:fromConnection:` (page 40)
  Called once for each frame that is dropped when `automaticallyDropsLateVideoFrames` is set to `YES`.
- `captureOutput:didOutputVideoFrame:withSampleBuffer:fromConnection:` (page 45) *delegate method*
  Called whenever the video preview output outputs a new video frame.

Instance Methods

automaticallyDropsLateVideoFrames

Returns whether the receiver discards video frames that are output before earlier frames have been processed.

```
- (BOOL)automaticallyDropsLateVideoFrames
```

Return Value
This method returns `YES` if the receiver drops late video frames and returns NO otherwise.

Discussion
If this method returns `YES`, the receiver will discard frames that are queued up while the thread handling existing frames is blocked in the `outputVideoFrame:withSampleBuffer:fromConnection:` or the `captureOutput:didOutputVideoFrame:withSampleBuffer:fromConnection:` delegate method. The delegate method `captureOutput:didDropVideoFrameWithSampleBuffer:fromConnection:` will be called for each frame that is dropped. The default value is `NO`.

Availability
Mac OS X v10.5 and later; QuickTime 7.6.3.

Declared In
`QTCaptureDecompressedVideoOutput.h`

captureOutput:didDropVideoFrameWithSampleBuffer:fromConnection:

Called once for each frame that is dropped when `automaticallyDropsLateVideoFrames` is set to `YES`.

```
- (void)captureOutput:(QTCaptureOutput *)captureOutput
    didDropVideoFrameWithSampleBuffer:(QTSampleBuffer *)sampleBuffer
    fromConnection:(QTCaptureConnection *)connection
```

Parameters

captureOutput

The `QTCaptureDecompressedVideoOutput` instance that dropped the late video frame.

sampleBuffer

A `QTSampleBuffer` instance containing metadata about the dropped frame, such as its duration and presentation time stamp. This sample buffer will contain none of the original video data, therefore its `bytesForAllSamples` method will return `NULL`.

connection

The connection from which the dropped video frame was received.

Discussion

When `automaticallyDropsLateVideoFrames` is set to `YES`, this method is called whenever a late video frame is dropped. This method is called once for each dropped frame and may be called before the call to the `outputVideoFrame:withSampleBuffer:fromConnection:` or the `captureOutput:didOutputVideoFrame:withSampleBuffer:fromConnection:` (page 45) delegate method during which those frames were dropped returns. The `QTSampleBuffer` object passed to this delegate method will contain metadata about the dropped video frame, such as its duration and presentation time stamp, but will contain no actual video data. Delegates should not assume that this method will be called on the main thread. Because this method may be called on the same thread that is responsible for outputting video frames, it must be efficient to prevent further capture performance problems, such as additional dropped video frames.

Availability

Mac OS X v10.5 and later; QuickTime 7.6.3.

delegate

Returns the receiver's delegate.

```
- (id)delegate
```

Availability

Mac OS X v10.5 and later.

Declared In

`QTCaptureDecompressedVideoOutput.h`

minimumVideoFrameInterval

Returns the minimum time interval between which the receiver will output consecutive video frames.

```
- (NSTimeInterval)minimumVideoFrameInterval
```

Return Value

An `NSTimeInterval` specifying the minimum interval between video frames. Returns 0 if there is no frame rate limit set.

Discussion
This method returns the minimum amount of time that should separate consecutive frames output by the receiver. This is equivalent to the inverse of the maximum frame rate. A value of 0 indicates an unlimited maximum frame rate. The default value is 0.

Availability
Mac OS X v10.5 and later; QuickTime 7.6.3.

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. 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.

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 receiver. 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`

setAutomaticallyDropsLateVideoFrames:

Sets whether the receiver discards video frames that are output before earlier frames have been processed.

```
- (void)setAutomaticallyDropsLateVideoFrames:(BOOL)automaticallyDropsLateVideoFrames
```

Parameters
automaticallyDropsLateVideoFrames
Whether the receiver should drop late video frames.

Discussion
Setting this to `YES` will cause the receiver to discard frames that are queued up while the thread handling existing frames is blocked in the `outputVideoFrame:withSampleBuffer:fromConnection:` or the `captureOutput:didOutputVideoFrame:withSampleBuffer:fromConnection:` (page 45) delegate

method. The delegate method `captureOutput:didDropVideoFrameWithSampleBuffer:fromConnection:` (page 40) will be called for each frame that is dropped. The default value is `NO`.

Availability
Mac OS X v10.5 and later; QuickTime 7.6.3.

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`

setMinimumVideoFrameInterval:

Sets the minimum time interval between which the receiver should output consecutive video frames.

```
- (void)setMinimumVideoFrameInterval:(NSTimeInterval)minimumVideoFrameInterval
```

Parameters

minimumVideoFrameInterval

An `NSTimeInterval` specifying the minimum interval between video frames. A value of 0 indicates that there should be no frame rate limit.

Discussion
This method sets the minimum amount of time that should separate consecutive frames output by the receiver. This is equivalent to the inverse of the maximum frame rate. A value of 0 indicates an unlimited maximum frame rate. The default value is 0.

Availability
Mac OS X v10.5 and later; QuickTime 7.6.3.

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 receiver. 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. 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	/System/Library/Frameworks/QTKit.framework
Declared in	QTKit/QTCaptureDevice.h
Availability	Available in QuickTime 7.2.1 and later.
Related sample code	LiveVideoMixer3 MyRecorder QT Capture Widget QTRecorder StillMotion

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:` (page 50) class method. An array of all currently connected devices can also be obtained using the `inputDevices` (page 50) 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:` (page 51) and `defaultInputDeviceWithMediaType:` (page 49) class methods. Table 7-1 details the media types supported by `QTCaptureDevice` and examples of devices that support them:

Table 7-1 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 49)
Returns a `QTCaptureDevice` instance for the default device connected to the user's system of the given media type.

`+ deviceWithUniqueID:` (page 50)
Returns a `QTCaptureDevice` instance with the identifier device UID.

`+ inputDevices` (page 50)
Returns an array of devices currently connected to the computer that can be used as input sources.

`+ inputDevicesWithMediaType:` (page 51)
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 52)
Releases application control over the device acquired in the `open:` method.

`- isConnected` (page 54)
Returns `YES` if the device is connected to the computer.

`- isInUseByAnotherApplication` (page 54)
Returns YES is the device is connected, but being exclusively used by another application.

`- open:` (page 56)
Attempts to give the application control over the device so that it can be used for capture.

`- isOpen` (page 55)
Returns `YES` if the device is open in the current application.

Getting Information About a Device

- `attributeForKey:` (page 52)
  Returns a device attribute for the given key.
- `attributeIsReadOnly:` (page 52)
  Returns whether the given attribute for the device cannot be modified.
- `deviceAttributes` (page 53)
  Returns a dictionary of the device's current attributes.
- `formatDescriptions` (page 53)
  Returns an array of stream formats currently in use by the device.
- `hasMediaType:` (page 54)
  Returns whether the receiver sends a stream with the given media type.
- `setAttribute:forKey:` (page 57)
  Sets a device attribute for the given key.
- `setDeviceAttributes:` (page 57)
  Sets attributes on the device from the key-value pairs in the given dictionary.
- `localizedDisplayName` (page 55)
  Returns a localized human-readable name for the receiver's device.
- `modelUniqueID` (page 56)
  Returns the unique ID of the model of the receiver's device.
- `uniqueID` (page 57)
  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.

Media types are defined in `QTMedia.h`.

Availability
Mac OS X v10.5 and later.

Related Sample Code
AudioDataOutputToAudioUnit
MyRecorder
QT Capture Widget
QTCompressionOptionsWindow
StillMotion

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.

Media types are defined in `QTMedia.h`.

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.

Applications can use key value coding with the `@"connected"` and `@"inUseByAnotherApplication"` keys to be notified of changes.

Availability
Mac OS X v10.5 and later.

Related Sample Code
AudioDataOutputToAudioUnit
QT Capture Widget
QTRecorder
StillMotion

Declared In
`QTCaptureDevice.h`

deviceAttributes

Returns a dictionary of the device's current attributes.

```
- (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.

Discussion
Media types are defined in QTMedia.h.

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.

Applications can use key value coding with the @"connected" and @"inUseByAnotherApplication" keys to be notified of changes.

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 has exclusive control over the current process.

Applications can use key value coding with the `@"connected"` and `@"inUseByAnotherApplication"` keys to be notified of changes.

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 successfully opened with the receiver's `open:` method. If this method returns `YES`, the device can be used immediately for capture.

Applications can use key value coding with the `@"connected"` and `@"inUseByAnotherApplication"` keys to be notified of changes.

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 receiver.

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.

Applications can use key value coding with the `@"connected"` and `@"inUseByAnotherApplication"` keys to be notified of changes.

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

Related Sample Code
AudioDataOutputToAudioUnit

MyRecorder
QT Capture Widget
QTRecorder
StillMotion

Declared In
`QTCaptureDevice.h`

setAttribute:forKey:

Sets a device attribute for the given key.

```
- (void)setAttribute:(id)attributeforKey
    :(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

Device Attributes

Constants for different device attributes.

```
NSString * const QTCaptureDeviceChangedAttributeKey;
NSString * const QTCaptureDeviceAvailableInputSourcesAttribute;
NSString * const QTCaptureDeviceInputSourceIdentifierAttribute;
NSString * const QTCaptureDeviceInputSourceIdentifierKey;
NSString * const QTCaptureDeviceInputSourceLocalizedDisplayNameKey;
NSString * const QTCaptureDeviceSuspendedAttribute;
NSString * const QTCaptureDeviceLinkedDevicesAttribute;
NSString * const QTCaptureDeviceLegacySequenceGrabberAttribute;
NSString * const QTCaptureDeviceAVCTransportControlsAttribute;
NSString * const QTCaptureDeviceAVCTransportControlsSpeedKey;
NSString * const QTCaptureDeviceAVCTransportControlsPlaybackModeKey;
```

Constants

`QTCaptureDeviceChangedAttributeKey`

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

Available in Mac OS X v10.5 and later.

Declared in `QTCaptureDevice.h`.

`QTCaptureDeviceAvailableInputSourcesAttribute`

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

Available in Mac OS X v10.5 and later.

Declared in `QTCaptureDevice.h`.

`QTCaptureDeviceInputSourceIdentifierAttribute`

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

Available in Mac OS X v10.5 and later.

Declared in `QTCaptureDevice.h`.

`QTCaptureDeviceInputSourceIdentifierKey`

An object representing a unique ID for the input source. 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.

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

Available in Mac OS X v10.5 and later.

Declared in `QTCaptureDevice.h`.

`QTCaptureDeviceInputSourceLocalizedDisplayNameKey`

The localized display name of an input source, suitable for display in a user interface. This string value can be used in key paths for key value coding, key value observing, and bindings.

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

Available in Mac OS X v10.5 and later.

Declared in `QTCaptureDevice.h`.

`QTCaptureDeviceSuspendedAttribute`

Returns whether or not data capture on the device is suspended due to a feature on the device. For example, this attribute is `YES` for the external iSight when its privacy iris is closed, or for the internal iSight on a notebook when the notebook's display is closed.

Available in Mac OS X v10.5 and later.

Declared in `QTCaptureDevice.h`.

`QTCaptureDeviceLinkedDevicesAttribute`

Returns an array of `QTCaptureDevice` objects that, although they are separate devices on the system, are a part of the same physical device as the receiver. For example, for the external iSight camera, this attribute returns an array containing a `QTCaptureDevice` for the external iSight microphone.

Available in Mac OS X v10.5 and later.

Declared in `QTCaptureDevice.h`.

`QTCaptureDeviceLegacySequenceGrabberAttribute`

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

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

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.

Available in Mac OS X v10.5 and later.

Not available to 64-bit applications.

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.

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

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

Available in Mac OS X v10.5 and later.

Declared in `QTCaptureDevice.h`.

`QTCaptureDeviceAVCTransportControlsSpeedKey`

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

Available in Mac OS X v10.5 and later.

Declared in `QTCaptureDevice.h`.

`QTCaptureDeviceAVCTransportControlsPlaybackModeKey`

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

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

Available in Mac OS X v10.5 and later.

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. 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`).

Enumerations

These are the values for the dictionary passed to `QTCaptureDeviceAVCTransportControlsAttribute`. For most cameras and tape decks, different speeds will affect the media speed.

```
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

`QTCaptureDeviceAVCTransportControlsFastestReverseSpeed`
: Media runs in reverse at greater than normal speed.

  Available in Mac OS X v10.5 and later.

  Declared in `QTCaptureDevice.h`.

`QTCaptureDeviceAVCTransportControlsVeryFastReverseSpeed`
: Media runs in reverse at greater than normal speed.

  Available in Mac OS X v10.5 and later.

  Declared in `QTCaptureDevice.h`.

`QTCaptureDeviceAVCTransportControlsFastReverseSpeed`
: Media runs in reverse at greater than normal speed.

  Available in Mac OS X v10.5 and later.

  Declared in `QTCaptureDevice.h`.

`QTCaptureDeviceAVCTransportControlsNormalReverseSpeed`
: Media runs in reverse at normal speed.

  Available in Mac OS X v10.5 and later.

  Declared in `QTCaptureDevice.h`.

`QTCaptureDeviceAVCTransportControlsSlowReverseSpeed`
Media runs in reverse at less than normal speed.
Available in Mac OS X v10.5 and later.
Declared in `QTCaptureDevice.h`.

`QTCaptureDeviceAVCTransportControlsVerySlowReverseSpeed`
Media runs in reverse at less than normal speed.
Available in Mac OS X v10.5 and later.
Declared in `QTCaptureDevice.h`.

`QTCaptureDeviceAVCTransportControlsSlowestReverseSpeed`
Media runs in reverse at less than normal speed.
Available in Mac OS X v10.5 and later.
Declared in `QTCaptureDevice.h`.

`QTCaptureDeviceAVCTransportControlsStoppedSpeed`
Media is paused.
Available in Mac OS X v10.5 and later.
Declared in `QTCaptureDevice.h`.

`QTCaptureDeviceAVCTransportControlsSlowestForwardSpeed`
Media runs forward at less than normal speed.
Available in Mac OS X v10.5 and later.
Declared in `QTCaptureDevice.h`.

`QTCaptureDeviceAVCTransportControlsVerySlowForwardSpeed`
Media runs forward at less than normal speed.
Available in Mac OS X v10.5 and later.
Declared in `QTCaptureDevice.h`.

`QTCaptureDeviceAVCTransportControlsSlowForwardSpeed`
Media runs forward at less than normal speed.
Available in Mac OS X v10.5 and later.
Declared in `QTCaptureDevice.h`.

`QTCaptureDeviceAVCTransportControlsNormalForwardSpeed`
Media runs forward at normal speed.
Available in Mac OS X v10.5 and later.
Declared in `QTCaptureDevice.h`.

`QTCaptureDeviceAVCTransportControlsFastForwardSpeed`
Media runs forward at greater than than normal speed.
Available in Mac OS X v10.5 and later.
Declared in `QTCaptureDevice.h`.

`QTCaptureDeviceAVCTransportControlsVeryFastForwardSpeed`
Media runs forward at greater than than normal speed.
Available in Mac OS X v10.5 and later.
Declared in `QTCaptureDevice.h`.

`QTCaptureDeviceAVCTransportControlsFastestForwardSpeed`
Media runs forward at greater than than normal speed.

Available in Mac OS X v10.5 and later.

Declared in `QTCaptureDevice.h`.

Notifications

QTCaptureDeviceWasConnectedNotification

Posted when a device is connected or turned on.

Availability
QuickTime 7.2.1 and later

Declared In
`QTCaptureDevice.h`

QTCaptureDeviceWasDisconnectedNotification

Posted when a device is disconnected or turned off.

Availability
QuickTime 7.2.1 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
QuickTime 7.2.1 and later

Declared In
`QTCaptureDevice.h`

QTCaptureDeviceFormatDescriptionsDidChangeNotification

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

Availability
QuickTime 7.2.1 and later

Declared In
`QTCaptureDevice.h`

QTCaptureDeviceAttributeWillChangeNotification

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

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

Availability
QuickTime 7.2.1 and later

Declared In
`QTCaptureDevice.h`

QTCaptureDeviceAttributeDidChangeNotification

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

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

Availability
QuickTime 7.2.1 and later

Declared In
`QTCaptureDevice.h`

QTCaptureDeviceInput Class Reference

Inherits from	QTCaptureInput : NSObject
Conforms to	NSObject (NSObject)
Framework	/System/Library/Frameworks/QTKit.framework
Declared in	QTKit/QTCaptureDeviceInput.h
Availability	Available in QuickTime 7.2.1 and later.
Related sample code	AudioDataOutputToAudioUnit LiveVideoMixer3 MyRecorder 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 66)
Returns the device associated with the receiver.

`- initWithDevice:` (page 67)
Returns an instance of `QTCaptureDeviceInput` associated with the given device.

`+ deviceInputWithDevice:` (page 66)
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.

```
+ (id)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.

Related Sample Code

QT Capture Widget

QTRecorder

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 Class Reference

Inherits from	QTCaptureOutput : NSObject
Conforms to	NSObject (NSObject)
Framework	/System/Library/Frameworks/QTKit.framework
Declared in	QTKit/QTCaptureFileOutput.h
Availability	Available in QuickTime 7.2.1 and later; QuickTime 7.2.1.
Related sample code	MyRecorder QT Capture Widget QTCompressionOptionsWindow 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 `recordToOutputFileURL:` (page 81) and `recordToOutputFileURL:bufferDestination:` (page 82) 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:` (page 72) method. Currently, the only concrete subclass of this class is `QTCaptureMovieFileOutput`.

Tasks

Recording File Outputs

- `outputFileURL` (page 79)
  Rreturns the file URL of the file to which the receiver is currently recording incoming buffers.
- `recordToOutputFileURL:` (page 81)
  Calls `recordToOutputFileURL:bufferDestination:` with a buffer destination of `QTCaptureFileOutputBufferDestinationNewFile`.

- `recordToOutputFileURL:bufferDestination:` (page 82)
  Sets the file written to by the receiver, specifying where the sample buffer currently in flight should be recorded.
- `recordedDuration` (page 80)
  Returns the duration of the media recorded by the receiver.
- `recordedFileSize` (page 81)
  Returns the size, in bytes, of the data recorded by the receiver to output files.
- `maximumRecordedDuration` (page 78)
  Returns the maximum duration of the media that should be recorded by the receiver.
- `setMaximumRecordedDuration:` (page 84)
  Sets the maximum duration of the media that should be recorded by the receiver.
- `maximumRecordedFileSize` (page 78)
  Returns the maximum file size, in bytes, of the file that should be recorded by the receiver.
- `setMaximumRecordedFileSize:` (page 84)
  Sets the maximum file size, in bytes, of the file that should be recorded by the receiver.
- `compressionOptionsForConnection:` (page 77)
  Returns the options the receiver uses to compress media on the given connection as it is being captured.
- `setCompressionOptions:forConnection:` (page 83)
  Sets the options the receiver uses to compress media on the given connection as it is being captured.
- `delegate` (page 77)
  Returns the receiver's delegate.
- `setDelegate:` (page 83)
  Sets the receiver's delegate.

Methods That Control Recording

- `isRecordingPaused` (page 77)
  Returns whether recording to the current output file is paused.
- `pauseRecording` (page 80)
  Pauses recording to the current output file.
- `resumeRecording` (page 82)
  Resumes recording to the current output file after it was previously paused using pauseRecording.
- `maximumVideoSize` (page 79)
  Returns the maximum dimensions within which the receiver will record video.
- `setMaximumVideoSize:` (page 84)
  Sets the maximum dimensions within which the receiver should record video.
- `minimumVideoFrameInterval` (page 79)
  Returns the minimum time interval between which the receiver will record consecutive video frames.
- `setMinimumVideoFrameInterval:` (page 85)
  Sets the minimum time interval between which the receiver should record consecutive video frames.

Methods Implemented by the Delegate

- `captureOutput:didOutputSampleBuffer:fromConnection:` (page 72)
  Gives the delegate the opportunity to inspect samples as they are received by the output and start and stop capturing at exact times.
- `captureOutput:willStartRecordingToOutputFileURL:forConnections:` (page 76)
  Informs the delegate when the output is about to start writing to a file.
- `captureOutput:didStartRecordingToOutputFileURL:forConnections:` (page 74)
  Informs the delegate when the output has started writing to a file.
- `captureOutput:shouldChangeOutputFileAtURL:forConnections:dueToError:` (page 75)
  Gives the delegate the opportunity to determine what should happen when an output file has reached a soft limit.
- `captureOutput:mustChangeOutputFileAtURL:forConnections:dueToError:` (page 74)
  Informs the delegate when an output file can no longer be written using the incoming media.
- `captureOutput:willFinishRecordingToOutputFileAtURL:forConnections:dueToError:` (page 76)
  Informs the delegate whenthe output will stop writing new samples to a file.
- `captureOutput:didFinishRecordingToOutputFileAtURL:forConnections:dueToError:` (page 71)
  Informs the delegate when an output file is ready to be opened by applications.
- `captureOutput:didPauseRecordingToOutputFileAtURL:forConnections:` (page 72)
  Called whenever the output is recording to a file and successfully pauses the recording at the request of the client.
- `captureOutput:didResumeRecordingToOutputFileAtURL:forConnections:` (page 73)
  Called whenever the output, at the request of the client, successfully resumes a file recording that was paused.

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.

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

Availability

Mac OS X v10.5 and later; QuickTime 7.2.1.

captureOutput:didOutputSampleBuffer:fromConnection:

Gives the delegate the opportunity to inspect 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.

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; QuickTime 7.2.1.

captureOutput:didPauseRecordingToOutputFileAtURL:forConnections:

Called whenever the output is recording to a file and successfully pauses the recording at the request of the client.

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

Parameters

captureOutput

The capture file output that has paused its file recording.

fileURL

The file URL of the file that is being written.

connections

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

Discussion

Delegates can use this method to be informed when a request to pause recording is actually respected. It is safe for delegates to change what the file output is currently doing (starting a new file, for example) from within this method. Clients should not assume that this method will be called on the main thread, and should also try to make this method as efficient as possible. If recording to a file is stopped, either manually or due to an error, this method is not guaranteed to be called, even if a previous call to `pauseRecording` was made.

Availability

QuickTime 7.2.1 or later.

captureOutput:didResumeRecordingToOutputFileAtURL:forConnections:

Called whenever the output, at the request of the client, successfully resumes a file recording that was paused.

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

Parameters

captureOutput

The capture file output that has resumed its paused file recording.

fileURL

The file URL of the file that is being written.

connections

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

Discussion

Delegates can use this method to be informed when a request to resume a paused recording is actually respected. It is safe for delegates to change what the file output is currently doing (starting a new file, for example) from within this method. Clients should not assume that this method will be called on the main thread, and should also try to make this method as efficient as possible. If recording to a file is stopped, either manually or due to an error, this method is not guaranteed to be called, even if a previous call to `resumeRecording` was made.

Availability

QuickTime 7.2.1 or 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; QuickTime 7.2.1.

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.

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

Availability
Mac OS X v10.5 and later; QuickTime 7.2.1.

captureOutput:shouldChangeOutputFileAtURL:forConnections:dueToError:

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.

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

Availability
Mac OS X v10.5 and later; QuickTime 7.2.1.

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.

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

Availability

Mac OS X v10.5 and later; QuickTime 7.2.1.

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; QuickTime 7.2.1.

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; QuickTime 7.2.1.

