QuickTime 7.1 Update Reference

QuickTime

2006-08-14

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

Simultaneously published in the United States and Canada.

Contents

Introduction

Framework	QuickTime
Header file directories	Movies.h, ImageCompression.h, QuickTimeComponents.h, MoviesFormat.h, ImageCodec.h
Declared in	ImageCodec.h ImageCompression.h MediaHandlers.h Movies.h MoviesFormat.h QuickTimeComponents.h

Introduction

Functions

PART I

Functions

Functions

Functions by Task

Visual Context APIs for Windows

`GetMovieVisualContext` (page 11)
Returns the current visual context for a movie.

`ICMDecompressionSessionCreateForVisualContext` (page 12)
Creates a session for decompressing video frames.

`QTDirect3DTextureContextCreate` (page 15)
Creates a Direct3D texture context for a specified Direct3D context and pixel format.

`QTVisualContextCopyImageForTime` (page 16)
Retrieves an image buffer from the visual context, indexed by the provided time.

`QTVisualContextGetAttribute` (page 17)
Returns a visual context attribute.

`QTVisualContextGetTypeID` (page 18)
Returns the `CFTypeID` for `QTVisualContextRef`.

`QTVisualContextIsNewImageAvailable` (page 18)
Queries whether a new image is available for a given time.

`QTVisualContextSetAttribute` (page 20)
Sets a visual context attribute.

`QTVisualContextSetImageAvailableCallback` (page 21)
Installs a user-defined callback to receive notifications when a new image becomes available.

`QTVisualContextRelease` (page 19)
Releases a visual context object.

`QTVisualContextRetain` (page 19)
Retains a visual context object.

`QTVisualContextTask` (page 21)
Enables the visual context to release internally held resources for later use.

`SetMovieVisualContext` (page 24)
Targets a movie to render into a visual context.

Aperture Mode APIs

`SetTrackApertureModeDimensionsUsingSampleDescription` (page 25)
Sets a track's aperture mode dimensions using values calculated using a sample description.

Audio API For Windows

Functions

GenerateMovieApertureModeDimensions

Examines a movie and sets up track aperture mode dimensions.

```
OSErr GenerateMovieApertureModeDimensions (
   Movie movie
);
```

Parameters

movie
The movie.

Discussion
This function can be used to add information needed to support aperture modes to tracks created with applications and/or versions of QuickTime that did not support aperture mode dimensions. If the image descriptions in video tracks lack tags describing clean aperture and pixel aspect ratio information, the media data may be scanned to see if the correct values can be determined and attached. Then the aperture mode dimensions are calculated and set. Afterwards, the kQTVisualPropertyID_HasApertureModeDimensions property will be set to TRUE for these tracks. Tracks that do not support aperture modes are not changed.

Version Notes
Introduced in QuickTime 7.1.

Availability
Carbon status: Supported C interface file: ImageCompression.h

Declared In
`Movies.h`

GenerateTrackApertureModeDimensions

Examines a track and sets up aperture mode dimensions.

```
OSErr GenerateTrackApertureModeDimensions (
   Track track
);
```

Parameters

track
The track.

Discussion
This function can be used to add information needed to support aperture modes to tracks created with applications and/or versions of QuickTime that did not support aperture mode dimensions. If the image descriptions in video tracks lack tags describing clean aperture and pixel aspect ratio information, the media data may be scanned to see if the correct values can be determined and attached. Then the aperture mode dimensions are calculated and set. Afterwards, the `kQTVisualPropertyID_HasApertureModeDimensions` property will be set to `TRUE` for these tracks. Tracks that do not support aperture modes are not changed.

Version Notes
Introduced in QuickTime 7.1.

Availability
Carbon status: Supported C interface file: ImageCompression.h

Declared In
`Movies.h`

GetMovieVisualContext

Returns the current visual context for a movie.

```
OSStatus GetMovieVisualContext (
   Movie movie,
   QTVisualContextRef *visualContext
);
```

Parameters

movie
The movie.

visualContext
A pointer to a variable to receive the visual context.

Return Value
An error code. Returns `noErr` if there is no error. Returns `memFullErr` if memory cannot be allocated. Returns `kQTVisualContextRequiredErr` if the movie is not using a visual context. Returns `paramErr` if the movie or *`visualContextOut`* is `NULL`.

Discussion
Returns the QTVisualContext object associated with the movie. You are responsible for retaining and releasing the object as needed (that is, if the returned object has not been retained for you). If the visual context was set to `NULL` (see `SetMovieVisualContext`), `noErr` is returned and *`visualContextOut`* receives `NULL`.

Version Notes
Introduced in QuickTime 7 for Mac OS X and in QuickTime 7.1 for Windows.

Availability
Carbon status: Supported C interface file: Movies.h.