Declared In
`QTCaptureFileOutput.h`

delegate

Returns the receiver's delegate.

```
- (id)delegate
```

Discussion
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; QuickTime 7.2.1.

Declared In
`QTCaptureFileOutput.h`

isRecordingPaused

Returns whether recording to the current output file is paused.

```
- (BOOL)isRecordingPaused
```

Return Value
Returns `YES` if recording to the current output file is paused and returns `NO` otherwise.

Discussion
This method returns whether recording to the file returned by `outputFileURL` has been previously paused using the `pauseRecording` method. When a recording is paused, captured samples are not written to the output file, but new samples can be written to the same file in the future by calling `resumeRecording`. The value of this method is key value observable using the key `@"recordingPaused"`.

Availability
QuickTime 7.6.3 or later; QuickTime 7.2.1.

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; QuickTime 7.2.1.

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; QuickTime 7.2.1.

Declared In
`QTCaptureFileOutput.h`

maximumVideoSize

Returns the maximum dimensions within which the receiver will record video.

```
- (NSSize)maximumVideoSize
```

Return Value
An `NSSize` specifying the maximum dimensions at which the receiver should record video. Returns `NSZeroSize` if there is no limit.

Discussion
This method returns the maximum limit on the dimensions of video that the receiver records to a file previously set by `setMaximumVideoSize:`. When a size is set, all video recorded by the receiver will be no larger than the specified size, while still preserving the original aspect ratio of the content. A value of `NSZeroSize` indicates that there should be no limit. If this is set to a value other than `NSZeroSize`, device native compressed video, such as DV video, will be decompressed so that it can be resized. By default, there is no limit on the maximum recorded video size.

Availability
QuickTime 7.6.3 or later.

Declared In
`QTCaptureFileOutput.h`

minimumVideoFrameInterval

Returns the minimum time interval between which the receiver will record consecutive video frames.

```
- (NSTimeInterval)minimumVideoFrameInterval
```

Return Value
An `NSTimeInterval` specifying the minimum interval between video frames. Returns 0 if there is no frame rate limit set.

Discussion
This method returns the minimum amount of time that should separate consecutive frames recorded by the receiver. This is equivalent to the inverse of the maximum frame rate. A value of 0 indicates an unlimited maximum frame rate. If this is set to a value other than 0, device native compressed video, such as DV video, will be decompressed so that its frame rate can be adjusted. The default value is 0.

Availability
QuickTime 7.6.3 or later.

Declared In
`QTCaptureFileOutput.h`

outputFileURL

Rreturns the file URL of the file to which the receiver is currently recording incoming buffers.

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

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

Availability
Mac OS X v10.5 and later; QuickTime 7.2.1.

Declared In
`QTCaptureFileOutput.h`

pauseRecording

Pauses recording to the current output file.

```
- (void)pauseRecording
```

Discussion
This method causes the receiver to stop writing captured samples to the current output file returned by `outputFileURL`, but leaves the file open so that samples can be written to it in the future, when resumeRecording is called. This allows clients to record multiple media segments that are not contiguous in time to a single file.

When clients stop recording or change files using `recordToOutputFileURL:bufferDestination:` or recording automatically stops due to an error condition while recording is paused, the output file will be finished and closed normally without requiring a matching call to `resumeRecording`. When there is no current output file, or when recording is already paused, this method does nothing. This method can be called within the `captureOutput:didOutputSampleBuffer:fromConnection:` delegate method to pause recording after an exact media sample.

Availability
QuickTime 7.6.3 or 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; QuickTime 7.2.1.

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; QuickTime 7.2.1.

Declared In
`QTCaptureFileOutput.h`

recordToOutputFileURL:

Calls `recordToOutputFileURL:bufferDestination:` with a buffer destination of `QTCaptureFileOutputBufferDestinationNewFile`.

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

Parameters
url

An `url` 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.

Related Sample Code
MyRecorder

QT Capture Widget
QTCompressionOptionsWindow
QTRecorder

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

resumeRecording

Resumes recording to the current output file after it was previously paused using pauseRecording.

```
- (void)resumeRecording
```

Discussion
This method causes the receiver to resume writing captured samples to the current output file returned by outputFileURL, after recording was previously paused using pauseRecording. This allows clients to record multiple media segments that are not contiguous in time to a single file. When there is no current output

file, or when recording is not paused, this method does nothing. This method can be called within the `captureOutput:didOutputSampleBuffer:fromConnection:` delegate method to resume recording at an exact media sample.

Availability
QuickTime 7.6.3 or 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; QuickTime 7.2.1.

Declared In
`QTCaptureFileOutput.h`

setDelegate:

Sets the receiver's delegate.

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

Discussion
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; QuickTime 7.2.1.

Related Sample Code
QT Capture Widget

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; QuickTime 7.2.1.

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; QuickTime 7.2.1.

Declared In
QTCaptureFileOutput.h

setMaximumVideoSize:

Sets the maximum dimensions within which the receiver should record video.

```
- (void)setMaximumVideoSize:(NSSize)maximumVideoSize
```

Parameters

maximumVideoSize

An NSSize specifying the maximum dimensions at which the receiver should record video. A value of NSZeroSize indicates that there should be no limit.

Discussion

This method sets the maximum limit on the dimensions of video that the receiver records to a file. When a size is set, all video recorded by the receiver will be no larger than the specified size, while still preserving the original aspect ratio of the content. A value of NSZeroSize indicates that there should be no limit. If this is set to a value other than NSZeroSize, device native compressed video, such as DV video, will be decompressed so that it can be resized. By default, there is no limit on the maximum recorded video size.

Availability

QuickTime 7.6.3 or later.

Declared In

QTCaptureFileOutput.h

setMinimumVideoFrameInterval:

Sets the minimum time interval between which the receiver should record consecutive video frames.

```
- (void)setMinimumVideoFrameInterval:(NSTimeInterval)minimumVideoFrameInterval
```

Parameters

minimumVideoFrameInterval

An NSTimeInterval specifying the minimum interval between video frames. A value of 0 indicates that there should be no frame rate limit.

Discussion

This method sets the minimum amount of time that should seperate consecutive frames recorded by the receiver. This is equivalent to the inverse of the maximum frame rate. A value of 0 indicates an unlimited maximum frame rate. If this is set to a value other than 0, device native compressed video, such as DV video, will be decompressed so that its frame rate can be adjusted. The default value is 0.

Availability

QuickTime 7.6.3 or 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
};
typedef NSUInteger QTCaptureFileOutputBufferDestination;
```

Constants

`QTCaptureFileOutputBufferDestinationNewFile`

This tells the output to include the buffer currently in flight in the old file.

Available in Mac OS X v10.5 and later.

Declared in `QTCaptureFileOutput.h`.

`QTCaptureFileOutputBufferDestinationOldFile`

This tells the output to include the buffer currently in flight in the new file.

Available in Mac OS X v10.5 and later.

Declared in `QTCaptureFileOutput.h`.

QTCaptureInput Class Reference

Inherits from	NSObject
Conforms to	NSObject (NSObject)
Framework	/System/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 87)
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` object describing the media type, format, and other attributes of the connection.

Availability
Mac OS X v10.5 and later.

Related Sample Code
QTRecorder

Declared In
`QTCaptureInput.h`

QTCaptureLayer Class Reference

Inherits from	CALayer : NSObject
Conforms to	NSCoding (CALayer) CAMediaTiming (CALayer) NSObject (NSObject)
Framework	/System/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. `QTCaptureLayer` renders a capture session within a layer hierarchy.

Tasks

Creating Capture Layers

`+ layerWithSession:` (page 90)
Creates an autoreleased `QTCaptureLayer` associated with the specified `QTCaptureSession` object.

`- initWithSession:` (page 90)
Creates a `QTCaptureLayer` associated with the specified `QTCaptureSession` object.

`- session` (page 90)
Returns the capture session associated with a `QTCaptureLayer` object.

`- setSession:` (page 91)
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
```

Parameters

session

The session with which to create an autoreleased QuickTime capture layer object.

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
```

Parameters

session

The session with which to initialize the QuickTime capture layer object.

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
```

Parameters

session

The session returned by the QuickTime capture layer object.

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
```

Parameters

session

The session set or reset by the QuickTime capture layer object.

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

Declared In
QTCaptureLayer.h

QTCaptureMovieFileOutput Class Reference

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

QTCaptureOutput Class Reference

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

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`. Similar to 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 95)
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
This class creates a new output connection for each input connection of a matching media type connected to the capture session. The `connections` 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	NSObject (NSObject)
Framework	/System/Library/Frameworks/QTKit.framework
Declared in	QTKit/QTCaptureSession.h
Availability	Available in QuickTime 7.2.1 and later.
Related sample code	AudioDataOutputToAudioUnit MyRecorder QT Capture Widget QTRecorder StillMotion

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 100)
  Returns whether the receiver is running.
- `startRunning` (page 101)
  Tells the receiver to start capturing data from its inputs and sending data to its outputs.
- `stopRunning` (page 102)
  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 98)
  Adds an input to the receiver.
- `addOutput:error:` (page 99)
  Adds an output to the receiver.
- `inputs` (page 99)
  Returns an array of inputs connected to the receiver.
- `outputs` (page 100)
  Returns an array of outputs connected to the receiver.
- `removeInput:` (page 101)
  Removes an input from the receiver.
- `removeOutput:` (page 101)
  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.

Related Sample Code
LiveVideoMixer3
QT Capture Widget

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.

Related Sample Code
LiveVideoMixer3
QT Capture Widget

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.

Related Sample Code

QT Capture Widget

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.

Related Sample Code

QT Capture Widget

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.

Related Sample Code
QT Capture Widget

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.

Related Sample Code
QT Capture Widget

Declared In
`QTCaptureSession.h`

Constants

Notification Keys

Constants used as notification keys.

```
NSString * const QTCaptureSessionErrorKey
NSString * const QTCaptureSessionRuntimeErrorNotification
```

Constants

`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.

QuickTime 7.2.1 and later.

Declared in `QTCaptureSession.h`.

`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.

QuickTime 7.2.1 and later.

Declared in `QTCaptureSession.h`.

QTCaptureVideoPreviewOutput Class Reference

Inherits from	QTCaptureOutput : NSObject
Conforms to	NSObject (NSObject)
Framework	/System/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 a `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:` (page 109) 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, since 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.

Note that clients should not attempt to access or configure a QTCaptureView's preview output.

Tasks

Previewing Output

- `delegate` (page 106)
  Returns the receiver's delegate.

- `pixelBufferAttributes` (page 107)
  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 108)
  Sets the CoreVideo pixel buffer attributes that determine what kind of pixel buffers are output by the receiver.
- `visualContextForConnection:` (page 109)
  Returns the QuickTime visual context used to preview the video for the given connection.
- `outputVideoFrame:withSampleBuffer:fromConnection:` (page 106)
  Called whenever the receiver outputs a new video frame.
- `setDelegate:` (page 108)
  Sets the receiver's delegate.
- `setVisualContext:forConnection:` (page 108)
  Sets the QuickTime visual context used to preview the video for the described connection.

Capturing Output

`- captureOutput:didOutputVideoFrame:withSampleBuffer:fromConnection:` (page 109) *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.

Declared In
`QTCaptureVideoPreviewOutput.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
`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`

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
`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 can 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	/System/Library/Frameworks/QTKit.framework
Declared in	QTKit/QTCaptureView.h
Availability	Available in QuickTime 7.2.1 and later.
Related sample code	MyRecorder QT Capture Widget QTCompressionOptionsWindow QTRecorder StillMotion

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 112)
  Returns an array of output video connections that can be previewed.
- `captureSession` (page 113)
  Returns the capture session being previewed by the receiver.
- `setCaptureSession:` (page 114)
  Sets the capture session to be previewed by the receiver.
- `setVideoPreviewConnection:` (page 115)
  Sets the output connection to be previewed by the receiver.

- `videoPreviewConnection` (page 116)
  Returns the output connection being previewed by the receiver.

Controlling View Appearance

- `fillColor` (page 113)
  Returns the fill color drawn in the area of the view not covered by the video preview.
- `preservesAspectRatio` (page 114)
  Returns whether the receiver preserves the aspect ratio of the video preview when drawing it.
- `previewBounds` (page 114)
  Returns the rectangle occupied by the video preview in the view.
- `setFillColor:` (page 115)
  Sets the fill color drawn in the area of the view not covered by the video preview.
- `setPreservesAspectRatio:` (page 115)
  Sets whether the receiver preserves the aspect ratio of the video preview when drawing it.

Getting and Setting a Delegate

- `delegate` (page 113)
  Returns the receiver's delegate.
- `setDelegate:` (page 115)
  Sets the receiver's delegate.

Methods Implemented by the Delegate

- `view:willDisplayImage:` (page 116) *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.

If there are multiple video connections that can be previewed, this method can determine which the view will display.

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.

Related Sample Code
QT Capture Widget

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.

If there are multiple video connections that can be previewed, this method can determine which the view will display.

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.

If there are multiple video connections that can be previewed, this method can determine which the view will display.

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 Class Reference

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

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 "`Compression Options Identifiers`" (page 122) section that describes the Compression Options Identifiers.

Note that not all documented identifiers may be available for a given system configuration. Clients should always query for available identifiers first.

Tasks

Creating and Configuring Compression Options

`+ compressionOptionsIdentifiersForMediaType:` (page 120)
Returns all of the possible identifiers for the given media type that can be used with `compressionOptionsWithIdentifier:` on the user's system.

`+ compressionOptionsWithIdentifier:` (page 120)
Returns a compression options object configured for the given identifier.

Receiving Compression Options

`- mediaType` (page 122)
The media type on which the receiver's compression options should be used.

- `localizedDisplayName` (page 121)
  A short localized name describing the receiver's compression options.
- `localizedCompressionOptionsSummary` (page 121)
  A localized summary of the receiver's compression options.
- `isEqualToCompressionOptions:` (page 121)
  Returns whether the receiver contains options identical to those in the given compression options object.

Class Methods

compressionOptionsIdentifiersForMediaType:

Returns all of the possible identifiers for the given media type that can be used with `compressionOptionsWithIdentifier:` on the user's system.

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

Parameters

mediaType
  A media type used to create compression options.

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

Discussion
Media types are defined in `QTMedia.h`.

Availability
Mac OS X v10.5 and later.

Related Sample Code
QTCompressionOptionsWindow

Declared In
`QTCompressionOptions.h`

compressionOptionsWithIdentifier:

Returns a compression options object configured for the given identifier.

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

Parameters

identifier
  The identifier for the compression options object.

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

Availability
Mac OS X v10.5 and later.

Related Sample Code
MyRecorder
QTCompressionOptionsWindow

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
```

Parameters

compressionOptions
The compression options of the compression options object.

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.

Related Sample Code
QTCompressionOptionsWindow

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

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. Each identifier represents a set of options that determine how media will be compressed.

```
QTCompressionOptionsLosslessAppleIntermediateVideo;
QTCompressionOptionsLosslessAnimationVideo;
QTCompressionOptions120SizeH264Video;
QTCompressionOptions240SizeH264Video;
QTCompressionOptionsSD480SizeH264Video;
QTCompressionOptions120SizeMPEG4Video;
QTCompressionOptions240SizeMPEG4Video;
QTCompressionOptionsSD480SizeMPEG4Video;
QTCompressionOptionsLosslessALACAudio;
QTCompressionOptionsHighQualityAACAudio;
QTCompressionOptionsVoiceQualityAACAudio;
```

Constants

`QTCompressionOptionsLosslessAppleIntermediateVideo`

Compresses video using the Apple Intermediate codec at lossless quality.

This is appropriate for an intermediate format for media that requires further processing.

Only available in 32-bit.

`QTCompressionOptionsLosslessAnimationVideo`

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

This is appropriate for an intermediate format for media that requires further processing.

`QTCompressionOptions120SizeH264Video`

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

This is appropriate for delivery to low-bandwidth and low-capacity destinations.

`QTCompressionOptions240SizeH264Video`

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

This is appropriate for delivery to medium-bandwidth and medium-capacity destinations.

`QTCompressionOptionsSD480SizeH264Video`

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

This is appropriate for delivery to medium and high-bandwidth and medium- and high-capacity destinations.

`QTCompressionOptions120SizeMPEG4Video`

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

This is appropriate for delivery to low-bandwidth and low-capacity destinations.

Only available in 32-bit.

`QTCompressionOptions240SizeMPEG4Video`

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

This is appropriate for delivery to medium-bandwidth and medium-capacity destinations.

Only available in 32-bit.

`QTCompressionOptionsSD480SizeMPEG4Video`

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

This is appropriate for delivery to medium and high-bandwidth and medium- and high-capacity destinations.

Only available in 32-bit.

`QTCompressionOptionsLosslessALACAudio`

Compresses audio using the Apple Lossless codec.

This is appropriate for an intermediate format for media that requires further processing.

`QTCompressionOptionsHighQualityAACAudio`

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

This is appropriate for delivery of high-quality music and other audio.

`QTCompressionOptionsVoiceQualityAACAudio`

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

This is appropriate for delivery of voice recordings.

QTDataReference Class Reference

Inherits from	NSObject
Conforms to	NSCoding NSObject (NSObject)
Framework	/System/Library/Frameworks/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 filenames or URLs, or in memory accessed using handles, pointers, or NSData objects.

Tasks

Creating a QTDataReference

`+ dataReferenceWithDataRef:type:` (page 127)
Creates a `QTDataReference` object of type *type* initialized with data from *dataRef*.

`+ dataReferenceWithDataRefData:type:` (page 127)
Creates a `QTDataReference` object of type *type* initialized with data from *dataRefData*.

`+ dataReferenceWithReferenceToFile:` (page 128)
Creates a `QTDataReference` object for the file *fileName*.

`+ dataReferenceWithReferenceToURL:` (page 129)
Creates a `QTDataReference` object for the URL *url*.

`+ dataReferenceWithReferenceToData:` (page 127)
Creates a `QTDataReference` object for the data block *data*.

`+ dataReferenceWithReferenceToData:name:MIMEType:` (page 128)
Creates a `QTDataReference` object for the data block *data*.

Initializing a QTDataReference

- `initWithDataRef:type:` (page 130)
  Initializes a newly created `QTDataReference` object with data from *`dataRef`*.
- `initWithDataRefData:type:` (page 130)
  Initializes a newly created QTDataReference object with data from *`dataRefData`*.
- `initWithReferenceToFile:` (page 131)
  Initializes a newly created `QTDataReference` object for the file *`fileName`*.
- `initWithReferenceToURL:` (page 131)
  Initializes a newly created `QTDataReference` object for the URL *`url`*.
- `initWithReferenceToData:` (page 130)
  Initializes a newly created `QTDataReference` object for the data block *`data`*.
- `initWithReferenceToData:name:MIMEType:` (page 130)
  Initializes a newly created `QTDataReference` object for the data block *`data`*.

Getting and Setting Data Reference Information

- `dataRef` (page 129)
  Returns the QuickTime data reference associated with a `QTDataReference` object.
- `dataRefData` (page 129)
  Returns the QuickTime data reference data associated with a `QTDataReference` object, stored in an NSData object.
- `dataRefType` (page 129)
  Returns the type of the data reference associated with a `QTDataReference` object.
- `referenceFile` (page 132)
  Returns the file name of the data reference associated with a `QTDataReference` object.
- `referenceURL` (page 132)
  Returns the URL of the data reference associated with a `QTDataReference` object.
- `referenceData` (page 132)
  Returns the reference data of a `QTDataReference` object, that is, the NSData object passed to `initWithReferenceToData` or `initWithReferenceToData:name:MIMEType`.
- `name` (page 132)
  Returns the name in a filenaming extension associated with a `QTDataReference` object.
- `MIMEType` (page 131)
  Returns the type in a MIME type extension associated with a `QTDataReference` object.
- `setDataRef:` (page 133)
  Sets the data reference data of a `QTDataReference` object to *`dataRef`*.
- `setDataRefType:` (page 133)
  Sets the data reference type of a `QTDataReference` object to *`type`*.

Class Methods

dataReferenceWithDataRef:type:

Creates a `QTDataReference` object of type *`type`* initialized with data from *`dataRef`*.

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

Parameters

`dataRef`

The data reference stored as a handle in a `QTDataReference` object.

`type`

The type of initialized data from a data reference.

Discussion

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:

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

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

Parameters

`dataRefData`

The `NSData` object with data referenced data.

`type`

The type initialized with data.

Availability

Available in Mac OS X v10.3 and later.

Declared In

`QTDataReference.h`

dataReferenceWithReferenceToData:

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

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

Parameters

data

The data for the `QTDataReference` object.

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

Declared In
`QTDataReference.h`

dataReferenceWithReferenceToData:name:MIMEType:

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

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

Parameters

data

The data of the `QTDataReference` object.

name

The name of the `QTDataReference` object.

MIMEType

The MIME type for the data reference.

Discussion
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:

Creates a `QTDataReference` object for the file *fileName*.

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

Parameters

fileName

The file name for a full path for a file.

Discussion
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:

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

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

Parameters

`url`

The URL for the `QTDataReference` object.

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

Declared In
`QTDataReference.h`

Instance Methods

dataRef

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

```
- (Handle)dataRef
```

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

Declared In
`QTDataReference.h`

dataRefData

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

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

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

Declared In
`QTDataReference.h`

dataRefType

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

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

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

Declared In
QTDataReference.h

initWithDataRef:type:

Initializes a newly created QTDataReference object with data from *dataRef*.

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

Discussion
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:

Initializes a newly created QTDataReference object with data from *dataRefData*.

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

Discussion
The QTDataReference is of type *dataRefType*.

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

Declared In
QTDataReference.h

initWithReferenceToData:

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

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

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

Declared In
QTDataReference.h

initWithReferenceToData:name:MIMEType:

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

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

Discussion
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:

Initializes a newly created `QTDataReference` object for the file *`fileName`*.

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

Parameters

`fileName`
The file name for the file.

Discussion
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:

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

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

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

Declared In
`QTDataReference.h`

MIMEType

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

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

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

Declared In
QTDataReference.h

name

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

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

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

Declared In
QTDataReference.h

referenceData

Returns the reference data of a QTDataReference object, that is, the NSData object passed to initWithReferenceToData or initWithReferenceToData:name:MIMEType.

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

Discussion
For some QTDataReference objects, this may be NIL.

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

Declared In
QTDataReference.h

referenceFile

Returns the file name of the data reference associated with a QTDataReference object.

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

Discussion
For some QTDataReference objects, this name may be NIL.

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

Declared In
QTDataReference.h

referenceURL

Returns the URL of the data reference associated with a QTDataReference object.

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

Discussion
For some `QTDataReference` objects, this URL may be `NIL`.

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

Declared In
`QTDataReference.h`

setDataRef:

Sets the data reference data of a `QTDataReference` object to *dataRef*.

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

Discussion
The previous data reference data is disposed of.

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

Declared In
`QTDataReference.h`

setDataRefType:

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

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

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

Declared In
`QTDataReference.h`

Constants

Data Reference Types

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

```
NSString * const QTDataReferenceTypeFile;
NSString * const QTDataReferenceTypeHandle;
NSString * const QTDataReferenceTypePointer;
NSString * const QTDataReferenceTypeResource;
NSString * const QTDataReferenceTypeURL;
```

Constants

`QTDataReferenceTypeFile`

The file type for a `QTDataReference` object.

Available in Mac OS X v10.4 and later.

Declared in `QTDataReference.h`.

`QTDataReferenceTypeHandle`

The handle type for a `QTDataReference` object.

Available in Mac OS X v10.4 and later.

Declared in `QTDataReference.h`.

`QTDataReferenceTypePointer`

The pointer type for a `QTDataReference` object.

Available in Mac OS X v10.4 and later.

Declared in `QTDataReference.h`.

`QTDataReferenceTypeResource`

The resource type for a `QTDataReference` object.

Available in Mac OS X v10.4 and later.

Declared in `QTDataReference.h`.

`QTDataReferenceTypeURL`

The URL type for a `QTDataReference` object.

Available in Mac OS X v10.4 and later.

Declared in `QTDataReference.h`.

QTFormatDescription Class Reference

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

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 "Core Audio and Video Types" (page 138) 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 136)
  Returns the current value of the format description attribute for the given key.
- `formatDescriptionAttributes` (page 136)
  Returns a dictionary of all attributes set for the receiver.
- `formatType` (page 137)
  Returns the format type of the described media, a four character code representing the format or codec type.
- `isEqualToFormatDescription:` (page 137)
  Returns whether the receiver describes the same format as the given format description.

- `localizedFormatSummary` (page 137)
  Returns a localized summary of the media format.
- `mediaType` (page 138)
  Returns the media type of the described media.
- `quickTimeSampleDescription` (page 138)
  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
```

Parameters

key
  The key for the desired format description attribute.

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.

Related Sample Code
AudioDataOutputToAudioUnit

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, a four character code representing the format or codec type.

```
- (UInt32)formatType
```

Parameters

formatType

The format type for the described media.

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
```

Parameters

formatDescription

The format description for the `QTFormatDescription` object.

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
```

Parameters

mediaType

The QuickTime media type of the described media object.

Return Value

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

Discussion

Media types are defined in `QTMedia.h`.

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.

Declared In

`QTFormatDescription.h`

Constants

Core Audio and Video Types

Constants for different core audio and video types.

```
NSString * const QTFormatDescriptionAudioChannelLayoutAttribute;
NSString * const QTFormatDescriptionAudioMagicCookieAttribute;
NSString * const QTFormatDescriptionAudioStreamBasicDescriptionAttribute;
NSString * const QTFormatDescriptionVideoCleanApertureDisplaySizeAttribute;
NSString * const QTFormatDescriptionVideoEncodedPixelsSizeAttribute;
NSString * const QTFormatDescriptionVideoProductionApertureDisplaySizeAttribute;
```

Constants

`QTFormatDescriptionAudioChannelLayoutAttribute`

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

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

Declared in `QTFormatDescription.h`.

QuickTime 7.2 and later.

`QTFormatDescriptionAudioMagicCookieAttribute`

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

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

Declared in `QTFormatDescription.h`.

QuickTime 7.2 and later.

`QTFormatDescriptionAudioStreamBasicDescriptionAttribute`

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

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

Declared in `QTFormatDescription.h`.

QuickTime 7.2 and later.

`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.

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

Declared in `QTFormatDescription.h`.

QuickTime 7.2 and later.

`QTFormatDescriptionVideoEncodedPixelsSizeAttribute`

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

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

Declared in `QTFormatDescription.h`.

QuickTime 7.2 and later.

`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.

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

Declared in `QTFormatDescription.h`.

QuickTime 7.2 and later.

QTMedia Class Reference

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

Overview

A `QTMedia` object is an object that represents the data associated with a `QTTrack` object. A `QTMovie` object typically contains one or more streams of media data, which are represented by `QTTrack` objects. A `QTTrack` object has exactly one `QTMedia` object associated with it. The `QTMedia` object exposes attributes such as media type and media characteristics. When a `QTMovie` object has been initialized with `QTMovieOpenForPlaybackAttribute` set to `NO`, a `QTMedia` object wraps the underlying QuickTime media (of type `Media`).

Tasks

Creating a QTMedia Object

`+ mediaWithQuickTimeMedia:error:` (page 142)
Returns a QTMedia object associated with a QuickTime Media.

Initializing a QTMedia Object

`- initWithQuickTimeMedia:error:` (page 144)
Returns a `QTMedia` object associated with a QuickTime Media.

Accessing Media Properties

- `track` (page 146)
  Returns the `QTTrack` object associated with a `QTMedia` object.
- `hasCharacteristic:` (page 143)
  Indicates whether a `QTMedia` object has a specified characteristic.
- `attributeForKey:` (page 143)
  Returns the current value of an attribute of a `QTMedia` object.
- `setAttribute:forKey:` (page 145)
  Sets an attribute of a `QTMedia` object to a specified value.
- `mediaAttributes` (page 144)
  Returns a dictionary containing the current values of all public attributes of a `QTMedia` object.
- `setMediaAttributes:` (page 146)
  Sets the attributes of a `QTMedia` object using the key-value pairs in a specified dictionary.

Accessing QuickTime Media Data

- `quickTimeMedia` (page 145)
  Returns the QuickTime media associated with the media object.

Class Methods

mediaWithQuickTimeMedia:error:

Returns a QTMedia object associated with a QuickTime Media.

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

Parameters

media
  The QuickTime media data with which to initialize the media object.

errorPtr
  On return, if the media object could not be created, a pointer to an error indicating the reason for the failure.

Return Value
The newly created media object.

Discussion
This method cannot be called when the movie containing this media has been initialized with `QTMovieOpenForPlaybackAttribute` set to `YES`. In addition, this method cannot be called by 64-bit applications.

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

Declared In
QTMedia.h

Instance Methods

attributeForKey:

Returns the current value of an attribute of a QTMedia object.

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

Parameters

attributeKey

An NSString object that specifies the attribute to be read; pass strings like QTMediaTimeScaleAttribute or QTMediaTypeAttribute. Possible attribute keys are listed in "Media Attributes" (page 150).

Return Value
The value of the specified attribute.

Discussion
This method can be called when the movie containing this media has been initialized with QTMovieOpenForPlaybackAttribute set to YES.

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

See Also
- setAttribute:forKey: (page 145)

Related Sample Code
QTMetadataEditor

Declared In
QTMedia.h

hasCharacteristic:

Indicates whether a QTMedia object has a specified characteristic.

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

Parameters

characteristic

An NSString object that specifies the characteristic to be read; pass strings like QTMediaCharacteristicVisual or QTMediaCharacteristicAudio. Possible characteristics are listed in "Media Characteristics" (page 149).

Return Value
Returns YES if the QTMedia object has the specified characteristic, NO otherwise.

Discussion
This method can be called when the movie containing this media has been initialized with `QTMovieOpenForPlaybackAttribute` set to `YES`.

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

Declared In
`QTMedia.h`

initWithQuickTimeMedia:error:

Returns a `QTMedia` object associated with a QuickTime Media.

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

Parameters

media

The QuickTime media with which to initialize the `QTMedia` object.

errorPtr

A pointer to an `NSError` object; if a `QTMedia` object cannot be created, an `NSError` object is returned in this location.

Return Value
The newly initialized media object.

Discussion
This method cannot be called when the movie containing this media has been initialized with `QTMovieOpenForPlaybackAttribute` set to `YES`. In addition, this method cannot be called by 64-bit applications.

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

Declared In
`QTMedia.h`

mediaAttributes

Returns a dictionary containing the current values of all public attributes of a `QTMedia` object.

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

Return Value
A dictionary containing all of the media's attributes.

Discussion
This method can be called when the movie containing this media has been initialized with `QTMovieOpenForPlaybackAttribute` set to `YES`. Possible attribute keys are listed in "Media Attributes" (page 150).

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

See Also
- `setMediaAttributes:` (page 146)

Declared In
`QTMedia.h`

quickTimeMedia

Returns the QuickTime media associated with the media object.

```
- (Media)quickTimeMedia
```

Return Value
The QuickTime media associated with the media object.

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

Declared In
`QTMedia.h`

setAttribute:forKey:

Sets an attribute of a `QTMedia` object to a specified value.

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

Parameters

value

An object that specifies the value of the attribute to be written.

attributeKey

An `NSString` object that specifies the attribute to be written; pass strings like `QTMediaTimeScaleAttribute` or `QTMediaTypeAttribute`. Possible attribute keys are listed in "Media Attributes" (page 150).

Discussion
This method can be called when the movie containing this media has been initialized with `QTMovieOpenForPlaybackAttribute` set to `YES`. However, certain attributes may not be writable when the movie containing this media has been initialized with `QTMovieOpenForPlaybackAttribute` set to `YES`.

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

See Also
- `attributeForKey:` (page 143)

Declared In
`QTMedia.h`

setMediaAttributes:

Sets the attributes of a `QTMedia` object using the key-value pairs in a specified dictionary.

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

Parameters

attributes

An `NSDictionary` object that specifies the attributes to set and their desired values.

Discussion

This method can be called when the movie containing this media has been initialized with `QTMovieOpenForPlaybackAttribute` set to `YES`. However, certain attributes may not be writable when the movie containing this media has been initialized with `QTMovieOpenForPlaybackAttribute` set to `YES`. Possible attribute keys are listed in "Media Attributes" (page 150).

Availability

Available in Mac OS X v10.3 and later.

See Also

`- mediaAttributes` (page 144)

Declared In

`QTMedia.h`

track

Returns the `QTTrack` object associated with a `QTMedia` object.

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

Return Value

The `QTTrack` object that contains the media.

Discussion

This method can be called when the movie containing this media has been initialized with `QTMovieOpenForPlaybackAttribute` set to `YES`.

Availability

Available in Mac OS X v10.3 and later.

Declared In

`QTMedia.h`

Constants

Media Types

Constants for different media types. Compare these constants with the value associated with the `QTMediaTypeAttribute` key.

```
NSString * const QTMediaTypeVideo;
NSString * const QTMediaTypeSound;
NSString * const QTMediaTypeText;
NSString * const QTMediaTypeBase;
NSString * const QTMediaTypeMPEG;
NSString * const QTMediaTypeMusic;
NSString * const QTMediaTypeTimeCode;
NSString * const QTMediaTypeSprite;
NSString * const QTMediaTypeFlash;
NSString * const QTMediaTypeMovie;
NSString * const QTMediaTypeTween;
NSString * const QTMediaType3D;
NSString * const QTMediaTypeSkin;
NSString * const QTMediaTypeQTVR;
NSString * const QTMediaTypeHint;
NSString * const QTMediaTypeStream;
NSString * const QTMediaTypeMuxed;
NSString * const QTMediaTypeQuartzComposer;
NSString * const QTMediaTypeSubtitle;
NSString * const QTMediaTypeClosedCaption;
```

Constants

`QTMediaTypeVideo`
The media type of a video track.
Available in Mac OS X v10.4 and later.
Declared in `QTMedia.h`.

`QTMediaTypeSound`
The media type of a sound track.
Available in Mac OS X v10.4 and later.
Declared in `QTMedia.h`.

`QTMediaTypeText`
The media type of a text track.
Available in Mac OS X v10.4 and later.
Declared in `QTMedia.h`.

`QTMediaTypeBase`
The media type of a base track.
Available in Mac OS X v10.4 and later.
Declared in `QTMedia.h`.

`QTMediaTypeMPEG`
The media type of a MPEG track.
Available in Mac OS X v10.4 and later.
Declared in `QTMedia.h`.

`QTMediaTypeMusic`
The media type of a music track.
Available in Mac OS X v10.4 and later.
Declared in `QTMedia.h`.

`QTMediaTypeTimeCode`
The media type of a timecode track.
Available in Mac OS X v10.4 and later.
Declared in `QTMedia.h`.

`QTMediaTypeSprite`
The media type of a sprite track.
Available in Mac OS X v10.4 and later.
Declared in `QTMedia.h`.

`QTMediaTypeFlash`
The media type of a flash track.
Available in Mac OS X v10.4 and later.
Declared in `QTMedia.h`.

`QTMediaTypeMovie`
The media type of a movie track.
Available in Mac OS X v10.4 and later.
Declared in `QTMedia.h`.

`QTMediaTypeTween`
The media type of a tween track.
Available in Mac OS X v10.4 and later.
Declared in `QTMedia.h`.

`QTMediaType3D`
The media type of a QuickDraw 3D track.
Available in Mac OS X v10.4 and later.
Declared in `QTMedia.h`.

`QTMediaTypeSkin`
The media type of a skin track
Available in Mac OS X v10.4 and later.
Declared in `QTMedia.h`.

`QTMediaTypeQTVR`
The media type of a QuickTime VR track.
Available in Mac OS X v10.4 and later.
Declared in `QTMedia.h`.

`QTMediaTypeHint`
The media type of a hint track.
Available in Mac OS X v10.4 and later.
Declared in `QTMedia.h`.

`QTMediaTypeStream`
The media type of a stream track.
Available in Mac OS X v10.4 and later.
Declared in `QTMedia.h`.

`QTMediaTypeMuxed`
The media type of a multiplexed audio and video track.

Available in Mac OS X v10.5 and later.

Declared in `QTMedia.h`.

`QTMediaTypeQuartzComposer`
The media type of a Quartz Composer track.

Available in Mac OS X v10.5 and later.

Declared in `QTMedia.h`.

`QTMediaTypeSubtitle`
The media type of a subtitle track.

Mac OS X v10.6 and QuickTime 7.6.3 and later.

Declared in `QTMedia.h`.

`QTMediaTypeClosedCaption`
The media type of a closed caption track.

Mac OS X v10.6 and QuickTime 7.6.3 and later.

Declared in `QTMedia.h`.

Media Characteristics

Characteristics of a given media. You can query for these characteristics using the `hasCharacteristic:` (page 143) method.

```
NSString * const QTMediaCharacteristicVisual;
NSString * const QTMediaCharacteristicAudio;
NSString * const QTMediaCharacteristicCanSendVideo;
NSString * const QTMediaCharacteristicProvidesActions;
NSString * const QTMediaCharacteristicNonLinear;
NSString * const QTMediaCharacteristicCanStep;
NSString * const QTMediaCharacteristicHasNoDuration;
NSString * const QTMediaCharacteristicHasSkinData;
NSString * const QTMediaCharacteristicProvidesKeyFocus;
NSString * const QTMediaCharacteristicHasVideoFrameRate;
```

Constants

`QTMediaCharacteristicVisual`
The media has visual data.

Available in Mac OS X v10.4 and later.

Declared in `QTMedia.h`.

`QTMediaCharacteristicAudio`
The media has audio data.

Available in Mac OS X v10.4 and later.

Declared in `QTMedia.h`.

`QTMediaCharacteristicCanSendVideo`
The media can send visual data to another track.

Available in Mac OS X v10.4 and later.

Declared in `QTMedia.h`.

`QTMediaCharacteristicProvidesActions`
The media has actions.
Available in Mac OS X v10.4 and later.
Declared in `QTMedia.h`.

`QTMediaCharacteristicNonLinear`
The media is non-linear.
Available in Mac OS X v10.4 and later.
Declared in `QTMedia.h`.

`QTMediaCharacteristicCanStep`
The media can step.
Available in Mac OS X v10.4 and later.
Declared in `QTMedia.h`.

`QTMediaCharacteristicHasNoDuration`
The media has no duration.
Available in Mac OS X v10.4 and later.
Declared in `QTMedia.h`.

`QTMediaCharacteristicHasSkinData`
The media has skin data.
Available in Mac OS X v10.4 and later.
Declared in `QTMedia.h`.

`QTMediaCharacteristicProvidesKeyFocus`
Key events can be focused at the media.
Available in Mac OS X v10.4 and later.
Declared in `QTMedia.h`.

`QTMediaCharacteristicHasVideoFrameRate`
The media has a video frame rate.
Available in Mac OS X v10.4 and later.
Declared in `QTMedia.h`.

Media Attributes

The following constants are keys for the media attributes that you can get and set using the `mediaAttributes` (page 144) and `setMediaAttributes:` (page 146) methods. To get or set a single attribute, use `attributeForKey:` (page 143) or `setAttribute:forKey:` (page 145).