Declared In
`Movies.h`

ICMDecompressionSessionCreateForVisualContext

Creates a session for decompressing video frames.

```
OSStatus ICMDecompressionSessionCreateForVisualContext (
   CFAllocatorRef allocator,
   ImageDescriptionHandle desc,
   ICMDecompressionSessionOptionsRef decompressionOptions,
   QTVisualContextRef visualContext,
   ICMDecompressionTrackingCallbackRecord *trackingCallback,
   ICMDecompressionSessionRef *decompressionSessionOut
);
```

Parameters

`allocator`
An allocator for the session. Pass `NULL` to use the default allocator.

`desc`
An image description for the source frames.

`decompressionOptions`
Options for the session. The session will retain this options object. You may change some options during the session by modifying the object.

`visualContext`
The target visual context.

`trackingCallback`
The callback to be called with information about queued frames and pixel buffers containing the decompressed frames.

`decompressionSessionOut`
Points to a variable to receive the new decompression session.

Return Value
An error code. Returns `noErr` if there is no error.

Discussion
Frames will be output to a visual context. If desired, the *trackingCallback* may attach additional data to pixel buffers before they are sent to the visual context.

Version Notes
Introduced in QuickTime 7 for Mac OS X and in QuickTime 7.1 for Windows.

Availability
Carbon status: Supported C interface file: ImageCompression.h

Related Sample Code
QTQuartzPlayer

Declared In
`ImageCompression.h`

MediaGenerateApertureModeDimensions

Examines media and sets up track aperture mode dimensions.

```
ComponentResult MediaGenerateApertureModeDimensions (
   MediaHandler mh
);
```

Parameters

mh
The media handler.

Discussion
If the sample descriptions tracks lack tags describing clean aperture and pixel aspect ratio information, the media data may be scanned to see if the correct values can be determined and attached. Then the aperture mode dimensions should be calculated and set by `MediaSetTrackApertureModeDimensionsUsingSampleDescription`.

Version Notes
Introduced in QuickTime 7.1

Availability
Carbon status: Supported C interface file: ImageCompression.h

Declared In
`MediaHandlers.h`

MediaGetApertureModeClipRectForSampleDescriptionIndex

Calculates a source clip rectangle appropriate for the current aperture mode and the given sample description.

```
ComponentResult MediaGetApertureModeClipRectForSampleDescriptionIndex (
   MediaHandler mh,
   long sampleDescIndex,
   FixedRect *clipFixedRectOut
);
```

Parameters

mh

The media handler.

sampleDescIndex

Indicates the sample description index of sample description in the media.

clipFixedRectOut

Points to a variable to receive the clip rectangle.

Discussion

If the track's aperture mode is `kQTApertureMode_CleanAperture`, the rectangle should be the clean aperture as described by the sample description (see `kICMImageDescriptionPropertyID_CleanApertureClipRect` (page 35)); otherwise, it should be the full dimensions of the sample description.

Version Notes

Introduced in QuickTime 7.1.

Availability

Carbon status: Supported C interface file: ImageCompression.h

Declared In

`MediaHandlers.h`

MediaGetApertureModeMatrixForSampleDescriptionIndex

Calculates a matrix appropriate for the current aperture mode and the given sample description.

```
ComponentResult MediaGetApertureModeMatrixForSampleDescriptionIndex (
   MediaHandler mh,
   long sampleDescIndex,
   MatrixRecord *matrixOut
);
```

Parameters

mh

The media handler.

sampleDescIndex

Indicates the sample description index of sample description in the media.

matrixOut

Points to a variable to receive the matrix.

Discussion

If the track's aperture mode is `kQTApertureMode_CleanAperture` or `kQTApertureMode_ProductionAperture`, the matrix should scale horizontally to compensate for the pixel aspect ratio. Otherwise, the matrix should be identity. If the track's aperture mode is

kQTApertureMode_CleanAperture, the matrix should translate the top-left point of the clean aperture to the origin. (See kICMImageDescriptionPropertyID_CleanApertureMatrix (page 35) and kICMImageDescriptionPropertyID_ProductionApertureMatrix (page 36)).

Version Notes
Introduced in QuickTime 7.1.

Availability
Carbon status: Supported C interface file: ImageCompression.h

Declared In
MediaHandlers.h

MediaSetTrackApertureModeDimensionsUsingSampleDescription

Sets the three aperture mode dimension properties on the track, calculating the values using the provided sample description.

```
ComponentResult MediaSetTrackApertureModeDimensionsUsingSampleDescription (
   MediaHandler mh,
   SampleDescriptionHandle sampleDesc
);
```

Parameters

mh
The media handler.

sampleDesc
The sample description handle.

Discussion
Use this function to calculate the values of the three aperture mode dimension properties on the track, using the provided sample description

Version Notes
Introduced in QuickTime 7.1.