```
NSString * const QTMediaCreationTimeAttribute;
NSString * const QTMediaDurationAttribute;
NSString * const QTMediaModificationTimeAttribute;
NSString * const QTMediaSampleCountAttribute;
NSString * const QTMediaQualityAttribute;
NSString * const QTMediaTimeScaleAttribute;
NSString * const QTMediaTypeAttribute;
```

Constants

`QTMediaCreationTimeAttribute`

The creation time. The value for this key is of type `NSDate`.

This attribute can be read but not written. This attribute can be read but not written when the movie containing this media has been initialized with `QTMovieOpenForPlaybackAttribute` set to `YES`.

Available in Mac OS X v10.4 and later.

Declared in `QTMedia.h`.

`QTMediaDurationAttribute`

The duration. The value for this key is of type `NSValue`, interpreted as a `QTTime` (page 297).

This attribute can be read but not written. This attribute can be read but not written when the movie containing this media has been initialized with `QTMovieOpenForPlaybackAttribute` set to `YES`.

Available in Mac OS X v10.4 and later.

Declared in `QTMedia.h`.

`QTMediaModificationTimeAttribute`

The modification time. The value for this key is of type `NSDate`.

This attribute can be read but not written. This attribute can be read but not written when the movie containing this media has been initialized with `QTMovieOpenForPlaybackAttribute` set to `YES`.

Available in Mac OS X v10.4 and later.

Declared in `QTMedia.h`.

`QTMediaSampleCountAttribute`

The media sample count. The value for this key is of type `NSNumber`, interpreted as a `long`.

This attribute can be read but not written. This attribute can be read but not written when the movie containing this media has been initialized with `QTMovieOpenForPlaybackAttribute` set to `YES`.

Available in Mac OS X v10.4 and later.

Declared in `QTMedia.h`.

`QTMediaQualityAttribute`

The media quality. The value for this key is of type `NSNumber`, interpreted as a `short`.

This attribute can be read but not written. This attribute can be read but not written when the movie containing this media has been initialized with `QTMovieOpenForPlaybackAttribute` set to `YES`.

Available in Mac OS X v10.4 and later.

Declared in `QTMedia.h`.

`QTMediaTimeScaleAttribute`

The media time scale. The value for this key is of type `NSNumber`, interpreted as a `long`.

This attribute can be read but not written. This attribute can be read but not written when the movie containing this media has been initialized with `QTMovieOpenForPlaybackAttribute` set to `YES`.

Available in Mac OS X v10.4 and later.

Declared in `QTMedia.h`.

`QTMediaTypeAttribute`

The media type. The value for this key is of type `NSString`. See "Media Types" (page 146) for the values this attribute can return.

This attribute can be read but not written. This attribute can be read but not written when the movie containing this media has been initialized with `QTMovieOpenForPlaybackAttribute` set to `YES`.

Available in Mac OS X v10.4 and later.

Declared in `QTMedia.h`.

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	QTAudioContextInsert QTAudioExtractionPanel QTKitPlayer QTKitTimeCode QTMetadataEditor

Overview

A `QTMovie` object is an object that represents a playable collection of media data.

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. It can also be used for other purposes, such as converting the media data into a different format.

The designated initializer for the `QTMovie` class is `initWithAttributes:error:`, whose first parameter is a dictionary of attribute keys and their desired values. One of these attributes must specify the location of the media data (for instance, using the `QTMovieURLAttribute` key). Other attributes may specify desired movie-opening behaviors, and others still may specify desired initial values of `QTMovie` properties (for instance, `QTMovieVolumeAttribute`).

There are two movie-opening behaviors. Specifying `QTMovieOpenForPlaybackAttribute` with the value `YES` indicates that the `QTMovie` object will be used only for playback, in which case QTKit may be able to use more efficient code paths for some media data. Specifying `QTMovieOpenAsyncRequiredAttribute` with the value `YES` indicates that all operations necessary to open the movie file (or other container) and to create a valid `QTMovie` object must occur asynchronously. In other words, `initWithAttributes:error:` will return almost immediately, performing any lengthy operations on another thread.

An exception, `QTDisallowedForInitializationPurposeException`, is raised whenever the client attempts to call a method that is not allowed under a requested movie-opening behavior. For example, if a `QTMovie` object is initialized with `QTMovieOpenForPlaybackAttribute` set to `YES`, then

`QTDisallowedForInitializationPurposeException` is raised if the client attempts to call methods that export the media data. 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 161)
Returns `YES` if the contents of the specified file can be used to initialize a `QTMovie` object.

`+ canInitWithURL:` (page 162)
Returns `YES` if the contents of the specified URL can be used to initialize a `QTMovie` object.

`+ canInitWithPasteboard:` (page 162)
Returns `YES` if the contents of the specified pasteboard can be used to initialize a `QTMovie` object.

`+ canInitWithDataReference:` (page 161)
Returns `YES` if the specified data reference can be used to initialize a `QTMovie` object.

`- initWithPasteboard:error:` (page 183)
Initializes a `QTMovie` object with the contents of the pasteboard specified by *`pasteboard`*.

Getting a List of Supported File Types

`+ movieFileTypes:` (page 164)
Returns an array of file types that can be opened as QuickTime movies.

`+ movieTypesWithOptions:` (page 165)
Returns an array of UTIs that QuickTime can open.

`+ movieUnfilteredFileTypes` (page 165)
Returns an array of file types that can be used to initialize a `QTMovie` object.

`+ movieUnfilteredPasteboardTypes` (page 165)
Returns an array of pasteboard types that can be used to initialize a `QTMovie` object.

Creating a Movie

`+ movie` (page 163)
Creates an empty `QTMovie` object.

`+ movieNamed:error:` (page 164)
Creates a `QTMovie` object initialized with the data from the QuickTime movie of the specified name in the application's bundle.

`+ movieWithData:error:` (page 166)
Creates a `QTMovie` object initialized with the data specified by *`data`*.

`+ movieWithURL:error:` (page 169)
Creates a `QTMovie` object initialized with the data in the URL specified by *`url`*.

+ `movieWithPasteboard:error:` (page 168)
  Creates a `QTMovie` object initialized with the contents of the pasteboard specified by *`pasteboard`*.
+ `movieWithFile:error:` (page 167)
  Creates a `QTMovie` object initialized with the data in the file specified by the name *`fileName`*.
+ `movieWithDataReference:error:` (page 167)
  Creates a `QTMovie` object intitalized with the data specified by the data reference *`dataReference`*.
+ `movieWithQuickTimeMovie:disposeWhenDone:error:` (page 168)
  Creates a `QTMovie` object initialized from an existing QuickTime movie *`movie`*.
+ `movieWithAttributes:error:` (page 166)
  Creates a `QTMovie` object initialized with the attributes specified in *`attributes`*.

Controlling Movie Playback

- `autoplay` (page 172)
  Sets a movie to start playing when a sufficient amount of media data is available.
- `play` (page 188)
  Plays the movie.
- `stop` (page 197)
  Stops the movie playing.
- `gotoBeginning` (page 177)
  Repositions the play position to the beginning of the movie.
- `gotoEnd` (page 178)
  Repositions the play position to the end of the movie.
- `gotoNextSelectionPoint` (page 178)
  Repositions the movie to the next selection point.
- `gotoPreviousSelectionPoint` (page 179)
  Repositions the movie to the previous selection point.
- `gotoPosterTime` (page 178)
  Repositions the play position to the movie's poster time.
- `setCurrentTime:` (page 193)
  Sets the movie's current time setting to `time`.
- `stepForward` (page 196)
  Sets the movie forward a single frame.
- `stepBackward` (page 196)
  Sets the movie backward a single frame.

Managing Threaded Operations of Movie Objects

+ `enterQTKitOnThread` (page 162)
  Performs any QuickTime-specific initialization for the current (non-main) thread; must be paired with a call to `exitQTKitOnThread`.
+ `enterQTKitOnThreadDisablingThreadSafetyProtection` (page 163)
  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 163)
  Performs any QuickTime-specific shut-down for the current (non-main) thread; must be paired with a call to `enterQTKitOnThread` or `enterQTKitOnThreadDisablingThreadSafetyProtection`.
- `attachToCurrentThread` (page 171)
  Attaches the receiver to the current thread; returns YES if successful, NO otherwise.
- `detachFromCurrentThread` (page 175)
  Detaches the receiver from the current thread; returns YES if successful, NO otherwise.

Initializing a QTMovie

- `initWithFile:error:` (page 182)
  Initializes a `QTMovie` object with the data in the file specified by the name *`fileName`*.
- `initWithURL:error:` (page 184)
  Initializes a `QTMovie` object with the data in the URL specified by *`url`*.
- `initWithData:error:` (page 181)
  Initializes a `QTMovie` object with the data specified by *`data`*.
- `initWithDataReference:error:` (page 181)
  Initializes a `QTMovie` object with the data reference setting specified by *`dataReference`*.
- `initWithMovie:timeRange:error:` (page 182)
  Initializes a `QTMovie` object with some or all of the data from an existing `QTMovie` object *`movie`*.
- `initWithQuickTimeMovie:disposeWhenDone:error:` (page 183)
  Initializes a `QTMovie` object with the data from an existing QuickTime movie *`movie`*.
- `initWithAttributes:error:` (page 180)
  Initializes a `QTMovie` object with the attributes specified in *`attributes`*.

Getting Information About a Movie and Its Chapters

- `hasChapters` (page 179)
  Returns YES if the receiver has chapters, NO otherwise.
- `chapterCount` (page 173)
  Returns the number of chapters in the receiver, or 0 if there are no chapters.
- `chapters` (page 173)
  Returns an NSArray containing information about the chapters in the receiver.
- `addChapters:withAttributes:error:` (page 169)
  Adds chapters to the receiver using the information specified in the chapters array.
- `removeChapters` (page 191)
  Removes any existing chapters from the receiver.
- `startTimeOfChapter:` (page 196)
  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 173)
  Returns the 0-based index of the chapter that contains the specified movie time.

Inspecting Movie Properties

- `duration` (page 175)
  Returns the duration of a `QTMovie` object as a structure of type `QTTime`.
- `currentTime` (page 174)
  Returns the current time of a `QTMovie` object as a structure of type `QTTime`.
- `rate` (page 190)
  Returns the current rate of a `QTMovie` object.
- `volume` (page 198)
  Returns the movie's volume as a scalar value of type `float`.
- `muted` (page 188)
  Returns the movie's mute setting.
- `movieWithTimeRange:error:` (page 188)
  Returns a `QTMovie` object whose data is the data in the specified time range.
- `attributeForKey:` (page 171)
  Returns the current value of the movie attribute *`attributeKey`*.
- `movieAttributes` (page 187)
  Returns a dictionary containing the current values of all defined movie attributes.

Managing QTMovie Idling States

- `setIdling:` (page 194)
  Sets the movie to idle `YES` or not to idle `NO`.
- `idling` (page 179)
  Returns the current idling state of a `QTMovie` object.

Setting QTMovie Properties

- `setRate:` (page 194)
  Sets the movie's rate to *`rate`*.
- `setVolume:` (page 195)
  Sets the movie's volume to *`volume`*.
- `setMuted:` (page 194)
  Sets the movie's mute setting to *`mute`*.

Returning QTMovie Time Frames

- `frameStartTime:` (page 177)
  Returns the start time of the frame that contains `atTime`, in the movie's time scale.
- `frameEndTime:` (page 176)
  Returns the end time of the frame that contains `atTime`, in the movie's time scale.
- `keyframeStartTime:` (page 186)
  Returns the start time of the keyframe that immediately precedes `atTime`, in the movie's time scale.

Setting Movie Attributes

- `setAttribute:forKey:` (page 192)
  Set the movie attribute *attributeKey* to the value specified by the *value* parameter.
- `setMovieAttributes:` (page 194)
  Set the movie attributes using the key-value pairs specified in the dictionary *attributes*.

Supporting Aperture Modes

- `generateApertureModeDimensions` (page 177)
  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 190)
  Removes aperture mode dimension information from a movie's tracks.

Getting and Setting Selection Times

- `selectionStart` (page 192)
  Returns the start time of the movie's current selection as a QTTime structure.
- `selectionEnd` (page 192)
  Returns the end point of the movie's current selection as a QTTime structure.
- `selectionDuration` (page 192)
  Returns the duration of the movie's current selection as a QTTime structure.
- `setSelection:` (page 195)
  Sets the movie's selection to *selection*.

Getting Movie Tracks

- `tracks` (page 197)
  Returns an array of QTTrack objects associated with the receiver.
- `tracksOfMediaType:` (page 197)
  Returns an array of tracks with the specified media type.

Getting Movie Images

- `posterImage` (page 189)
  Returns an `NSImage` for the poster frame of a `QTMovie`.
- `currentFrameImage` (page 174)
  Returns an `NSImage` for the frame at the current time in a `QTMovie`.
- `frameImageAtTime:` (page 176)
  Returns an NSImage for the frame at the time *time* in a `QTMovie`.

- `frameImageAtTime:withAttributes:error:` (page 176)
  Returns an `NSImage*`, `CIImage*`, `CGImageRef`, `CVPixelBufferRef`, or `CVOpenGLTextureRef` for the movie image at the specified time

Storing Movie Data

- `initToWritableDataReference:error:` (page 180)
  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.
- `invalidate` (page 186)
  Invalidates a `QTMovie` object immediately.
- `initToWritableFile:error:` (page 180)
  Useful for directly passing filenames and data objects. The `QTMovie` returned by this method is editable.
- `initToWritableData:error:` (page 179)
  Useful for directly passing filenames and data objects. The `QTMovie` returned by this method is editable.
- `movieFormatRepresentation` (page 187)
  Returns the movie's data in an NSData object.
- `writeToFile:withAttributes:` (page 199)
  Returns `YES` if the movie file was successfully created and `NO` otherwise.
- `writeToFile:withAttributes:error:` (page 199)
  Returns an NSError object if an error occurs and if errorPtr is non-NULL.

Editing a Movie

- `replaceSelectionWithSelectionFromMovie:` (page 191)
  Replaces the current selection in a `QTMovie` with the current selection in *`movie`*.
- `appendSelectionFromMovie:` (page 171)
  Appends to a `QTMovie` the current selection in *`movie`*.
- `insertSegmentOfMovie:timeRange:atTime:` (page 185)
  Inserts into a `QTMovie` at time *`time`* the selection in `movie` delimited by the time range *`range`*.
- `insertSegmentOfMovie:fromRange:scaledToRange:` (page 184)
  Inserts the specified segment from the movie into the receiver, scaled to the range `dstRange`.
- `insertEmptySegmentAt:` (page 184)
  inserts into a `QTMovie` an empty segment delimited by the range *`range`*.
- `deleteSegment:` (page 175)
  Deletes from a `QTMovie` the segment delimited by *`segment`*.
- `scaleSegment:newDuration:` (page 191)
  Scales the `QTMovie` segment delimited by the segment *`segment`* so that it will have the new duration *`newDuration`*.
- `insertSegmentOfTrack:timeRange:atTime:` (page 185)
  Inserts the specified segment of a `QTTrack` object into a `QTMovie`, at the specified time in the target `QTMovie`.

- `insertSegmentOfTrack:fromRange:scaledToRange:` (page 185)
  Inserts the specified segment of a `QTTrack` object into a `QTMovie`, scaling it as necessary to fit into the specified target range.
- `removeTrack:` (page 191)
  Removes a QTTrack from a movie.
- `addImage:forDuration:withAttributes:` (page 170)
  Adds an image for the specified duration to the receiver, using attributes specified in the attributes dictionary.

Saving a Movie

- `canUpdateMovieFile` (page 172)
  Indicates whether a movie file can be updated with changes made to the movie object.
- `updateMovieFile` (page 198)
  Updates the movie file of a `QTMovie`.

Getting QTMovie Primitives

- `quickTimeMovie` (page 189)
  Returns the QuickTime movie associated with a `QTMovie` object.
- `quickTimeMovieController` (page 190)
  Returns the QuickTime movie controller associated with a `QTMovie` object.

Getting and Setting QTMovie Delegates

- `delegate` (page 174)
  Returns the delegate of a `QTMovie` object.
- `setDelegate:` (page 193)
  Sets the movie's delegate to *`delegate`*.
- `externalMovie:` (page 200) *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).
- `movieShouldLoadData:` (page 187)
  If implemented by a delegate of a `QTMovie` object, called periodically while the movie is loading its data.
- `movieShouldTask:` (page 202) *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 201) *delegate method*
  If implemented, this method is called periodically during lengthy operations (such as exporting a movie).

- movie:linkToURL: (page 200) *delegate method*
  If implemented by a delegate of a QTMovie object, called to handle the movie controller actionmcActionLinkToURL.

Accessing QTMovie Visual Contexts

- setVisualContext: (page 195)
  Sets the visual context of the QTMovie.
- visualContext (page 198)
  Allows access to the visual context of the QTMovie.

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
```

Parameters
fileName
  An NSString object that specifies a full pathname to a file.

Return Value
YES if a QTMovie object can be initialized from the specified file, NO otherwise.

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

Related Sample Code
QTAudioContextInsert
QTKitAdvancedDocument
QTKitCreateMovie
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
```

Parameters

pasteboard
An NSPasteboard object.

Return Value
YES if a QTMovie object can be initialized from the specified pasteboard, NO otherwise.

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

Related Sample Code
iChatTheater

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
```

Parameters

url
An NSURL object.

Return Value
YES if a QTMovie object can be initialized from the specified URL, NO otherwise.

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.

Related Sample Code
QTAudioContextInsert

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
QTAudioContextInsert
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
QTAudioContextInsert
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). For more information, refer to "Constants For Use With movieFileTypes: Method" (page 202).

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

Related Sample Code
LiveVideoMixer
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
QuickTime 7.2.1 or 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
QTCoreVideo103
QTCoreVideo201
QTCoreVideo202
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.

Related Sample Code
GLUT

Declared In
QTMovie.h

movieWithAttributes:error:

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

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

Parameters

attributes

An NSDictionary object whose key-value pairs specify the attributes to use when initializing the movie.

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, for instance, QTMovieFileNameAttribute.
- Attributes that specify how the movie is to be instantiated, for instance, QTMovieOpenForPlaybackAttribute.
- Attributes that specify playback characteristics of the movie or other properties of the QTMovie object, for instance, QTMovieVolumeAttribute.

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

Related Sample Code
MyMediaPlayer

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
Movie Overlay
QTAudioExtractionPanel
QTKitCommandLine
QTKitMovieFrameImage
QTKitPlayer

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.

Related Sample Code
GLUT

Declared In
`QTMovie.h`

movieWithQuickTimeMovie:disposeWhenDone:error:

Creates a `QTMovie` object initialized from an existing QuickTime movie *`movie`*.

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

Discussion
This method cannot be called by 64-bit applications.

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
Movie Overlay
QTAudioContextInsert
QTKitCreateMovie
QTKitFrameStepper
QTKitPlayer

Declared In
QTMovie.h

Instance Methods

addChapters:withAttributes:error:

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

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

Parameters

chapters

An NSArray that contains one dictionary per chapter. The dictionary entries are:

- QTMovieChapterName, an NSString object that is the chapter name.
- QTMovieChapterStartTime, an NSValue object that wraps a QTTime structure that indicates the start time of the chapter.

attributes

An NSDictionary that contains settings for the new chapter track and its text. The following keys are currently recognized:

- `QTMovieChapterTargetTrackAttribute`, a `QTTrack` that is the target of the chapter track; if none is specified, use first video track in movie.
- `QTTrackDisplayNameAttribute`, an `NSString` that is the name of the chapter track; if none is specified, use "Chapter Track".
- `QTTrackTimeScaleAttribute`, an `NSNumber` that wraps a long; this is the time scale of the chapter track. If not present, the time scale of the target track is used.
- `QTTrackBoundsAttribute`, an `NSValue` that wraps an `NSRect` that specifies the desired position and size of the chapter track. The default width and height are those of the receiver `QTMovie` object.
- `QTTrackEnabledAttribute`, an `NSNumber` that wraps a `BOOL`; if `YES`, the chapter track is enabled, otherwise disabled (which is the default).
- `QTTrackLayerAttribute`, an `NSNumber` that wraps a short; this is the layer of the chapter track (default is -1).

errorPtr

A pointer to an `NSError` instance; if non-NULL, return any error in that location.

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.

Declared In

`QTMovie.h`

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
StillMotion
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 "Settable and Gettable Movie Attributes" (page 203) section.

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

Related Sample Code
QTAudioContextInsert
QTAudioExtractionPanel
QTKitImport
QTKitPlayer
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
QuickTime 7.2.1 or 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 176)
- posterImage (page 189)

Related Sample Code
QTKitFrameStepper

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.

Related Sample Code
CIVideoDemoGL
QTKitMovieFrameImage

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
```

Discussion
This method can be called when the movie has been initialized with `QTMovieOpenForPlaybackAttribute` set to `YES`.

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

Related Sample Code
QTKitCreateMovie
QTKitMovieShuffler
QTKitTimeCode
StillMotion

Declared In
`QTMovie.h`

frameEndTime:

Returns the end time of the frame that contains `atTime`, in the movie's time scale.

```
- (QTTime)frameEndTime:(QTTime)atTime
```

Availability
QuickTime 7.6.3 and later.

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 174)
- `posterImage` (page 189)

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 that contain these keys is described in "Dictionary of Frame Image Attributes " (page 216).

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.

Related Sample Code
QTKitMovieFrameImage

Declared In
`QTMovie.h`

frameStartTime:

Returns the start time of the frame that contains `atTime`, in the movie's time scale.

```
- (QTTime)frameStartTime:(QTTime)atTime
```

Availability
QuickTime 7.6.3 and later.

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.

Related Sample Code
CIColorTracking
CIVideoDemoGL

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`

gotoPosterTime

Repositions the play position to the movie's poster time.

```
- (void)gotoPosterTime
```

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.

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

Declared In
`QTMovie.h`

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.

Special Considerations

This method cannot be called when the movie has been initialized with `QTMovieOpenForPlaybackAttribute` set to `YES`.

Availability
QuickTime 7.2.1 or later.

Related Sample Code
StillMotion

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
```

Special Considerations

This method cannot be called when the movie has been initialized with `QTMovieOpenForPlaybackAttribute` set to `YES`.

Availability
QuickTime 7.2.1 or 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
```

Special Considerations

This method cannot be called when the movie has been initialized with `QTMovieOpenForPlaybackAttribute` set to `YES`.

Availability
QuickTime 7.2.1 or 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
```

Parameters

attributes

An `NSDictionary` object whose key-value pairs specify the attributes to use when initializing the movie.

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, for instance, `QTMovieFileNameAttribute`.
- Attributes that specify how the movie is to be instantiated, for instance, `QTMovieOpenForPlaybackAttribute`.
- Attributes that specify playback characteristics of the movie or other properties of the `QTMovie` object, for instance, `QTMovieVolumeAttribute`.

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
QTCoreVideo103
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
```

Parameters

`movie`
A QuickTime movie (of type `Movie`).

`dispose`
A `BOOL` value that indicates whether QTKit should call `DisposeMovie` on the specified QuickTime movie when the `QTMovie` object is deallocated. Passing `YES` effectively transfers ownership of the Movie to QTKit.

Discussion
This method cannot be called by 64-bit applications.

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.) Command-line tools that pass NO for the dispose 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.

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
StillMotion

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`

insertSegmentOfTrack:fromRange:scaledToRange:

Inserts the specified segment of a `QTTrack` object into a `QTMovie`, scaling it as necessary to fit into the specified target range.

```
- (QTTrack *)insertSegmentOfTrack:(QTTrack *)track
    fromRange:(QTTimeRange)srcRange
    scaledToRange:(QTTimeRange)dstRange
```

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

Declared In
`QTMovie.h`

insertSegmentOfTrack:timeRange:atTime:

Inserts the specified segment of a `QTTrack` object into a `QTMovie`, at the specified time in the target `QTMovie`.

```
- (QTTrack *)insertSegmentOfTrack:(QTTrack *)track
    timeRange:(QTTimeRange)range
    atTime:(QTTime)time
```

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

Declared In
QTMovie.h

invalidate

Invalidates a QTMovie object immediately.

```
- (void)invalidate
```

Discussion
By the time this method has returned, the receiver will have detached itself from any resources it is using, disposing of these resources when appropriate. Attempting to make any non-trivial use of the receiver after invalidating it will result in undefined behavior. This method does not release the receiver, so under retain-release memory management, release must still be called on the receiver for it to be fully deallocated. Because this method defeats sharing of QTMovie objects, it should only be called when it is known that the object is no longer needed.

Clients that pass NO for the dispose parameter must invalidate the QTMovie object (by calling -[QTMovie invalidate]) before calling DisposeMovie on the specified QuickTime movie. Failure to do this may result in a crash.

Special Considerations

This method can be called when the movie has been initialized with QTMovieOpenForPlaybackAttribute set to YES.

Availability
QuickTime 7.2.1 or later.

See Also
- initWithQuickTimeMovie:disposeWhenDone:error: (page 183)
+ movieWithQuickTimeMovie:disposeWhenDone:error: (page 168)

Declared In
QTMovie.h

keyframeStartTime:

Returns the start time of the keyframe that immediately precedes atTime, in the movie's time scale.

```
- (QTTime)keyframeStartTime:(QTTime)atTime
```

Availability
QuickTime 7.6.3 and later.

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 "Settable and Gettable Movie Attributes" (page 203) section.

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

Related Sample Code
CIColorTracking
CIVideoDemoGL

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 199)

Related Sample Code
QTAudioContextInsert
QTAudioExtractionPanel
QTKitImport
QTKitPlayer
QTMetadataEditor

Declared In
`QTMovie.h`

movieShouldLoadData:

If implemented by a delegate of a `QTMovie` object, called periodically while the movie is loading its data.

```
- (BOOL)movieShouldLoadData:(id)sender
```

Parameters
sender
The `QTMovie` object that is loading its data.

Return Value
A `BOOL` value; this value is ignored by QTKit.

Special Considerations

This delegate method is deprecated and should not be used in new code. This delegate method is not called when the movie has been initialized with `QTMovieOpenForPlaybackAttribute` set to `YES`.

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.

Special Considerations

This method cannot be called when the movie has been initialized with `QTMovieOpenForPlaybackAttribute` set to `YES`.

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
CIColorTracking
CIVideoDemoGL

QTKitMovieShuffler
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 174),
- frameImageAtTime: (page 176)

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 190)

Related Sample Code
QTAudioExtractionPanel
QTCoreVideo103
QTCoreVideo202
QTExtractAndConvertToAIFF
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 189)

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
QTQuartzPlayer

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`

removeTrack:

Removes a QTTrack from a movie.

```
- (void)removeTrack:(QTTrack *)track
```

Availability
Available in 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 "Settable and Gettable Movie Attributes" (page 203) section.

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

Related Sample Code
QTAudioExtractionPanel
QTKitImport
QTKitMovieShuffler
QTKitPlayer
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.

Related Sample Code
CIColorTracking
CIVideoDemoGL
StillMotion

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 "Settable and Gettable Movie Attributes" (page 203) 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
QTCoreImage101
QTCoreVideo101
QTCoreVideo103
QTCoreVideo202
QTQuartzPlayer

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`

setVisualContext:

Sets the visual context of the `QTMovie`.

```
- (void)setVisualContext:(QTVisualContextRef)visualContext
```

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

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.

Related Sample Code
CIVideoDemoGL

Declared In
QTMovie.h

stepForward

Sets the movie forward a single frame.

```
- (void)stepForward
```

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

Related Sample Code
CIVideoDemoGL
QTKitFrameStepper

Declared In
QTMovie.h

stop

Stops the movie playing.

```
- (void)stop
```

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

Related Sample Code
CIVideoDemoGL
MyMediaPlayer
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
QTAudioContextInsert
QTAudioExtractionPanel
QTKitPlayer
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
QTAudioContextInsert
QTKitCommandLine
QTKitPlayer
QTMetadataEditor
WritableFileDemo

Declared In
`QTMovie.h`

visualContext

Allows access to the visual context of the `QTMovie`.

```
- (QTVisualContextRef)visualContext
```

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

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 *)fileName
    withAttributes
    :(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
QuickTime 7.2.1 or later.

See Also
- movieFormatRepresentation (page 187)

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 187)

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
```

Parameters

dictionary
An NSDictionary object that contains information about the desired external movie.

Return Value
A QTMovie object.

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.

This delegate method is not called when the movie has been initialized with QTMovieOpenForPlaybackAttribute set to YES.

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

Declared In
QTMovie.h

movie:linkToURL:

If implemented by a delegate of a QTMovie object, called to handle the movie controller actionmcActionLinkToURL.

```
- (BOOL)movie:(QTMovie *)movielinkToURL
    :(NSURL *)url
```

Parameters

movie
A QTMovie object.

url

An NSURL object.

Return Value
A BOOL value; a delegate should return YES if it handled this method, NO otherwise.

Discussion
QTMovie objects can contain requests to open URLs. An application can implement this delegate method to override the default URL-opening mechanism in QTKit. In general, most applications will not need to install a delegate to handle this. This delegate method is not called when the movie has been initialized with QTMovieOpenForPlaybackAttribute set to YES.

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
```

Parameters

op

An NSString object that is a localized description of the operation being performed.

phase

A value of type QTMovieOperationPhase that indicates indicates whether the operation is just beginning (QTMovieOperationBeginPhase), ending (QTMovieOperationEndPhase), or is at a certain percentage of completion (QTMovieOperationUpdatePercentPhase).

percent

When the phase parameter is QTMovieOperationUpdatePercentPhase, the approximate percentage of the operation completed.

attributes

An NSDictionary object that the same dictionary passed to a QTMovie method that caused the lengthy operation (for example, the attributes dictionary passed to writeToFile:withAttributes:error:). This parameter may be nil.

Return Value
A BOOL value; a delegate should return YES to continue the lengthy operation, NO to cancel it.

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
}
```

Special Considerations

This delegate method is not called when the movie has been initialized with `QTMovieOpenForPlaybackAttribute` set to `YES`.

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
```

Parameters

movie

The `QTMovie` object that is about to perform idle processing.

Return Value
A `BOOL` value; a delegate should return `YES` to cancel the standard movie idle processing, `NO` otherwise.

Discussion
The delegate can cancel that normal processing by returning `YES`.

Special Considerations

This delegate method is deprecated and should not be used in new code. This delegate method is not called when the movie has been initialized with `QTMovieOpenForPlaybackAttribute` set to `YES`.

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

Declared In
`QTMovie.h`

Constants

Constants For Use With movieFileTypes: Method

The following values can be used to include some or all of the file types that are normally excluded:

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

Constants

`QTIncludeStillImageTypes`
This value adds to the array all file types for still images that can be opened using a graphics importer.

Available in Mac OS X v10.3 and later.

Declared in `QTMovie.h`.

`QTIncludeTranslatableTypes`
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.

Available in Mac OS X v10.3 and later.

Declared in `QTMovie.h`.

`QTIncludeAggressiveTypes`
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).

Available in Mac OS X v10.3 and later.

Declared in `QTMovie.h`.

`QTIncludeCommonTypes`
This value adds to the array all common file types that QuickTime can open in place on the current system.

Available in Mac OS X v10.3 and later.

Declared in `QTMovie.h`.

`QTIncludeAllTypes`
This value adds to the array all file types that QuickTime can open on the current system, using any available movie or graphics importer.

Available in Mac OS X v10.3 and later.

Declared in `QTMovie.h`.

Settable and Gettable Movie Attributes

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`.

```
NSString * const QTMovieApertureModeAttribute;
NSString * const QTMovieActiveSegmentAttribute;
NSString * const QTMovieAutoAlternatesAttribute;
NSString * const QTMovieCopyrightAttribute;
NSString * const QTMovieCreationTimeAttribute;
NSString * const QTMovieCurrentSizeAttribute;
NSString * const QTMovieCurrentTimeAttribute;
NSString * const QTMovieDataSizeAttribute;
NSString * const QTMovieDelegateAttribute;
NSString * const QTMovieDisplayNameAttribute;
NSString * const QTMovieDurationAttribute;
NSString * const QTMovieEditableAttribute;
NSString * const QTMovieFileNameAttribute;
NSString * const QTMovieHasApertureModeDimensionsAttribute;
NSString * const QTMovieHasAudioAttribute;
NSString * const QTMovieHasDurationAttribute;
NSString * const QTMovieHasVideoAttribute;
NSString * const QTMovieIsActiveAttribute;
NSString * const QTMovieIsInteractiveAttribute;
NSString * const QTMovieIsLinearAttribute;
NSString * const QTMovieIsSteppableAttribute;
NSString * const QTMovieLoadStateAttribute;
NSString * const QTMovieLoadStateErrorAttribute;
NSString * const QTMovieLoopsAttribute;
NSString * const QTMovieLoopsBackAndForthAttribute;
NSString * const QTMovieModificationTimeAttribute;
NSString * const QTMovieMutedAttribute;
NSString * const QTMovieNaturalSizeAttribute;
NSString * const QTMoviePlaysAllFramesAttribute;
NSString * const QTMoviePlaysSelectionOnlyAttribute;
NSString * const QTMoviePosterTimeAttribute;
NSString * const QTMoviePreferredMutedAttribute;
NSString * const QTMoviePreferredRateAttribute;
NSString * const QTMoviePreferredVolumeAttribute;
NSString * const QTMoviePreviewModeAttribute;
NSString * const QTMoviePreviewRangeAttribute;
NSString * const QTMovieRateAttribute;
NSString * const QTMovieSelectionAttribute;
NSString * const QTMovieTimeScaleAttribute;
NSString * const QTMovieURLAttribute;
NSString * const QTMovieVolumeAttribute;
NSString * const QTMovieRateChangesPreservePitchAttribute;
```

Constants

`QTMovieApertureModeAttribute`

Sets the aperture mode attribute on a `QTMovie` object to indicate whether aspect ratio and clean aperture correction should be performed.

This attribute can be read but not written. This attribute can be read but not written when the movie has been initialized with `QTMovieOpenForPlaybackAttribute` set to `YES`.

Available in Mac OS X v10.5 and later.

Declared in `QTMovie.h`.

`QTMovieActiveSegmentAttribute`
The active segment of a `QTMovie` object; the value for this key is of type `NSValue`, interpreted as a `QTTimeRange` structure. This constant is available in Mac OS X 10.4 and later, but deprecated in Mac OS X 10.5.

This attribute can be read but not written. This attribute can be read but not written when the movie has been initialized with `QTMovieOpenForPlaybackAttribute` set to `YES`.

Available in Mac OS X v10.4 and later.

Declared in `QTMovie.h`.

`QTMovieAutoAlternatesAttribute`
The auto-alternate state of a `QTMovie` object. The value for this key is of type `NSNumber`, interpreted as a `BOOL`.

This attribute can be read but not written. This attribute can be read but not written when the movie has been initialized with `QTMovieOpenForPlaybackAttribute` set to `YES`.

Available in Mac OS X v10.4 and later.

Declared in `QTMovie.h`.

`QTMovieCopyrightAttribute`
The copyright string of a `QTMovie` object; the value for this key is of type `NSString`.

This attribute can be read but not written. This attribute can be read but not written when the movie has been initialized with `QTMovieOpenForPlaybackAttribute` set to `YES`.

Available in Mac OS X v10.4 and later.

Declared in `QTMovie.h`.

`QTMovieCreationTimeAttribute`
The creation time of a `QTMovie` object; the value for this key is of type `NSDate`.

This attribute can be read but not written. This attribute can be read but not written when the movie has been initialized with `QTMovieOpenForPlaybackAttribute` set to `YES`.

Available in Mac OS X v10.4 and later.

Declared in `QTMovie.h`.

`QTMovieCurrentSizeAttribute`
The current size of a `QTMovie` object; the value for this key is of type `NSValue`, interpreted as an `NSSize` structure.

This attribute can be read and written. This attribute cannot be read or written when the movie has been initialized with `QTMovieOpenForPlaybackAttribute` set to `YES`. This attribute is deprecated in QTKit version 7.6 and later.

Available in Mac OS X v10.4 and later.

Declared in `QTMovie.h`.

`QTMovieCurrentTimeAttribute`
The current time of a `QTMovie` object; the value for this key is of type `NSValue`, interpreted as a QTTime structure.

This attribute can be read and written. This attribute can be read and written when the movie has been initialized with `QTMovieOpenForPlaybackAttribute` set to `YES`.

Available in Mac OS X v10.4 and later.

Declared in `QTMovie.h`.

`QTMovieDataSizeAttribute`
The data size of a `QTMovie`. The value for this key is of type `NSNumber`, which is interpreted as a `longlong`.

This attribute can be read but not written. This attribute can be read but not written when the movie has been initialized with `QTMovieOpenForPlaybackAttribute` set to `YES`.

Available in Mac OS X v10.4 and later.

Declared in `QTMovie.h`.

`QTMovieDelegateAttribute`
The delegate for a `QTMovie` object. The value for this key is of type `NSObject`.

This attribute can be read and written. This attribute can be read and written when the movie has been initialized with `QTMovieOpenForPlaybackAttribute` set to `YES`.

Available in Mac OS X v10.4 and later.

Declared in `QTMovie.h`.

`QTMovieDisplayNameAttribute`
The display name of a `QTMovie` object. A display name is stored as user data in a movie file and hence may differ from the base name of the movie's filename or URL. The value for this key is of type `NSString`.

This attribute can be read and written. This attribute can be read but not written when the movie has been initialized with `QTMovieOpenForPlaybackAttribute` set to `YES`.

Available in Mac OS X v10.4 and later.

Declared in `QTMovie.h`.

`QTMovieDontInteractWithUserAttribute`
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`.

Available in Mac OS X v10.4 and later.

Declared in `QTMovie.h`.

`QTMovieDurationAttribute`
The duration of a `QTMovie` object; the value for this key is of type `NSValue`, interpreted as a `QTTime` structure.

Available in Mac OS X v10.4 and later.

Declared in `QTMovie.h`.

`QTMovieEditableAttribute`
The editable setting; the value for this key is of type `NSNumber`, interpreted as a `BOOL`. This value is `YES` if the movie can be edited.

This attribute can be read and written. This attribute can be read but not written when the movie has been initialized with `QTMovieOpenForPlaybackAttribute` set to `YES`.

Available in Mac OS X v10.4 and later.

Declared in `QTMovie.h`.

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

This attribute can be read but not written. This attribute can be read but not written when the movie has been initialized with `QTMovieOpenForPlaybackAttribute` set to `YES`.

Available in Mac OS X v10.4 and later.

Declared in `QTMovie.h`.

`QTMovieHasApertureModeDimensionsAttribute`

The aperture mode dimensions set on any track in this `QTMovie` object, even if those dimemsions are all identical to the classic dimensions (as is the case for content with square pixels and no edge-processing region). The value for this key is of type `NSNumber`, interpreted as a `BOOL`.

This attribute can be read but not written. This attribute can be read but not written when the movie has been initialized with `QTMovieOpenForPlaybackAttribute` set to `YES`.

Available in Mac OS X v10.5 and later.

Declared in `QTMovie.h`.

`QTMovieHasAudioAttribute`

The audio data setting; the value for this key is of type `NSNumber`, interpreted as a `BOOL`. This value is `YES` if the movie contains audio data.

This attribute can be read but not written. This attribute can be read but not written when the movie has been initialized with `QTMovieOpenForPlaybackAttribute` set to `YES`.

Available in Mac OS X v10.4 and later.

Declared in `QTMovie.h`.

`QTMovieHasDurationAttribute`

The duration setting; the value for this key is of type `NSNumber`, interpreted as a `BOOL`. This value is `YES` if the movie has a duration. (Some types of movies, for instance QuickTime VR movies, have no duration.)

This attribute can be read but not written. This attribute can be read but not written when the movie has been initialized with `QTMovieOpenForPlaybackAttribute` set to `YES`.

Available in Mac OS X v10.4 and later.

Declared in `QTMovie.h`.

`QTMovieHasVideoAttribute`

The video data setting; the value for this key is of type `NSNumber`, interpreted as a `BOOL`. This value is `YES` if the movie contains video data.

This attribute can be read but not written. This attribute can be read but not written when the movie has been initialized with `QTMovieOpenForPlaybackAttribute` set to `YES`.

Available in Mac OS X v10.4 and later.

Declared in `QTMovie.h`.

`QTMovieIsActiveAttribute`

The active setting; the value for this key is of type `NSNumber`, interpreted as a `BOOL`.

This attribute can be read and written. This attribute can be read and written when the movie has been initialized with `QTMovieOpenForPlaybackAttribute` set to `YES`.

Available in Mac OS X v10.4 and later.

Declared in `QTMovie.h`.

`QTMovieIsInteractiveAttribute`

The interactive setting; the value for this key is of type `NSNumber`, interpreted as a `BOOL`. This value is `YES` if the movie is interactive.

This attribute can be read but not written. This attribute can be read but not written when the movie has been initialized with `QTMovieOpenForPlaybackAttribute` set to `YES`.

Available in Mac OS X v10.4 and later.

Declared in `QTMovie.h`.

`QTMovieIsLinearAttribute`
The linear setting; the value for this key is of type `NSNumber`, interpreted as a `BOOL`. This value is `YES` if the movie is linear, as opposed to a non-linear QuickTime VR movie.

This attribute can be read but not written. This attribute can be read but not written when the movie has been initialized with `QTMovieOpenForPlaybackAttribute` set to `YES`.

Available in Mac OS X v10.4 and later.

Declared in `QTMovie.h`.

`QTMovieIsSteppableAttribute`
The steppable setting; the value for this key is of type `NSNumber`, interpreted as a `BOOL`. This value is `YES` if the movie can step from frame to frame.

This attribute can be read but not written. This attribute can be read but not written when the movie has been initialized with `QTMovieOpenForPlaybackAttribute` set to `YES`.

Available in Mac OS X v10.4 and later.

Declared in `QTMovie.h`.

`QTMovieLoadStateAttribute`
The load state value; the value for this key is of type `NSNumber`, interpreted as a `long`.

This attribute can be read but not written. This attribute can be read but not written when the movie has been initialized with `QTMovieOpenForPlaybackAttribute` set to `YES`.

Declared in `QTMovie.h`.

Mac OS X v10.5 and later.

`QTMovieLoadStateErrorAttribute`
The load state error of a `QTMovie` object; the value for this key is of type `NSError`.

This attribute can be read but not written. This attribute can be read but not written when the movie has been initialized with `QTMovieOpenForPlaybackAttribute` set to `YES`.

Declared in `QTMovie.h`.

QuickTime 7.6.3 and later.

`QTMovieLoopsAttribute`
The looping setting; the value for this key is of type `NSNumber`, interpreted as a `BOOL`. This value is `YES` if the movie is set to loop, `NO` otherwise.

This attribute can be read and written. This attribute can be read and written when the movie has been initialized with `QTMovieOpenForPlaybackAttribute` set to `YES`.

Available in Mac OS X v10.4 and later.

Declared in `QTMovie.h`.

`QTMovieLoopsBackAndForthAttribute`
The palindrome looping setting; the value for this key is of type `NSNumber`, interpreted as a `BOOL`. This value is `YES` if the movie is set to loop back and forth. Note that `QTMovieLoopsAttribute` and `QTMovieLoopsBackAndForthAttribute` are independent and indeed exclusive. `QTMovieLoopsAttribute` is used to get and set the state of normal looping; `QTMovieLoopsBackAndForthAttribute` is used to get and set the state of palindrome looping.

This attribute can be read and written. This attribute can be read and written when the movie has been initialized with `QTMovieOpenForPlaybackAttribute` set to `YES`.

Available in Mac OS X v10.4 and later.

Declared in `QTMovie.h`.

`QTMovieModificationTimeAttribute`
The modification time of a `QTMovie` object; the value for this key is of type `NSDate`.

This attribute can be read but not written. This attribute can be read but not written when the movie has been initialized with `QTMovieOpenForPlaybackAttribute` set to `YES`.

Available in Mac OS X v10.4 and later.

Declared in `QTMovie.h`.

`QTMovieMutedAttribute`
The mute setting; the value for this key is of type `NSNumber`, interpreted as a `BOOL`. This value is `YES` if the movie volume is muted.

This attribute can be read and written. This attribute can be read and written when the movie has been initialized with `QTMovieOpenForPlaybackAttribute` set to `YES`.

Available in Mac OS X v10.4 and later.

Declared in `QTMovie.h`.

`QTMovieNaturalSizeAttribute`
The natural size of a `QTMovie` object; the value for this key is of type `NSValue`, interpreted as an `NSSize` structure.

This attribute can be read but not written. This attribute can be read but not written when the movie has been initialized with `QTMovieOpenForPlaybackAttribute` set to `YES`.

Available in Mac OS X v10.4 and later.

Declared in `QTMovie.h`.

`QTMoviePlaysAllFramesAttribute`
The play-all-frames setting; the value for this key is of type `NSNumber`, interpreted as a `BOOL`. This value is `YES` if the movie will play all frames.

This attribute can be read and written. This attribute can be read and written when the movie has been initialized with `QTMovieOpenForPlaybackAttribute` set to `YES`.

Available in Mac OS X v10.4 and later.

Declared in `QTMovie.h`.

`QTMoviePlaysSelectionOnlyAttribute`
The play-selection setting; the value for this key is of type `NSNumber`, interpreted as a `BOOL`. This value is `YES` if the movie will play only the current selection.

This attribute can be read and written. This attribute can be read and written when the movie has been initialized with `QTMovieOpenForPlaybackAttribute` set to `YES`.

Available in Mac OS X v10.4 and later.

Declared in `QTMovie.h`.

`QTMoviePosterTimeAttribute`
The movie poster time of a `QTMovie` object; the value for this key is of type `NSValue`, interpreted as a `QTTime` structure.

This attribute can be read and written. This attribute can be read and written when the movie has been initialized with `QTMovieOpenForPlaybackAttribute` set to `YES`.

Available in Mac OS X v10.4 and later.

Declared in `QTMovie.h`.

`QTMoviePreferredMutedAttribute`
The preferred mute setting; the value for this key is of type `NSNumber`, interpreted as a `BOOL`. This value is `YES` if the movie preferred mute setting is muted.

This attribute can be read and written. This attribute can be read but not written when the movie has been initialized with `QTMovieOpenForPlaybackAttribute` set to `YES`.

Available in Mac OS X v10.4 and later.

Declared in `QTMovie.h`.

`QTMoviePreferredRateAttribute`
The preferred rate; the value for this key is of type `NSNumber`, interpreted as a `float`.

This attribute can be read and written. This attribute can be read but not written when the movie has been initialized with `QTMovieOpenForPlaybackAttribute` set to `YES`.

Available in Mac OS X v10.4 and later.

Declared in `QTMovie.h`.

`QTMoviePreferredVolumeAttribute`
The preferred volume; the value for this key is of type `NSNumber`, interpreted as a `float`.

This attribute can be read and written. This attribute can be read but not written when the movie has been initialized with `QTMovieOpenForPlaybackAttribute` set to `YES`.

Available in Mac OS X v10.4 and later.

Declared in `QTMovie.h`.

`QTMoviePreviewModeAttribute`
The preview mode setting; the value for this key is of type `NSNumber`, interpreted as a `BOOL`. This value is `YES` if the movie is in preview mode.

This attribute can be read and written. This attribute can be read but not written when the movie has been initialized with `QTMovieOpenForPlaybackAttribute` set to `YES`.

Available in Mac OS X v10.4 and later.

Declared in `QTMovie.h`.

`QTMoviePreviewRangeAttribute`
The preview range of a `QTMovie` object; the value for this key is of type `NSValue`, interpreted as a `QTTimeRange` structure.

This attribute can be read and written. This attribute can be read but not written when the movie has been initialized with `QTMovieOpenForPlaybackAttribute` set to `YES`.

Available in Mac OS X v10.4 and later.

Declared in `QTMovie.h`.

`QTMovieRateAttribute`
The movie rate; the value for this key is of type `NSNumber`, interpreted as a `float`.

This attribute can be read and written. This attribute can be read and written when the movie has been initialized with `QTMovieOpenForPlaybackAttribute` set to `YES`.

Available in Mac OS X v10.4 and later.

Declared in `QTMovie.h`.

`QTMovieRateChangesPreservePitchAttribute`

When the playback rate is not unity, audio must be resampled in order to play at the new rate. The default resampling affects the pitch of the audio (for example, playing at 2x speed raises the pitch by an octave, 1/2x lowers an octave). If this property is set on the movie, an alternative algorithm is used, which alters the speed without changing the pitch. Since this is more computationally expensive, this property may be silently ignored on some slow CPUs.

This attribute can be read but not written; it must be among the initialization attributes to have any effect. This attribute can be read but not written when the movie has been initialized with `QTMovieOpenForPlaybackAttribute` set to `YES`.

Available in Mac OS X v10.4 and later.

Declared in `QTMovie.h`.

`QTMovieSelectionAttribute`

The selection range of a `QTMovie` object; the value for this key is of type `NSValue`, interpreted as a `QTTimeRange` structure.

This attribute can be read and written. This attribute cannot be read or written when the movie has been initialized with `QTMovieOpenForPlaybackAttribute` set to `YES`.

Available in Mac OS X v10.4 and later.

Declared in `QTMovie.h`.

`QTMovieTimeScaleAttribute`

The time scale of a `QTMovie` object; the value for this key is of type `NSNumber`, interpreted as a `long`. This attribute can be read and (in Mac OS X 10.5 and later) written; in earlier versions of Mac OS X, this attribute is readable only. In general, you should set this attribute only on newly-created movies or on movies that have not been edited. Also, you should only increase the time scale value, and you should try to use integer multiples of the existing time scale.

This attribute can be read but not written when the movie has been initialized with `QTMovieOpenForPlaybackAttribute` set to `YES`.

Available in Mac OS X v10.4 and later.

Declared in `QTMovie.h`.

`QTMovieURLAttribute`

The URL of a `QTMovie` object; the value for this key is of type `NSURL`.

This attribute can be read but not written. This attribute can be read but not written when the movie has been initialized with `QTMovieOpenForPlaybackAttribute` set to `YES`.

Available in Mac OS X v10.4 and later.

Declared in `QTMovie.h`.

`QTMovieVolumeAttribute`

The movie volume; the value for this key is of type `NSNumber`, interpreted as a `float`.

This attribute can be read and written. This attribute can be read and written when the movie has been initialized with `QTMovieOpenForPlaybackAttribute` set to `YES`.

Available in Mac OS X v10.4 and later.

Declared in `QTMovie.h`.

Aperture Modes

When a movie is in clean, production, or encoded pixels aperture mode, each track's dimensions are overridden by special dimensions for that mode. The original track dimensions are preserved and can be restored by setting the movie into classic aperture mode. Aperture modes are not saved in movies. The associated value is of type `NSString` and is assumed to be one of the following strings:

```
NSString * const QTMovieApertureModeClassic;
NSString * const QTMovieApertureModeClean;
NSString * const QTMovieApertureModeProduction;
NSString * const QTMovieApertureModeEncodedPixels;
```

Constants

`QTMovieApertureModeClassic`

No aspect ratio or clean aperture correction is performed. This is the default aperture mode and provides compatibility with behavior in QuickTime 7.0.x and earlier. If you call `-[QTTrack setDimensions]`, the movie is automatically switched to classic mode.

Available in Mac OS X v10.5 and later.

Declared in `QTMovie.h`.

`QTMovieApertureModeClean`

An aperture mode for general display. Where possible, video will be displayed at the correct pixel aspect ratio, trimmed to the clean aperture. A movie in clean aperture mode sets each track's dimensions to match the size returned by `-[QTTrack apertureModeDimensionsForMode:QTMovieApertureModeClean]`.

Available in Mac OS X v10.5 and later.

Declared in `QTMovie.h`.

`QTMovieApertureModeProduction`

An aperture mode for modal use in authoring applications. Where possible, video will be displayed at the correct pixel aspect ratio, but without trimming to the clean aperture so that the edge processing region can be viewed. A movie in production aperture mode sets each track's dimensions to match the size returned by `-[QTTrack apertureModeDimensionsForMode:QTMovieApertureModeProduction]`.

Available in Mac OS X v10.5 and later.

Declared in `QTMovie.h`.

`QTMovieApertureModeEncodedPixels`

An aperture mode for technical use. Displays all encoded pixels with no aspect ratio or clean aperture compensation. A movie in encoded pixels aperture mode sets each track's dimensions to match the size returned by `-[QTTrack apertureModeDimensionsForMode:QTMovieApertureModeEncodedPixels]`.

Available in Mac OS X v10.5 and later.

Declared in `QTMovie.h`.

Movie Load State Values

The movie load state values. The `attributeForKey:QTMovieLoadStateAttribute` returns an `NSNumber` that wraps a long integer; the enumerated constants shown here are the possible values of that long integer.

```
enum {
  QTMovieLoadStateError = -1L,
  QTMovieLoadStateLoading = 1000,
  QTMovieLoadStateLoaded = 2000,
  QTMovieLoadStatePlayable = 10000,
  QTMovieLoadStatePlaythroughOK = 20000,
  QTMovieLoadStateComplete = 100000L
};
typedef NSInteger QTMovieLoadState;
```

Constants

`QTMovieLoadStateError`
An error occurred while loading the movie.
Available in Mac OS X v10.5 and later.
Declared in `QTMovie.h`.

`QTMovieLoadStateLoading`
The movie is loading.
Available in Mac OS X v10.5 and later.
Declared in `QTMovie.h`.

`QTMovieLoadStateLoaded`
The movie atom has loaded; it's safe to query movie properties.
Available in Mac OS X v10.5 and later.
Declared in `QTMovie.h`.

`QTMovieLoadStatePlayable`
The movie has loaded enough media data to begin playing.
Available in Mac OS X v10.5 and later.
Declared in `QTMovie.h`.

`QTMovieLoadStatePlaythroughOK`
The movie has loaded enough media data to play through to the end.
Available in Mac OS X v10.5 and later.
Declared in `QTMovie.h`.

`QTMovieLoadStateComplete`
The movie has loaded completely.
Mac OS X v10.5 and later.
Declared in `QTMovie.h`.

Dictionary Items Passed to QTMovie Notifications

The following constants specify items in dictionaries passed to `QTMovie` notifications and delegate methods.

```
NSString * const QTMovieMessageNotificationParameter;
NSString * const QTMovieRateDidChangeNotificationParameter;
NSString * const QTMovieStatusFlagsNotificationParameter;
NSString * const QTMovieStatusCodeNotificationParameter;
NSString * const QTMovieStatusStringNotificationParameter;
NSString * const QTMovieTargetIDNotificationParameter;
NSString * const QTMovieTargetNameNotificationParameter;
```

Constants

`QTMovieMessageNotificationParameter`

Used as a key in the userInfo dictionary passed to the `QTMovieMessageNotification` notification to indicate the message. The associated value is an `NSString`.

Available in Mac OS X v10.4 and later.

Declared in `QTMovie.h`.

`QTMovieRateDidChangeNotificationParameter`

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`.

Available in Mac OS X v10.4 and later.

Declared in `QTMovie.h`.

`QTMovieStatusFlagsNotificationParameter`

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`.

Available in Mac OS X v10.4 and later.

Declared in `QTMovie.h`.

`QTMovieStatusCodeNotificationParameter`

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`.

Available in Mac OS X v10.4 and later.

Declared in `QTMovie.h`.

`QTMovieStatusStringNotificationParameter`

Used as a key in the userInfo dictionary passed to the `QTMovieStatusStringPostedNotification` notification to indicate a status string.

Available in Mac OS X v10.4 and later.

Declared in `QTMovie.h`.

`QTMovieTargetIDNotificationParameter`

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.

Available in Mac OS X v10.4 and later.

Declared in `QTMovie.h`.

`QTMovieTargetNameNotificationParameter`

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.

Available in Mac OS X v10.4 and later.

Declared in `QTMovie.h`.

Dictionary Keys For Movie Export

The following constants are dictionary keys that you can use to specify movie attributes, using the `writeToFile` method.

```
NSString * const QTMovieExport;
NSString * const QTMovieExportType;
NSString * const QTMovieFlatten;
NSString * const QTMovieExportSettings;
NSString * const QTMovieExportManufacturer;
```

Constants

`QTMovieExport`

The movie export setting; the value for this key is of type `NSNumber`, interpreted as a `BOOL`.

Available in Mac OS X v10.4 and later.

Declared in `QTMovie.h`.

`QTMovieExportType`

The movie export type; the value for this key is of type `NSNumber`, interpreted as a `long`.

Available in Mac OS X v10.4 and later.

Declared in `QTMovie.h`.

`QTMovieFlatten`

The movie flatten setting; the value for this key is of type `NSNumber`, interpreted as a `BOOL`.

Available in Mac OS X v10.4 and later.

Declared in `QTMovie.h`.

`QTMovieExportSettings`

The movie export settings; the value of this key is of type `NSData`, interpreted as a `QTAtomContainer`.

Available in Mac OS X v10.4 and later.

Declared in `QTMovie.h`.

`QTMovieExportManufacturer`

The export manufacturer value; the value for this key is of type `NSNumber`, interpreted as a `long`.

Available in Mac OS X v10.4 and later.

Declared in `QTMovie.h`.

Dictionary Keys For Image Codecs

The following constants are dictionary keys that you can use to specify movie attributes, using the `addImage` method.

```
NSString * const QTAddImageCodecType;
NSString * const QTAddImageCodecQuality;
```

Constants

`QTAddImageCodecType`

The image codec string; the value for this key is of type `NSString`.

Available in Mac OS X v10.4 and later.

Declared in `QTMovie.h`.

`QTAddImageCodecQuality`
The image codec value; the value for this key is of type `NSNumber`.

Available in Mac OS X v10.4 and later.

Declared in `QTMovie.h`.

Dictionary of Frame Image Attributes

The following is a dictionary of attributes that can contain frame image keys, using the `frameImageAtTime:withAttributes:error:` method.

```
NSString * const QTMovieFrameImageSize;
NSString * const QTMovieFrameImageType;
NSString * const QTMovieFrameImageTypeNSImage;
NSString * const QTMovieFrameImageTypeCGImageRef;
NSString * const QTMovieFrameImageTypeCIImage;
NSString * const QTMovieFrameImageTypeCVPixelBufferRef;
NSString * const QTMovieFrameImageTypeCVOpenGLTextureRef;
NSString * const QTMovieFrameImageRepresentationsType;
NSString * const QTMovieFrameImageOpenGLContext;
NSString * const QTMovieFrameImagePixelFormat;
NSString * const QTMovieFrameImageInterlaced;
NSString * const QTMovieFrameImageHighQuality;
NSString * const QTMovieFrameImageSingleField;
NSString * const QTMovieFrameImageSessionMode;
```

Constants

`QTMovieFrameImageSize`
Size of the image. The value is an `NSValue` containing an `NSSize` record. The default image size is the current movie size.

Available in Mac OS X v10.5 and later.

Declared in `QTMovie.h`.

`QTMovieFrameImageType`
Type of the image. The value is an `NSString`. The default image type is NSImage.

Available in Mac OS X v10.5 and later.

Declared in `QTMovie.h`.

`QTMovieFrameImageTypeNSImage`
A value for the `QTMovieFrameImageType` key of the `QTMovie frameImageAtTime:withAttributes:error:` attributes dictionary. Specifies that the type of image returned should be an `NSImage`.

Available in Mac OS X v10.5 and later.

Declared in `QTMovie.h`.

`QTMovieFrameImageTypeCGImageRef`
A value for the `QTMovieFrameImageType` key of the `QTMovie frameImageAtTime:withAttributes:error:` attributes dictionary. Specifies that the type of image returned should be a `CGImageRef`.

Available in Mac OS X v10.5 and later.

Declared in `QTMovie.h`.

`QTMovieFrameImageTypeCIImage`
A value for the `QTMovieFrameImageType` key of the `QTMovie frameImageAtTime:withAttributes:error:` attributes dictionary. Specifies that the type of image returned should be a `CIImage`.

Available in Mac OS X v10.5 and later.

Declared in `QTMovie.h`.

`QTMovieFrameImageTypeCVPixelBufferRef`
A value for the `QTMovieFrameImageType` key of the `QTMovie frameImageAtTime:withAttributes:error:` attributes dictionary. Specifies that the type of image returned should be a `CVPixelBufferRef`.

Available in Mac OS X v10.5 and later.

Declared in `QTMovie.h`.

`QTMovieFrameImageTypeCVOpenGLTextureRef`
A value for the `QTMovieFrameImageType` key of the `QTMovie frameImageAtTime:withAttributes:error:` attributes dictionary. Specifies that the type of image returned should be a `CVOpenGLTextureRef`. Clients that specify this attribute must also specify the OpenGL context and pixel format for the texture using the `QTMovieFrameImageOpenGLContext` and `QTMovieFrameImagePixelFormat` attribute keys.

Available in Mac OS X v10.5 and later.

Declared in `QTMovie.h`.

`QTMovieFrameImageRepresentationsType`
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.

Available in Mac OS X v10.5 and later.

Declared in `QTMovie.h`.

`QTMovieFrameImageOpenGLContext`
For `CVOpenGLTextureRef`, the OpenGL context to use. The value is an `NSValue` (`CGLContextObj`).

Available in Mac OS X v10.5 and later.

Declared in `QTMovie.h`.

`QTMovieFrameImagePixelFormat`
For `CVOpenGLTextureRef`, the pixel format to use. Value is an `NSValue` (`CGLPixelFormatObj`).

Available in Mac OS X v10.5 and later.

Declared in `QTMovie.h`.

`QTMovieFrameImageInterlaced`
Image is interlaced. Value is an `NSNumber` (`BOOL`) (default = `NO`).

`QTMovieFrameImageHighQuality`
Image is high quality. Value is an `NSNumber` (`BOOL`) (default = `YES`).

Available in Mac OS X v10.5 and later.

Declared in `QTMovie.h`.

`QTMovieFrameImageSingleField`
Image is single field. Value is an `NSNumber` (`BOOL`) (default = `YES`). The returned object is an autorelease object.

Available in Mac OS X v10.5 and later.

Declared in `QTMovie.h`.

`QTMovieFrameImageSessionMode`
: Indicates that two or more calls to `frameImageAtTime:withAttributes:error:` will be made on the same `QTMovie` object.

  By adding this key with the associated value that is an `NSNumber` wrapping the `BOOLYES` to the dictionary of attributes, an application indicates that it will make more than one call to `frameImageAtTime:withAttributes:error:` on the same `QTMovie` object. This knowledge permits `QTMovie` to cache certain objects and data structures used to generate a frame image, thereby improving performance. When the caller has obtained all the frame images desired from a given `QTMovie` object, the caller should follow those session calls with a call where this value is `NO`; this is a signal to `QTMovie` to dispose of that cached data.

  Declared in `QTMovie.h`.

  Mac OS X v10.6; QuickTime 7.6.3 and later.

Data Locator Attributes

The following constants are data locators that you can use to specify movie attributes, using the `movieWithAttributes` and `initWithAttributes` methods.

```
NSString * const QTMovieDataReferenceAttribute;
NSString * const QTMoviePasteboardAttribute;
NSString * const QTMovieDataAttribute;
```

Constants

`QTMovieDataReferenceAttribute`
: The data reference of a `QTMovie` object;the value for this key is of type `QTDataReference`.

  This attribute can be read but not written. This attribute can be read but not written when the movie has been initialized with `QTMovieOpenForPlaybackAttribute` set to `YES`.

  Available in Mac OS X v10.4 and later.

  Declared in `QTMovie.h`.

`QTMoviePasteboardAttribute`
: The pasteboard setting of a `QTMovie` object.

  Available in Mac OS X v10.4 and later.

  Declared in `QTMovie.h`.

`QTMovieDataAttribute`
: The data of a `QTMovie` object.

  Available in Mac OS X v10.4 and later.

  Declared in `QTMovie.h`.

Movie Instantiation Options

The following constants are movie instantiation options that you can use to specify movie attributes, using the `movieWithAttributes` and `initWithAttributes` methods.

```
NSString * const QTMovieFileOffsetAttribute;
NSString * const QTMovieResolveDataRefsAttribute;
NSString * const QTMovieAskUnresolvedDataRefAttribute;
NSString * const QTMovieOpenAsyncOKAttribute;
NSString * const QTMovieOpenAsyncRequiredAttribute;
NSString * const QTMovieOpenForPlaybackAttribute;
```

Constants

`QTMovieFileOffsetAttribute`

The file offset value; the value for this key is of type `NSNumber`, interpreted as a `long long`.

Available in Mac OS X v10.4 and later.

Declared in `QTMovie.h`.

`QTMovieResolveDataRefsAttribute`

Indicates whether external data references in a movie file should be resolved (`NSNumber YES`) or not resolved (`NSNumber NO`).

A movie file can contain references to media data in other locations. By default, `QTMovie` attempts to resolve these references at the time that the movie file is opened and a `QTMovie` object is instantiated. You can prevent that resolution from occurring by passing an `NSNumber` wrapping the value `NO` as the value of this attribute.

Available in Mac OS X v10.4 and later.

Declared in `QTMovie.h`.

`QTMovieAskUnresolvedDataRefAttribute`

Indicates whether the user should be prompted to help find any unresolved data references (`NSNumber YES`) or not (`NSNumber NO`).

When the value of the `QTMovieResolveDataRefsAttribute` attribute is an `NSNumber` wrapping the value `YES` and a movie file contains unresolved data references, this attribute indicates whether the user should be prompted to help find the missing referenced data (`NSNumber YES`) or not (`NSNumber` NO). Typically, `QTMovie` will display a dialog box that allows the user to navigate to the file or URL containing the referenced data. By setting this attribute to `NO`, you can prevent that dialog box from being displayed and thereby speed up the movie opening and initialization process.

`QTMovieOpenAsyncOKAttribute`

Indicates whether a movie file can be opened asynchronously if possible (`NSNumber NO`) or not (`NSNumber YES`).

Opening a movie file and initializing a `QTMovie` object for that file may require a considerable amount of time, perhaps to convert the data in the file from one format to another. By setting this attribute to an `NSNumber` wrapping the value `YES`, you grant `QTMovie` permission to return a non-nil `QTMovie` identifier to your application immediately and then to continue processing the file data internally. If a movie is opened asynchronously, you must monitor the movie load state and ensure that it has reached the appropriate threshold before attempting to perform certain operations on the movie. For instance, you cannot export or copy a QTMovie `object` until its load state has reached `QTMovieLoadStateComplete`.

Available in Mac OS X v10.4 and later.

Declared in `QTMovie.h`.

`QTMovieOpenAsyncRequiredAttribute`

Indicates whether the `QTMovie` must be opened asynchronously (`NSNumber YES`) or not (`NSNumber NO`).

Set this attribute to an `NSNumber` wrapping `YES` to indicate that all operations necessary to open the movie file (or other container) and create a valid `QTMovie` object must occur asynchronously. That is to say, the methods `+movieWithAttributes:error:` and `-initWithAttributes:error:` must return almost immediately, performing any lengthy operations on another thread. Your application can monitor the movie load state to determine the progress of those operations.

If you require asynchronous opening but QTMovie is unable to honor your request, then the methods `+movieWithAttributes:error:` and `-initWithAttributes:error:` return `nil` with an NSError having the error domain QTKitErrorDomain and code QTErrorMovieOpeningCannotBeAsynchronous.

Declared in `QTMovie.h`.

Mac OS X v10.6 and later; QuickTime 7.6.3 and later.

`QTMovieOpenForPlaybackAttribute`

Indicates whether the `QTMovie` will be used only for playback (`NSNumber YES`) or not (`NSNumber NO`).

Set this attribute to an `NSNumber` wrapping `YES` to indicate that you intend to use movie playback methods (such as `-play` or `-stop`, or corresponding movie view methods such as `-play:` or `-pause:`) to control the movie, but do not intend to use other methods that edit, export, or in any way modify the movie. Knowing that you need playback services only may allow `QTMovie` to use more efficient code paths for some media files.

This attribute is meaningful only when added to the dictionary passed to `-initWithAttributes:error:`. In particular, setting this attribute on a `QTMovie` object that is already open has no effect.

Declared in `QTMovie.h`.

Mac OS X v10.6 and later; QuickTime 7.6.3 and later.

Movie Chapter Information

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.

```
NSString * const QTMovieChapterName;
NSString * const QTMovieChapterStartTime;
NSString * const QTMovieChapterTargetTrackAttribute;
```

Constants

`QTMovieChapterName`

A key indicating the chapter name in the dictionaries that are array elements in the array returned by `QTMoviechapters` or passed to `QTMovieaddChapters:withAttributes:error`.

Available in Mac OS X v10.5 and later.

Declared in `QTMovie.h`.

`QTMovieChapterStartTime`

A key indicating the chapter start time in the dictionaries that are array elements in the array returned by `QTMoviechapters` or passed to `QTMovieaddChapters:withAttributes:error`.

Available in Mac OS X v10.5 and later.

Declared in `QTMovie.h`.

`QTMovieChapterTargetTrackAttribute`
A key indicating the track in the `QTMovie object` that is the target of the chapter track.

Available in Mac OS X v10.5 and later.

Declared in `QTMovie.h`.

Pasteboard Support

The following constant is the type of movie data passed on the pasteboard.

```
NSString * const QTMoviePasteboardType;
```

Constants
`QTMoviePasteboardType`
Specifies the type of movie data passed on the pasteboard.

Available in Mac OS X v10.4 and later.

Declared in `QTMovie.h`.

Exceptions

The following exception is raised when calling a method requiring editing or modification of a movie that is uneditable.

```
NSString * const QTMovieUneditableException;
```

Constants
`QTMovieUneditableException`
Raised when the developer tries to call a method that requires editing or modifying the movie on an uneditable movie.

Available in Mac OS X v10.4 and later.

Declared in `QTMovie.h`.

Notifications

QTMovieApertureModeDidChangeNotification

Issued when the aperture mode of the target `QTMovie` object changes.

Availability
Available in Mac OS X v10.5 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 v10.4 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 v10.4 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 v10.4 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 v10.4 and later.

Declared In
`QTMovie.h`

QTMovieEditabilityDidChangeNotification

Sent when the editable state of a movie has changed.

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

Declared In
`QTMovie.h`

QTMovieLoadStateDidChangeNotification

Sent when the load state of a movie has changed.

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

Declared In
`QTMovie.h`

QTMovieNaturalSizeDidChangeNotification

Posted whenever the natural size (that is, the original dimensions of the movie when it was authored) changes, either because the movie was edited or because new information about the movie was loaded asynchronously.

All clients that display movies using dimensions based on the `QTMovieNaturalSizeAttribute` should respond to this notification to update their display as necessary.

Availability
QuickTime 7.6.3 and later.

Declared In
`QTMovie.h`

QTMovieSizeDidChangeNotification

Sent when the size of a movie has changed.

This notification contains no userInfo dictionary.

Availability
QuickTime 7.0 and later, but deprecated in QuickTime 7.6.3 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 v10.4 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 v10.4 and later.

Declared In
`QTMovie.h`

QTMovieVolumeDidChangeNotification

Sent when the volume of a movie has changed.

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

Declared In
`QTMovie.h`

QTMovieLayer Class Reference

Inherits from	CALayer : NSObject
Conforms to	NSCoding (CALayer) CAMediaTiming (CALayer) NSObject (NSObject)
Framework	/System/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. `QTMovieLayer` renders a `QTMovie` within a layer hierarchy. Note that this class requires rendering using visual contexts. Do not attempt to directly modify the `contents` property of an `QTMovieLayer` object. Doing so will effectively turn it into a regular `CALayer`.

Tasks

Creating Movie Layers

`+ layerWithMovie:` (page 228)
Creates an autoreleased `QTMovieLayer` associated with the specified `QTMovie` object.

`- initWithMovie:` (page 228)
Creates a `QTMovieLayer` associated with the specified `QTMovie` object.

`- movie` (page 229)
Returns the movie associated with a `QTMovieLayer` object.

`- setMovie:` (page 229)
Sets the `QTMovie` object in a `QTMovieLayer` to *`movie`*.

Class Methods

layerWithMovie:

Creates an autoreleased `QTMovieLayer` associated with the specified `QTMovie` object.

```
+ (id)layerWithMovie:(QTMovie *)movie
```

Parameters

movie

The QuickTime movie with which to create an autoreleased QuickTime layer object.

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
```

Parameters

movie

The QuickTime movie with which to initialize the QuickTime layer object.

Discussion
This is the designated initializer. 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`

setMovie:

Sets the `QTMovie` object in a `QTMovieLayer` to *`movie`*.

```
- (void)setMovie:(QTMovie *)movie
```

Discussion
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 v10.5 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	QTAudioContextInsert QTAudioExtractionPanel QTKitCreateMovie 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` object 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 239)
  Initializes a newly allocated `QTMovieView` with *`frame`* as its frame rectangle.

Getting View Characteristics

- `movie` (page 242)
  Returns the `QTMovie` object associated with the `QTMovieView`.
- `isControllerVisible` (page 240)
  Returns an indication of whether the `QTMovieView` has been requested to display a built-in movie controller user interface.
- `isEditable` (page 241)
  Returns `YES` if the `QTMovieView` object is editable.
- `preservesAspectRatio` (page 244)
  Returns `YES` if the `QTMovieView` object maintains the aspect ratio of the movie when drawing it in the view.
- `fillColor` (page 237)
  Returns the fill color of the `QTMovieView`.
- `movieBounds` (page 243)
  Returns the rectangle currently occupied by the movie in a `QTMovieView`.
- `movieControllerBounds` (page 243)
  Returns the rectangle currently occupied by the movie controller bar (if it's visible) in a `QTMovieView`.
- `controllerBarHeight` (page 236)
  Returns the height of the controller bar.

Setting View Characteristics

- `setMovie:` (page 248)
  Sets the QTMovie object in a QTMovieView to *`movie`*.
- `setControllerVisible:` (page 246)
  Sets the visibility state of the movie controller bar in a `QTMovieView` to *`controllerVisible`*.
- `setPreservesAspectRatio:` (page 248)
  Sets the aspect ratio state of a `QTMovieView` to *`preservesAspectRatio`*.
- `setShowsResizeIndicator:` (page 249)
  Shows or hides the movie controller grow box.
- `setFillColor:` (page 247)
  Sets the fill color of a QTMovieView to *`fillColor`*.
- `setEditable:` (page 247)
  Sets the edit state of a `QTMovieView` to *`editable`*.

- `selectNone:` (page 245)
  Selects nothing.

Controlling Movie Playback

- `play:` (page 244)
  Starts the movie playing at its current location.
- `pause:` (page 244)
  Pauses the movie playing.
- `gotoBeginning:` (page 238)
  Sets the current movie time to the beginning of the movie.
- `gotoEnd:` (page 238)
  Sets the current movie time to the end of the movie.
- `gotoNextSelectionPoint:` (page 238)
  Sets the current movie time to the next selection point.
- `gotoPreviousSelectionPoint:` (page 239)
  Sets the current movie time to the previous selection point.
- `gotoPosterFrame:` (page 239)
  Sets the current movie time to the movie poster frame.
- `stepForward:` (page 251)
  Steps the movie forward one frame.
- `stepBackward:` (page 250)
  Steps the movie backward one frame.

Editing a Movie

- `cut:` (page 236)
  Deletes the current movie selection from the movie, placing it on the clipboard.
- `copy:` (page 236)
  Copies the current movie selection onto the clipboard.
- `paste:` (page 243)
  Inserts the contents of the clipboard (if it contains a movie clip) into the movie at the current play position.
- `selectAll:` (page 245)
  Selects the entire movie.
- `delete:` (page 237)
  Deletes the current movie selection from the movie, placing it on the clipboard.
- `add:` (page 235)
  Adds the contents of the clipboard to the movie at the current movie time.
- `addScaled:` (page 235)
  Adds the contents of the clipboard to the movie, scaled to fit into the current movie selection.
- `replace:` (page 245)
  Replaces the current movie selection with the contents of the clipboard.

- `trim:` (page 251)
  Trims the movie to the current movie selection.

Showing and Hiding Buttons in the Movie Controller Bar

- `setBackButtonVisible:` (page 246)
  Sets the specified controller bar button to be visible or invisible, according to the state parameter.
- `setCustomButtonVisible:` (page 246)
  Sets the specified controller bar button to be visible or invisible, according to the state parameter.
- `setHotSpotButtonVisible:` (page 248)
  Sets the specified controller bar button to be visible or invisible, according to the state parameter.
- `setStepButtonsVisible:` (page 249)
  Sets the specified controller bar button to be visible or invisible, according to the state parameter.
- `setTranslateButtonVisible:` (page 250)
  Sets the specified controller bar button to be visible or invisible, according to the state parameter.
- `setVolumeButtonVisible:` (page 250)
  Sets the specified controller bar button to be visible or invisible, according to the state parameter.
- `setZoomButtonsVisible:` (page 250)
  Sets the specified controller bar button to be visible or invisible, according to the state parameter.
- `isBackButtonVisible` (page 239)
  Returns the current visibility state of the specified controller bar button.
- `isCustomButtonVisible` (page 240)
  Returns the current visibility state of the specified controller bar button.
- `isHotSpotButtonVisible` (page 241)
  Returns the current visibility state of the specified controller bar button.
- `areStepButtonsVisible` (page 235)
  Returns the current visibility state of the specified controller bar button.
- `isTranslateButtonVisible` (page 241)
  Returns the current visibility state of the specified controller bar button.
- `isVolumeButtonVisible` (page 242)
  Returns the current visibility state of the specified controller bar button.
- `areZoomButtonsVisible` (page 235)
  Returns the current visibility state of the specified controller bar button.

Delegate Methods

- `menuForEventDelegate:` (page 242)
  Returns an `NSMenu` object that is the contextual menu for the specified event.
- `delegate` (page 237)
  Returns the receiver's delegate.
- `setDelegate:` (page 247)
  Sets the receiver's delegate.

Instance Methods

add:

Adds the contents of the clipboard to the movie at the current movie time.

```
- (IBAction)add:(id)sender
```

Discussion
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:

Adds the contents of the clipboard to the movie, scaled to fit into the current movie selection.

```
- (IBAction)addScaled:(id)sender
```

Discussion
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
QuickTime 7.2.1 or later.

Related Sample Code
QTKitButtonTester

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
QuickTime 7.2.1 or later.

Related Sample Code
QTKitButtonTester

Declared In
`QTMovieView.h`

controllerBarHeight

Returns the height of the controller bar.

```
- (float)controllerBarHeight
```

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

Related Sample Code
QTAudioExtractionPanel
QTKitAdvancedDocument

Declared In
`QTMovieView.h`

copy:

Copies the current movie selection onto the clipboard.

```
- (IBAction)copy:(id)sender
```

Discussion
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:

Deletes the current movie selection from the movie, placing it on the clipboard.

```
- (IBAction)cut:(id)sender
```

Discussion
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`

delegate

Returns the receiver's delegate.

```
- (id)delegate
```

Availability
QuickTime 7.2.1 or later.

Declared In
`QTMovieView.h`

delete:

Deletes the current movie selection from the movie, placing it on the clipboard.

```
- (IBAction)delete:(id)sender
```

Discussion
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

Returns the fill color of the `QTMovieView`.

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

Parameters
`fillColor`
The fill color of the `QTMovieView` object.

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

Related Sample Code
QTAudioContextInsert

QTAudioExtractionPanel
QTKitImport
QTKitPlayer

Declared In
`QTMovieView.h`

gotoBeginning:

Sets the current movie time to the beginning of the movie.

```
- (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:

Sets the current movie time to the end of the movie.

```
- (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:

Sets the current movie time to the next selection point.

```
- (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:

Sets the current movie time to the movie poster frame.

```
- (IBAction)gotoPosterFrame:(id)sender
```

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

Declared In
QTMovieView.h

gotoPreviousSelectionPoint:

Sets the current movie time to the previous selection point.

```
- (IBAction)gotoPreviousSelectionPoint:(id)sender
```

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

Declared In
QTMovieView.h

initWithFrame:

Initializes a newly allocated `QTMovieView` with *frame* as its frame rectangle.

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

Parameters

frame

The `NSRect` object with which to initialize the `QTMovieView` with its frame rectangle.

Discussion
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 through Mac OS X v10.5.

Declared In
QTMovieView.h

isBackButtonVisible

Returns the current visibility state of the specified controller bar button.

```
- (BOOL)isBackButtonVisible
```

Availability
QuickTime 7.2.1 or later.

Related Sample Code
QTKitButtonTester

Declared In
`QTMovieView.h`

isControllerVisible

Returns an indication of whether the `QTMovieView` has been requested to display a built-in movie controller user interface.

```
- (BOOL)isControllerVisible
```

Discussion
Using the `setControllerVisible:` (page 246) method, the client tells `QTMovieView` whether or not to display a user interface for controlling the movie within its bounds. Using the `isControllerVisible` method, the client can determine whether a `QTMovieView` has been configured to display such an interface. By using the `controllerBarHeight` (page 236) method, you can determine the height of the portion of the `QTMovieView` that is required to display that interface. Note that some types of QuickTime content are authored to display their own user interface; for those types of content it is possible for the `controllerBarHeight` (page 236) method to return 0 even when the `isControllerVisible` method is `YES`.

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

Related Sample Code
QTAudioContextInsert
QTAudioExtractionPanel
QTKitImport
QTKitPlayer

Declared In
`QTMovieView.h`

isCustomButtonVisible

Returns the current visibility state of the specified controller bar button.

```
- (BOOL)isCustomButtonVisible
```

Availability
QuickTime 7.2.1 or later.

Related Sample Code
QTKitButtonTester

Declared In
QTMovieView.h

isEditable

Returns YES if the QTMovieView object is editable.

```
- (BOOL)isEditable
```

Parameters
isEditable
The editable state being returned by the QTMovieView object.

Discussion
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
QuickTime 7.2.1 or later.

Related Sample Code
QTKitButtonTester

Declared In
QTMovieView.h

isTranslateButtonVisible

Returns the current visibility state of the specified controller bar button.

```
- (BOOL)isTranslateButtonVisible
```

Availability
QuickTime 7.2.1 or later.

Related Sample Code
QTKitButtonTester

Declared In
QTMovieView.h

isVolumeButtonVisible

Returns the current visibility state of the specified controller bar button.

```
- (BOOL)isVolumeButtonVisible
```

Availability
QuickTime 7.2.1 or later.

Related Sample Code
QTKitButtonTester

Declared In
`QTMovieView.h`

menuForEventDelegate:

Returns an `NSMenu` object that is the contextual menu for the specified event.

```
- (NSMenu *)menuForEventDelegate:(NSEvent *)event
```

Parameters

event
An `NSEvent` object that specifies an event.

Discussion
This delegate method can be used instead of subclassing `QTMovieView` in cases where an application cannot hard-link against the QTKit framework.

Availability
Mac OS X v10.6; QuickTime 7.6.3 or later.

movie

Returns the `QTMovie` object associated with the `QTMovieView`.

```
- (QTMovie *)movie
```

Parameters

movie
The QuickTime movie to be returned with the `QTMovieView` object.

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

Related Sample Code
iChatTheater
MyMediaPlayer
QTKitThreadedExport
QTKitTimeCode

Declared In
`QTMovieView.h`

movieBounds

Returns the rectangle currently occupied by the movie in a `QTMovieView`.

```
- (NSRect)movieBounds
```

Parameters

movieBounds

The `NSRect` rectangle returned by the movie in a `QTMovieView` object.

Discussion

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

Returns the rectangle currently occupied by the movie controller bar (if it's visible) in a `QTMovieView`.

```
- (NSRect)movieControllerBounds
```

Availability

Available in Mac OS X v10.3 and later.

Related Sample Code

QTAudioExtractionPanel

QTKitImport

QTKitPlayer

Declared In

`QTMovieView.h`

paste:

Inserts the contents of the clipboard (if it contains a movie clip) into the movie at the current play position.

```
- (IBAction)paste:(id)sender
```

Discussion

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:

Pauses the movie playing.

```
- (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
ExtractMovieAudioToAIFF
MyMovieFilter
QTExtractAndConvertToAIFF

Declared In
`QTMovieView.h`

play:

Starts the movie playing at its current location.

```
- (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
iChatTheater
MyMediaPlayer
MyMovieFilter

Declared In
`QTMovieView.h`

preservesAspectRatio

Returns `YES` if the `QTMovieView` object maintains the aspect ratio of the movie when drawing it in the view.

```
- (BOOL)preservesAspectRatio
```

Parameters
preservesAspectRatio
The state of the aspect ratio returned by the `QTMovieView` object.

Discussion
The remainder is filled with `fillColor`.

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

Related Sample Code
QTAudioContextInsert
QTAudioExtractionPanel
QTKitImport
QTKitPlayer

Declared In
`QTMovieView.h`

replace:

Replaces the current movie selection with the contents of the clipboard.

```
- (IBAction)replace:(id)sender
```

Discussion
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:

Selects the entire movie.

```
- (IBAction)selectAll:(id)sender
```

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

Declared In
`QTMovieView.h`

selectNone:

Selects nothing.

```
- (IBAction)selectNone:(id)sender
```

Discussion
Note that this method 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
QuickTime 7.2.1 or later.

Related Sample Code
QTKitButtonTester

Declared In
`QTMovieView.h`

setControllerVisible:

Sets the visibility state of the movie controller bar in a `QTMovieView` to *`controllerVisible`*.

```
- (void)setControllerVisible:(BOOL)controllerVisible
```

Parameters
`controllerVisible`
The state of controller visibility set in a `QTMovieView` object.

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

Related Sample Code
QTAudioContextInsert
QTAudioExtractionPanel
QTKitImport
QTKitPlayer

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
QuickTime 7.2.1 or later.

Related Sample Code
QTKitButtonTester

Declared In
QTMovieView.h

setDelegate:

Sets the receiver's delegate.

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

Availability
QuickTime 7.2.1 or later.

Declared In
QTMovieView.h

setEditable:

Sets the edit state of a QTMovieView to *editable*.

```
- (void)setEditable:(BOOL)editable
```

Parameters
editable
The editable state of the QTMovieView object.

Discussion
The default state is NO.

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

Related Sample Code
GLUT

Declared In
QTMovieView.h

setFillColor:

Sets the fill color of a QTMovieView to *fillColor*.

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

Discussion
Note that this may cause a redraw.

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

Related Sample Code
QTAudioContextInsert
QTAudioExtractionPanel

QTKitImport
QTKitPlayer

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
QuickTime 7.2.1 or later.

Related Sample Code
QTKitButtonTester

Declared In
QTMovieView.h

setMovie:

Sets the QTMovie object in a QTMovieView to *movie*.

```
- (void)setMovie:(QTMovie *)movie
```

Discussion
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.

Related Sample Code
ExtractMovieAudioToAIFF
MyMediaPlayer
QTAudioExtractionPanel
QTExtractAndConvertToAIFF
QTKitPlayer

Declared In
QTMovieView.h

setPreservesAspectRatio:

Sets the aspect ratio state of a `QTMovieView` to *preservesAspectRatio*.

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

Parameters

preservesAspectRatio

The aspect ratio of the movie rectangle.

Discussion

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.

Related Sample Code

QTAudioContextInsert

QTAudioExtractionPanel

QTKitImport

QTKitPlayer

Declared In

QTMovieView.h

setShowsResizeIndicator:

Shows or hides the movie controller grow box.

```
- (void)setShowsResizeIndicator:(BOOL)show
```

Availability

Available in Mac OS X v10.3 and later.

Related Sample Code

QTKitPlayer

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

QuickTime 7.2.1 or later.

Related Sample Code

QTKitButtonTester

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
QuickTime 7.2.1 or later.

Related Sample Code
QTKitButtonTester

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
QuickTime 7.2.1 or later.

Related Sample Code
QTKitButtonTester

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
QuickTime 7.2.1 or later.

Related Sample Code
QTKitButtonTester

Declared In
`QTMovieView.h`

stepBackward:

Steps the movie backward one frame.

```
- (IBAction)stepBackward:(id)sender
```

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

Declared In
QTMovieView.h

stepForward:

Steps the movie forward one frame.

```
- (IBAction)stepForward:(id)sender
```

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

Declared In
QTMovieView.h

trim:

Trims the movie to the current movie selection.

```
- (IBAction)trim:(id)sender
```

Discussion
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

Constants

Movie View Bindings

Constants for different movie view bindings.

```
NSString * const QTMovieViewMovieBinding;
NSString * const QTMovieViewControllerVisibleBinding;
NSString * const QTMovieViewPreservesAspectRatioBinding;
NSString * const QTMovieViewFillColorBinding;
```

Constants
QTMovieViewMovieBinding
A QTMovieView binding that determines the receiver's movie. Value is a QTMovie.
Available in Mac OS X v10.4 and later.
Declared in QTMovieView.h.

`QTMovieViewControllerVisibleBinding`
A `QTMovieView` binding that determines whether or not the controls are visible. Value is a boolean.

Available in Mac OS X v10.4 and later.

Declared in `QTMovieView.h`.

`QTMovieViewPreservesAspectRatioBinding`
A `QTMovieView` binding that determines whether or not the receiver preserves the natural aspect ratio of the movie. Value is a boolean.

Available in Mac OS X v10.4 and later.

Declared in `QTMovieView.h`.

`QTMovieViewFillColorBinding`
A `QTMovieView` binding that determines the fill color. Value is an `NSColor`.

Available in Mac OS X v10.4 and later.

Declared in `QTMovieView.h`.

QTSampleBuffer Class Reference

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

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 254)
  Returns a sample buffer attribute for the given key.
- `audioBufferListWithOptions:` (page 255)
  Returns a pointer to a Core Audio `AudioBufferList` containing audio data owned by the receiver.
- `bytesForAllSamples` (page 255)
  Returns a pointer to the bytes of media data contained in the sample buffer.
- `decodeTime` (page 256)
  Returns the decode time of the buffer.

- `decrementSampleUseCount` (page 256)
  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 257)
  Returns the duration of the buffer.
- `formatDescription` (page 257)
  Returns the format description of the buffer.
- `getAudioStreamPacketDescriptions:inRange:` (page 257)
  Gets an array of Core Audio AudioStreamPacketDescriptions describing the lengths of samples in variable bit- rate audio buffers.
- `incrementSampleUseCount` (page 258)
  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 258)
  Returns the length of the buffer returned by `bytesForAllSamples`.
- `numberOfSamples` (page 259)
  Returns the number of media samples contained in the buffer.
- `presentationTime` (page 259)
  Returns the presentation time of the buffer.
- `sampleBufferAttributes` (page 259)
  Returns a dictionary of the sample buffer's current attirbutes.
- `sampleUseCount` (page 260)
  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 *)key
```

Parameters

key

The key of the returned attribute. Attribute keys are described in the "Sample Buffer Attributes" (page 260) 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.

Related Sample Code
AudioDataOutputToAudioUnit

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.

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 analogous 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.

Related Sample Code
AudioDataOutputToAudioUnit

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.

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.

Related Sample Code
AudioDataOutputToAudioUnit

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

Sample Buffer Attributes

The following are constants for different sample buffer attributes.

```
NSString * const QTSampleBufferHostTimeAttribute;
NSString * const QTSampleBufferSMPTETimeAttribute
NSString * const QTSampleBufferSceneChangeTypeAttribute;
NSString * const QTSampleBufferDateRecordedAttribute;
NSString * const QTSampleBufferExplicitSceneChange;
NSString * const QTSampleBufferTimeStampDiscontinuitySceneChange;
```

Constants

`QTSampleBufferHostTimeAttribute`

Returns the buffer's host time, if the buffer is from a real time source.

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.

Available in Mac OS X v10.5 and later.

Declared in `QTSampleBuffer.h`.

`QTSampleBufferSMPTETimeAttribute`

Returns the SMPTE timecode of the sample buffer, if it has one.

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.

Available in Mac OS X v10.5 and later.

Declared in `QTSampleBuffer.h`.

`QTSampleBufferSceneChangeTypeAttribute`

If the buffer marks a scene change in the input content, returns a constant.

The returned constant specifies the type of scene change. This string value can be used in key paths for key-value coding, key-value observing, and bindings.

Available in Mac OS X v10.5 and later.

Declared in `QTSampleBuffer.h`.

`QTSampleBufferDateRecordedAttribute`

Returns the date on which the media in the buffer was originally recorded.

The value is an `NSDate`. This string value can be used in key paths for key-value coding, key-value observing, and bindings.

Available in Mac OS X v10.5 and later.

Declared in `QTSampleBuffer.h`.

`QTSampleBufferExplicitSceneChange`

Indicates that a scene change was explicitly marked in the sample buffer's metadata.

This constant is returned by `QTSampleBufferSceneChangeTypeAttribute` specifying what kind of scene change, if any, is marked by a sample buffer.

Available in Mac OS X v10.5 and later.

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.

This constant is returned by `QTSampleBufferSceneChangeTypeAttribute` specifying what kind of scene chnage, if any, is marked by a sample buffer.

Available in Mac OS X v10.5 and later.

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	MoviePlayer - C# QTAudioContextInsert QTAudioExtractionPanel QTKitTimeCode QTMetadataEditor

Overview

A `QTTrack` object is an object that represents the ordering and other characteristics of media data in a `QTMovie` object, such as a single video track or audio track. A `QTMovie` object typically contains one or more streams of media data, which are represented by `QTTrack` objects. When a `QTMovie` object has been initialized with `QTMovieOpenForPlaybackAttribute` set to `NO`, a `QTTrack` object wraps the underlying QuickTime track (of type `Track`). A `QTMovie` object may have several `QTTrack` objects associated with it. By constrast, a `QTTrack` object has exactly one `QTMedia` object associated with it.

Tasks

Creating a QTTrack

`+ trackWithQuickTimeTrack:error:` (page 265)
Returns a `QTTrack` object associated with a QuickTime Track.

Initializing a QTTrack

`- initWithQuickTimeTrack:error:` (page 268)
Returns a `QTTrack` object associated with a QuickTime Track.

Getting Track Properties

- `movie` (page 271)
  Returns the `QTMovie` object associated with a `QTTrack` object.
- `media` (page 270)
  Returns the `QTMedia` object associated with a `QTTrack` object.
- `isEnabled` (page 270)
  Returns `YES` if the `QTTrack` object is currently enabled, `NO` otherwise.
- `volume` (page 275)
  Returns the volume of a `QTTrack` object.
- `attributeForKey:` (page 267)
  Returns the current value of an attribute of a `QTTrack` object.
- `trackAttributes` (page 274)
  Returns a dictionary containing the current values of public attributes of a `QTTrack` object.

Setting Track Properties

- `setEnabled:` (page 273)
  Sets the enabled state of a `QTTrack` to *`enabled`*.
- `setVolume:` (page 274)
  Sets the volume of a `QTTrack` object.
- `setAttribute:forKey:` (page 273)
  Set the track attribute *`attributeKey`* to the value specified by the *`value`* parameter.
- `setTrackAttributes:` (page 274)
  Sets the attributes of a `QTTrack` object using the key-value pairs in a specified dictionary.

Editing Track Properties

- `addImage:forDuration:withAttributes:` (page 266)
  Adds an image to a `QTTrack` object for the specified duration, using attributes specified in the attributes dictionary.
- `deleteSegment:` (page 267)
  Deletes a specified segment from a `QTTrack` object.
- `insertEmptySegmentAt:` (page 269)
  Inserts into a QTTrack an empty segment delimited by the range *`range`*.
- `insertSegmentOfTrack:timeRange:atTime:` (page 270)
  Inserts into a `QTTrack` object the specified segment of another `QTTrack` object.
- `insertSegmentOfTrack:fromRange:scaledToRange:` (page 269)
  Inserts into a `QTTrack` object the specified segment of another `QTTrack` object, scaling that new segment to a specified start time and duration.
- `scaleSegment:newDuration:` (page 272)
  Scales the `QTTrack` segment delimited by the segment *`segment`* so that it will have the new duration *`newDuration`*.

Getting QTTrack Primitives

- `quickTimeTrack` (page 271)
  Returns the QuickTime track associated with a `QTTrack` object.

Getting and Setting Aperture Mode Dimensions

- `apertureModeDimensionsForMode:` (page 266)
  Returns an NSSize value that indicates the dimensions of the target track for the specified movie aperture mode.
- `setApertureModeDimensions:forMode:` (page 272)
  Sets the dimensions of the target track for the specified movie aperture mode.
- `generateApertureModeDimensions` (page 268)
  Adds aperture mode dimensions information to a `QTTrack` object.
- `removeApertureModeDimensions` (page 272)
  Removes aperture mode dimensions information from the `QTTrack` object.

Class Methods

trackWithQuickTimeTrack:error:

Returns a `QTTrack` object associated with a QuickTime Track.

```
+ (id)trackWithQuickTimeTrack:(Track)track
    error:(NSError **)errorPtr
```

Parameters

track
  A QuickTime Track with which to initialize the `QTTrack` object.

errorPtr
  A pointer to an `NSError` object; if a `QTTrack` object cannot be created, an `NSError` object is returned in this location.

Discussion
This method cannot be called when the movie containing this track has been initialized with `QTMovieOpenForPlaybackAttribute` set to `YES`. In addition, this method cannot be called by 64-bit applications. Pass `NULL` 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:

Adds an image to a `QTTrack` object for the specified duration, using attributes specified in the attributes dictionary.

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

Parameters

image

An `NSImage` that is to be appended to the target track.

duration

A `QTTime` structure that indicates the desired duration of the appended image in the track.

attributes

An `NSDictionary` object that specifies attributes of the appended image.

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.)

Discussion
This method cannot be called when the movie containing this track has been initialized with `QTMovieOpenForPlaybackAttribute` set to `YES`.

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.

```
- (NSSize)apertureModeDimensionsForMode:(NSString *)mode
```

Parameters

mode

An `NSString` object that indicates the aperture mode whose dimensions are to be returned; pass values like `QTMovieApertureModeClean`.

Discussion
For instance, passing a mode of `QTMovieApertureModeClean` would cause `apertureModeDimensionsForMode:` to return the track dimensions to use in clean aperture mode. This method cannot be called when the movie containing this track has been initialized with `QTMovieOpenForPlaybackAttribute` set to `YES`.

Availability
QuickTime 7.2 or later.

Declared In
`QTTrack.h`

attributeForKey:

Returns the current value of an attribute of a `QTTrack` object.

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

Parameters

`attributeKey`

An `NSString` object that specifies the attribute to be read; pass strings like `QTTrackTimeScaleAttribute` or `QTTrackVolumeAttribute`.

Return Value
An `NSObject` that is the value of the specified attribute key.

Discussion
This method can be called when the movie containing this track has been initialized with `QTMovieOpenForPlaybackAttribute` set to `YES`. A list of supported track attributes and their acceptable values can be found in the "Track Attributes" (page 275) section.

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

Related Sample Code
QTKitPlayer
QTMetadataEditor
TrackFormatDemo

Declared In
`QTTrack.h`

deleteSegment:

Deletes a specified segment from a `QTTrack` object.

```
- (void)deleteSegment:(QTTimeRange)segment
```

Parameters

`segment`

A `QTTimeRange` structure that indicates the segment in the target track that is to be deleted.

Discussion
This method cannot be called when the movie containing this track has been initialized with `QTMovieOpenForPlaybackAttribute` set to `YES`. If the movie containing this 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 aperture mode dimensions information to a QTTrack object.

```
- (void)generateApertureModeDimensions
```

Discussion
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. This method cannot be called when the movie containing this track has been initialized with QTMovieOpenForPlaybackAttribute set to YES.

Availability
QuickTime 7.2 or later.

Declared In
QTTrack.h

initWithQuickTimeTrack:error:

Returns a QTTrack object associated with a QuickTime Track.

```
- (id)initWithQuickTimeTrack:(Track)track
    error:(NSError **)errorPtr
```

Parameters

track
A QuickTime Track with which to initialize the QTTrack object.

errorPtr
A pointer to an NSError object; if a QTTrack object cannot be created, an NSError object is returned in this location.

Discussion
This method cannot be called when the movie containing this track has been initialized with QTMovieOpenForPlaybackAttribute set to YES. In addition, this method cannot be called by 64-bit applications. Pass NULL 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:

Inserts into a QTTrack an empty segment delimited by the range *range*.

```
- (void)insertEmptySegmentAt:(QTTimeRange)range
```

Parameters

range

A QTTimeRange structure that indicates the segment in the target track at which an empty segment is to be inserted.

Discussion

This method cannot be called when the movie containing this track has been initialized with QTMovieOpenForPlaybackAttribute set to YES. If the movie containing this 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:

Inserts into a QTTrack object the specified segment of another QTTrack object, scaling that new segment to a specified start time and duration.

```
- (void)insertSegmentOfTrack:(QTTrack *)track
    fromRange:(QTTimeRange)srcRange
    scaledToRange:(QTTimeRange)dstRange
```

Parameters

track

The QTTrack object from which the segment to be inserted is copied.

srcRange

A QTTimeRange structure that indicates the segment in track to be copied.

dstRange

A QTTimeRange structure that indicates the range in the target track into which the copied segment is to be inserted.

Discussion

This method cannot be called when the movie containing this track has been initialized with QTMovieOpenForPlaybackAttribute set to YES. This is essentially an Add Scaled operation on a track. If the movie containing this 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:

Inserts into a `QTTrack` object the specified segment of another `QTTrack` object.

```
- (void)insertSegmentOfTrack:(QTTrack *)track
    timeRange:(QTTimeRange)range
    atTime:(QTTime)time
```

Parameters

track

The `QTTrack` object from which the segment to be inserted is copied.

range

A `QTTimeRange` structure that indicates the segment in track to be copied.

time

A `QTTime` structure that indicates the time in the target track at which the copied segment is to be inserted.

Discussion
This method cannot be called when the movie containing this track has been initialized with `QTMovieOpenForPlaybackAttribute` set to `YES`. If the movie containing this track is not editable, this method raises an exception.

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

Declared In
`QTTrack.h`

isEnabled

Returns `YES` if the `QTTrack` object is currently enabled, `NO` otherwise.

```
- (BOOL)isEnabled
```

Discussion
This method can be called when the movie containing this track has been initialized with `QTMovieOpenForPlaybackAttribute` set to `YES`.

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

Declared In
`QTTrack.h`

media

Returns the `QTMedia` object associated with a `QTTrack` object.

```
- (QTMedia *)media
```

Discussion
This method can be called when the movie containing this track has been initialized with `QTMovieOpenForPlaybackAttribute` set to `YES`.

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

Related Sample Code
QTKitTimeCode
QTMetadataEditor

Declared In
`QTTrack.h`

movie

Returns the `QTMovie` object associated with a `QTTrack` object.

```
- (QTMovie *)movie
```

Discussion
This method can be called when the movie containing this track has been initialized with `QTMovieOpenForPlaybackAttribute` set to `YES`.

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

Declared In
`QTTrack.h`

quickTimeTrack

Returns the QuickTime track associated with a `QTTrack` object.

```
- (Track)quickTimeTrack
```

Discussion
This method cannot be called when the movie containing this track has been initialized with `QTMovieOpenForPlaybackAttribute` set to `YES`. In addition, this method cannot be called by 64-bit applications.

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

Related Sample Code
QTAudioContextInsert
QTAudioExtractionPanel
QTKitTimeCode

Declared In
`QTTrack.h`

removeApertureModeDimensions

Removes aperture mode dimensions information from the `QTTrack` object.

```
- (void)removeApertureModeDimensions
```

Discussion
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`. This method cannot be called when the movie containing this track has been initialized with `QTMovieOpenForPlaybackAttribute` set to `YES`.

Availability
QuickTime 7.2 or later.

Declared In
`QTTrack.h`

scaleSegment:newDuration:

Scales the `QTTrack` segment delimited by the segment *`segment`* so that it will have the new duration *`newDuration`*.

```
- (void)scaleSegment:(QTTimeRange)segment
    newDuration:(QTTime)newDuration
```

Parameters

`segment`
A `QTTimeRange` structure that indicates the segment in the target track that is to be scaled.

`newDuration`
A `QTTime` structure that indicates the desired duration of the segment that is to be scaled.

Discussion
This method cannot be called when the movie containing this track has been initialized with `QTMovieOpenForPlaybackAttribute` set to `YES`. 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
```

Parameters

`dimensions`
An `NSSize` structure that indicates the desired dimensions for the specified aperture mode.

mode

An `NSString` object that indicates the aperture mode whose dimensions are to be set; pass values like `QTMovieApertureModeClean`.

Discussion
This method cannot be called when the movie containing this track has been initialized with `QTMovieOpenForPlaybackAttribute` set to `YES`.

Availability
QuickTime 7.2 or later.

Declared In
`QTTrack.h`

setAttribute:forKey:

Set the track attribute *attributeKey* to the value specified by the *value* parameter.

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

Parameters

attributes

An `NSDictionary` object that specifies the attributes to set and their desired values.

Discussion
This method can be called when the movie containing this track has been initialized with `QTMovieOpenForPlaybackAttribute` set to `YES`. However, certain attributes may not be writable when the movie containing this track has been initialized with `QTMovieOpenForPlaybackAttribute` set to `YES`. A list of supported track attributes and their acceptable values can be found in the "Track Attributes" (page 275) section.

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

Declared In
`QTTrack.h`

setEnabled:

Sets the enabled state of a `QTTrack` to *enabled*.

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

Parameters

enabled

The desired track enabled state.

Discussion
This method can be called when the movie containing this track has been initialized with `QTMovieOpenForPlaybackAttribute` set to `YES`.

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

Related Sample Code
QTKitTimeCode

Declared In
`QTTrack.h`

setTrackAttributes:

Sets the attributes of a `QTTrack` object using the key-value pairs in a specified dictionary.

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

Parameters

`attributes`
An `NSDictionary` object that specifies the attributes to set and their desired values.

Discussion
This method can be called when the movie containing this track has been initialized with `QTMovieOpenForPlaybackAttribute` set to `YES`. However, certain attributes may not be writable when the movie containing this track has been initialized with `QTMovieOpenForPlaybackAttribute` set to `YES`. A list of supported track attributes and their acceptable values can be found in the "Track Attributes" (page 275) section.

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

Declared In
`QTTrack.h`

setVolume:

Sets the volume of a `QTTrack` object.

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

Parameters

`volume`
The desired track volume.

Discussion
This method can be called when the movie containing this track has been initialized with `QTMovieOpenForPlaybackAttribute` set to `YES`. The valid range is 0.0 to 1.0.

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

Declared In
`QTTrack.h`

trackAttributes

Returns a dictionary containing the current values of public attributes of a `QTTrack` object.

```
- (NSDictionary *)trackAttributes
```

Discussion
This method can be called when the movie containing this track has been initialized with `QTMovieOpenForPlaybackAttribute` set to `YES`. A list of supported track attributes and their acceptable values can be found in the "Track Attributes" (page 275) section.

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

Declared In
`QTTrack.h`

volume

Returns the volume of a `QTTrack` object.

```
- (float)volume
```

Discussion
This method can be called when the movie containing this track has been initialized with `QTMovieOpenForPlaybackAttribute` set to `YES`. The valid range is 0.0 to 1.0.

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

Declared In
`QTTrack.h`

Constants

Track Attributes

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`.

```
NSString * const QTTrackBoundsAttribute;
NSString * const QTTrackCreationTimeAttribute;
NSString * const QTTrackDimensionsAttribute;
NSString * const QTTrackDisplayNameAttribute;
NSString * const QTTrackEnabledAttribute;
NSString * const QTTrackFormatSummaryAttribute;
NSString * const QTTrackIsChapterTrackAttribute;
NSString * const QTTrackHasApertureModeDimensionsAttribute;
NSString * const QTTrackIDAttribute;
NSString * const QTTrackLayerAttribute;
NSString * const QTTrackMediaTypeAttribute;
NSString * const QTTrackModificationTimeAttribute;
NSString * const QTTrackRangeAttribute;
NSString * const QTTrackTimeScaleAttribute;
NSString * const QTTrackUsageInMovieAttribute;
NSString * const QTTrackUsageInPosterAttribute;
NSString * const QTTrackUsageInPreviewAttribute;
NSString * const QTTrackVolumeAttribute;
```

Constants

`QTTrackBoundsAttribute`

The bounding rectangle of a `QTTrack` object; the value for this key is of type `NSValue`, interpreted as an `NSRect`.

This attribute can be read and written. This attribute can be read and written when the movie containing this track has been initialized with `QTMovieOpenForPlaybackAttribute` set to `YES`.

Available in Mac OS X v10.4 and later.

Declared in `QTTrack.h`.

`QTTrackCreationTimeAttribute`

The creation time of a `QTTrack` object; the value for this key is of type `NSDate`.

This attribute can be read and written. This attribute can be read and written when the movie containing this track has been initialized with `QTMovieOpenForPlaybackAttribute` set to `YES`.

Available in Mac OS X v10.4 and later.

Declared in `QTTrack.h`.

`QTTrackDimensionsAttribute`

The dimensions of a `QTTrack` object; the value for this key is of type `NSValue`, interpreted as an `NSSize`.

This attribute can be read and written. This attribute can be read and written when the movie containing this track has been initialized with `QTMovieOpenForPlaybackAttribute` set to `YES`.

Available in Mac OS X v10.4 and later.

Declared in `QTTrack.h`.

`QTTrackDisplayNameAttribute`

The display name of a `QTTrack` object; the value for this key is of type `NSString`.

This attribute can be read and written. This attribute can be read and written when the movie containing this track has been initialized with `QTMovieOpenForPlaybackAttribute` set to `YES`.

Available in Mac OS X v10.4 and later.

Declared in `QTTrack.h`.

`QTTrackEnabledAttribute`
: The track enabled state of a `QTTrack` object; the value for this key is of type `NSNumber`, interpreted as a `BOOL`.

  This attribute can be read and written. This attribute can be read and written when the movie containing this track has been initialized with `QTMovieOpenForPlaybackAttribute` set to `YES`.

  Available in Mac OS X v10.4 and later.

  Declared in `QTTrack.h`.

`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 can be read and written. This attribute can be read and written when the movie containing this track has been initialized with `QTMovieOpenForPlaybackAttribute` set to `YES`.

  Declared in `QTTrack.h`.

  Mac OS X v10.5 and later.

`QTTrackIsChapterTrackAttribute`
: Whether a `QTTrack` object is a chapter track for some other `QTTrack` object; the value for this key is of type NSNumber, interpreted as a `BOOL`.

  This attribute can be read and written. This attribute can be read and written when the movie containing this track has been initialized with `QTMovieOpenForPlaybackAttribute` set to `YES`.

  Available in Mac OS X v10.5 and later.

  Declared in `QTTrack.h`.

`QTTrackHasApertureModeDimensionsAttribute`
: Whether aperture mode dimensions have been set on a `QTTrack` object; the value for this key is of type `NSNumber`, interpreted as a `BOOL`.

  This attribute can be read and written. This attribute can be read and written when the movie containing this track has been initialized with `QTMovieOpenForPlaybackAttribute` set to `YES`.

  Available in Mac OS X v10.5 and later.

  Declared in `QTTrack.h`.

`QTTrackIDAttribute`
: The track ID of a `QTTrack` object; the value for this key is of type `NSNumber`, interpreted as a `long`.

  This attribute can be read and written. This attribute can be read and written when the movie containing this track has been initialized with `QTMovieOpenForPlaybackAttribute` set to `YES`.

  Available in Mac OS X v10.4 and later.

  Declared in `QTTrack.h`.

`QTTrackLayerAttribute`
: The track layer of a `QTTrack` object; the value for this key is of type `NSNumber`, interpreted as a `short`.

  This attribute can be read and written. This attribute can be read and written when the movie containing this track has been initialized with `QTMovieOpenForPlaybackAttribute` set to `YES`.

  Available in Mac OS X v10.4 and later.

  Declared in `QTTrack.h`.

`QTTrackMediaTypeAttribute`
The media type of a `QTTrack` object; the value for this key is of type `NSString`.

This attribute can be read and written. This attribute can be read and written when the movie containing this track has been initialized with `QTMovieOpenForPlaybackAttribute` set to `YES`.

Available in Mac OS X v10.4 and later.

Declared in `QTTrack.h`.

`QTTrackModificationTimeAttribute`
The modification time of a `QTTrack` object; the value for this key is of type `NSDate`.

This attribute can be read and written. This attribute can be read and written when the movie containing this track has been initialized with `QTMovieOpenForPlaybackAttribute` set to `YES`.

Available in Mac OS X v10.4 and later.

Declared in `QTTrack.h`.

`QTTrackRangeAttribute`
The range of time this track occupies; the value for this key is of type `NSValue`, interpreted as a `QTTimeRange`.

This attribute can be read and written. This attribute can be read and written when the movie containing this track has been initialized with `QTMovieOpenForPlaybackAttribute` set to `YES`.

Available in Mac OS X v10.4 and later.

Declared in `QTTrack.h`.

`QTTrackTimeScaleAttribute`
The time scale of a `QTTrack` object; the value for this key is of type `NSNumber`, interpreted as a `long`.

This attribute can be read and written. This attribute can be read and written when the movie containing this track has been initialized with `QTMovieOpenForPlaybackAttribute` set to `YES`.

Available in Mac OS X v10.4 and later.

Declared in `QTTrack.h`.

`QTTrackUsageInMovieAttribute`
Whether a `QTTrack` object contributes data to the movie; the value for this key is of type `NSNumber`, interpreted as a `BOOL`.

This attribute can be read and written. This attribute can be read and written when the movie containing this track has been initialized with `QTMovieOpenForPlaybackAttribute` set to `YES`.

Available in Mac OS X v10.4 and later.

Declared in `QTTrack.h`.

`QTTrackUsageInPosterAttribute`
Whether a `QTTrack` object contributes data to the movie poster; the value for this key is of type `NSNumber`, interpreted as a `BOOL`.

This attribute can be read and written. This attribute can be read and written when the movie containing this track has been initialized with `QTMovieOpenForPlaybackAttribute` set to `YES`.

Available in Mac OS X v10.4 and later.

Declared in `QTTrack.h`.

`QTTrackUsageInPreviewAttribute`

Whether a `QTTrack` object contributes data to the movie preview; the value for this key is of type `NSNumber`, interpreted as a `BOOL`.

This attribute can be read and written. This attribute can be read and written when the movie containing this track has been initialized with `QTMovieOpenForPlaybackAttribute` set to `YES`.

Available in Mac OS X v10.4 and later.

Declared in `QTTrack.h`.

`QTTrackVolumeAttribute`

The volume of a `QTTrack` object; the value for this key is of type `NSNumber`, interpreted as a `float`.

This attribute can be read and written. This attribute can be read and written when the movie containing this track has been initialized with `QTMovieOpenForPlaybackAttribute` set to `YES`.

Available in Mac OS X v10.4 and later.

Declared in `QTTrack.h`.

Functions

QTKit Functions Reference

Framework: /System/Library/Frameworks/QTKit.framework
Declared in QTKit/QTTime.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 286)
Creates a `QTTime` structure.

`QTMakeTimeScaled` (page 287)
Returns a `QTTime` structure.

`QTTimeFromString` (page 291)
Returns a `QTTime` structure.

`QTMakeTimeWithTimeRecord` (page 288)
Creates a `QTTime` structure.

`QTMakeTimeWithTimeInterval` (page 288)
Creates a `QTTime` structure.

Getting and Setting Times

The following functions are used to get and set times.

`QTGetTimeRecord` (page 285)
Returns the value of a `QTTime` structure expressed as a `TimeRecord`.

`QTGetTimeInterval` (page 285)
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 290)
Returns a value of type `NSComparisonResult`.

`QTSMPTETimeCompare` (page 289)
Compares two `SMPTETime` structures.

`QTStringFromSMPTETime` (page 289)
Returns a human-readable string from the `SMPTETime`. The returned string is of the form hh:mm:ss.ff.

Adding and Subtracting Times

The following functions are used to add and subtract times:

`QTTimeIncrement` (page 291)
Adds two `QTTime` structures.

`QTTimeDecrement` (page 291)
Subtracks one `QTTime` from another.

Getting a Time Description

The following function is used to get a time description:

`QTStringFromTime` (page 289)
Returns a description of a `QTTime` structure.

Time Range Functions

`QTEqualTimeRanges` (page 285)
Returns `YES` if the specified time ranges are identical.

`QTIntersectionTimeRange` (page 286)
Returns a `QTTimeRange` structure that represents the intersection of the two ranges.

`QTMakeTimeRange` (page 287)
Returns a `QTTimeRange` structure initialized using the QTTime structures `time` and `duration`.

`QTStringFromTimeRange` (page 290)
Returns a description of a `QTTimeRange` structure.

`QTTimeInTimeRange` (page 292)
Returns `YES` if the specified time time lies in the time range range.

`QTTimeRangeEnd` (page 292)
Returns a `QTTime` structure representing the end of the specified time range.

`QTTimeRangeFromString` (page 292)
Returns a `QTTimeRange` structure

`QTUnionTimeRange` (page 293)
Returns a `QTTimeRange` structure.

QuickTime Helper Functions

`QTStringForOSType` (page 289)
Returns an NSString representing the specified four-character code type.

`QTOSTypeForString` (page 288)
Returns a four-character code representing the specified `NSString`.

Functions

QTEqualTimeRanges

Returns `YES` if the specified time ranges are identical.

```
BOOL QTEqualTimeRanges (
   QTTimeRange range,
   QTTimeRange range2
);
```

Discussion
This function returns `YES` if the specified time ranges are identical.

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

Declared In
`QTTimeRange.h`

QTGetTimeInterval

Returns the value of a `QTTime` structure expressed as an `NSTimeInterval`.

```
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 v10.4 and later.

Declared In
`QTTime.h`

QTGetTimeRecord

Returns the value of a `QTTime` structure expressed as a `TimeRecord`.

```
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 v10.4 and later.

Related Sample Code
QTAudioContextInsert
QTAudioExtractionPanel

Declared In
`QTTime.h`

QTIntersectionTimeRange

Returns a `QTTimeRange` structure that represents the intersection of the two ranges.

```
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 v10.4 and later.

Declared In
`QTTimeRange.h`

QTMakeTime

Creates a `QTTime` structure.

```
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 v10.4 and later.

Related Sample Code
QTAudioContextInsert
QTAudioExtractionPanel
QTKitCommandLine
QTKitCreateMovie
QTKitMovieShuffler

Declared In
`QTTime.h`

QTMakeTimeRange

Returns a `QTTimeRange` structure initialized using the QTTime structures `time` and `duration`.

```
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 v10.4 and later.

Related Sample Code
QTKitCommandLine
QTKitMovieShuffler

Declared In
`QTTimeRange.h`

QTMakeTimeScaled

Returns a `QTTime` structure.

```
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 v10.4 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 v10.4 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 v10.4 and later.

Related Sample Code
QTAudioContextInsert
QTAudioExtractionPanel

Declared In
`QTTime.h`

QTOSTypeForString

Returns a four-character code representing the specified `NSString`.

```
OSType QTOSTypeForString (
   NSString *string
);
```

Discussion
This function returns a four-character code representing the specified `NSString`.

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

Declared In
`QTUtilities.h`

QTSMPTETimeCompare

Compares two `SMPTETime` structures.

```
NSComparisonResult QTSMPTETimeCompare(SMPTETime time, SMPTETIme otherTime)
```

QTStringForOSType

Returns an NSString representing the specified four-character code type.

```
NSString * QTStringForOSType (
   OSType type
);
```

Discussion
This function returns an `NSString` representing the specified four-character code type.

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

Declared In
`QTUtilities.h`

QTStringFromSMPTETime

Returns a human-readable string from the `SMPTETime`. The returned string is of the form hh:mm:ss.ff.

```
NSString* QTStringFromSMPTETime(SMPTETime time)
```

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

Declared In
`QTTime.h`

QTStringFromTime

Returns a description of a `QTTime` structure.

```
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 v10.4 and later.

Related Sample Code
CIVideoDemoGL

QTAudioExtractionPanel
QTKitPlayer
QTRecorder

Declared In
QTTime.h

QTStringFromTimeRange

Returns a description of a QTTimeRange structure.

```
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 v10.4 and later.

Declared In
QTTimeRange.h

QTTimeCompare

Returns a value of type NSComparisonResult.

```
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 v10.4 and later.

Related Sample Code
QTAudioContextInsert
QTAudioExtractionPanel
QTKitMovieShuffler

Declared In
QTTime.h

QTTimeDecrement

Subtracts one `QTTime` from another.

```
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 v10.4 and later.

Related Sample Code
QTAudioContextInsert
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 v10.4 and later.

Related Sample Code
QTAudioContextInsert
QTAudioExtractionPanel

Declared In
`QTTime.h`

QTTimeIncrement

Adds two `QTTime` structures.

```
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 v10.4 and later.

Declared In
`QTTime.h`

QTTimeInTimeRange

Returns `YES` if the specified time time lies in the time range range.

```
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 v10.4 and later.

Declared In
`QTTimeRange.h`

QTTimeRangeEnd

Returns a `QTTime` structure representing the end of the specified time range.

```
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 v10.4 and later.

Declared In
`QTTimeRange.h`

QTTimeRangeFromString

Returns a `QTTimeRange` structure

```
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 v10.4 and later.

Declared In
`QTTimeRange.h`

QTUnionTimeRange

Returns a `QTTimeRange` structure.

```
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 v10.4 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`

Constants

PART IV

Constants

QTKit Constants Reference

Framework: /System/Library/Frameworks/QTKit.framework

Declared in QTKit/QTError.h

Overview

This document defines constants in the QTKit framework that are not associated with a particular class.

Constants

QTKit Error Domain

The QTKit error domain identifier, and keys for extracting specific values from the userInfo dictionary of an error returned by QTKit.

```
NSString * const QTKitErrorDomain;
NSString * const QTErrorCaptureInputKey;
NSString * const QTErrorCaptureOutputKey;
NSString * const QTErrorDeviceKey;
NSString * const QTErrorExcludingDeviceKey;
NSString * const QTErrorTimeKey;
NSString * const QTErrorFileSizeKey;
NSString * const QTErrorRecordingSuccesfullyFinishedKey;
```

Constants

`QTKitErrorDomain`

The QTKit error domain identifier.

Available in Mac OS X v10.5 and later.

Declared in `QTError.h`.

`QTErrorCaptureInputKey`

Use this key to retrieve the `QTCaptureInput` object for which the error occurred.

Available in Mac OS X v10.5 and later.

Declared in `QTError.h`.

`QTErrorCaptureOutputKey`

Use this key to retrieve the `QTCaptureOutput` object for which the error occurred.

Available in Mac OS X v10.5 and later.

Declared in `QTError.h`.

`QTErrorDeviceKey`

Use this key to retrieve the `QTCaptureDevice` object for which the error occurred.

Available in Mac OS X v10.5 and later.

Declared in `QTError.h`.

`QTErrorExcludingDeviceKey`

Use this key to retrieve the `QTCaptureDevice` object for the device whose presence is excluding the device for which the error occurred.

Available in Mac OS X v10.5 and later.

Declared in `QTError.h`.

`QTErrorTimeKey`

An `NSValue` interpreted as `QTTime`.

Mac OS X v10.6; QuickTime 7.6.1.

Declared in `QTError.h`.

`QTErrorFileSizeKey`

Use this key to interpret the file size in bytes as an `NSNumber`.

Mac OS X v10.6; QuickTime 7.6.1.

Declared in `QTError.h`.

`QTErrorRecordingSuccesfullyFinishedKey`

Use this key to determine whether the products of a recording were successfully finished after recording stopped due to an error. The value is an `NSNumber` interpreted as a `BOOL`.

Available in Mac OS X v10.5 and later.

Declared in `QTError.h`.

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,
    QTErrorMaximumNumberOfSamplesForFileFormatReached   = 1208,
    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`.

`QTErrorInvalidInputOrOutputs`
: 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`.

`QTErrorMaximumNumberOfSamplesForFileFormatReached`
: Indicates the maximum number of samples reached for the file format.

  Mac OS X v10.6; QuickTime 7.6.3.

  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`.

Document Revision History

This table describes the changes to *QTKit Framework Reference*.

Date	Notes
2009-02-26	Added QTCaptureDecompressedAudioOutput class to collection; minor edits and fixes to the Introduction.
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

D

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W