Availability
Carbon status: Supported C interface file: ImageCompression.h

Declared In
MediaHandlers.h

QTDirect3DTextureContextCreate

Creates a Direct3D texture context for a specified Direct3D context and pixel format.

```
OSErr QTDirect3DTextureContextCreate (
 CFAllocatorRef              allocator,
 void*                       d3dDevice, // LPDIRECT3DDEVICE9
 UInt32                      d3dPixelFormat,
 CFDictionaryRef             attributes,
 QTVisualContextRef          *newTextureContext );
```

Parameters

allocator

An allocator used to create the texture context.

d3dDevice

A Direct3D device used to create textures.

d3dPixelFormat

The Direct3D pixel format used to create the Direct3D device.

attributes

A dictionary of attributes.

newTextureContext

A pointer to a variable to receive the new Direct3D texture context.

Return Value
An error code. Returns `noErr` if there is no error.

`kQTVisualContextNotAllowedErr` if the graphics hardware is not supported.

`memFullErr` if there is insufficient memory to allocate the visual context.

Discussion
This routine lets you create a visual context you can draw into, if given a Direct3D device. The routine works similar to `QTOpenGLTextureContextCreate` in QuickTime 7 for Mac OS X.

Version Notes
Introduced in QuickTime 7 for Mac OS X and in QuickTime 7.1 for Windows.

Availability
Carbon status: Supported C interface file: ImageCompression.h

Declared In
`ImageCompression.h`

QTVisualContextCopyImageForTime

Retrieves an image buffer from the visual context, indexed by the provided time.

```
OSStatus QTVisualContextCopyImageForTime (
   QTVisualContextRef visualContext,
   CFAllocatorRef allocator,
   const CVTimeStamp *timeStamp,
   CVImageBufferRef *newImage
);
```

Parameters

visualContext

The visual context.

allocator
Allocator used to create new `CVImageBufferRef`.

timeStamp
Time in question. Pass `NULL` to request the image at the current time.

newImage
Points to variable to receive the new image.

Return Value
An error code. Returns `noErr` if there is no error.

Discussion
You should not request image buffers further ahead of the current time than the read-ahead time specified with the `kQTVisualContextExpectedReadAheadKey` attribute. You may skip images by passing later times, but you may not pass an earlier time than passed to a previous call to this function.

Version Notes
Introduced in QuickTime 7 for Mac OS X and in QuickTime 7.1 for Windows.

Availability
Carbon status: Supported C interface file: ImageCompression.h

Related Sample Code
LiveVideoMixer3
QTCoreVideo102
QTCoreVideo201
QTCoreVideo301
VideoViewer

Declared In
`ImageCompression.h`

QTVisualContextGetAttribute

Returns a visual context attribute.

```
OSStatus QTVisualContextGetAttribute (
   QTVisualContextRef visualContext,
   CFStringRef attributeKey,
   CFTypeRef *attributeValueOut
);
```

Parameters

visualContext
The visual context.

attributeKey
Identifier of attribute to get.

attributeValueOut
A pointer to a variable that will receive the attribute value or `NULL` if the attribute is not set.

Return Value
An error code. Returns `noErr` if there is no error.

Discussion
This routine returns a visual context attribute.

Version Notes
Introduced in QuickTime 7 for Mac OS X and in QuickTime 7.1 for Windows.

Availability
Carbon status: Supported C interface file: ImageCompression.h

Declared In
`ImageCompression.h`

QTVisualContextGetTypeID

Returns the `CFTypeID` for `QTVisualContextRef`.

```
CFTypeID QTVisualContextGetTypeID (
   void
);
```

Discussion
Use this function to test whether a `CFTypeRef` that extracted from a CF container such as a CFArray was a `QTVisualContextRef`.

Version Notes
Introduced in QuickTime 7 for Mac OS X and in QuickTime 7.1 for Windows.

Availability
Carbon status: Supported C interface file: ImageCompression.h

Declared In
`ImageCompression.h`

QTVisualContextIsNewImageAvailable

Queries whether a new image is available for a given time.

```
Boolean QTVisualContextIsNewImageAvailable (
   QTVisualContextRef visualContext,
   const CVTimeStamp *timeStamp
);
```

Parameters

`visualContext`
The visual context.

`timeStamp`
Time in question.

Return Value
A Boolean.

Discussion
This function returns `TRUE` if there is a image available for the specified time that is different from the last image retrieved from `QTVisualContextCopyImageForTime`.

Version Notes
Introduced in QuickTime 7 for Mac OS X and in QuickTime 7.1 for Windows.

Availability
Carbon status: Supported C interface file: ImageCompression.h

Related Sample Code
LiveVideoMixer3
QTCoreImage101
QTCoreVideo102
QTCoreVideo201
QTCoreVideo301

Declared In
`ImageCompression.h`

QTVisualContextRelease

Releases a visual context object.

```
void QTVisualContextRelease (
   QTVisualContextRef visualContext
);
```

Parameters

`visualContext`
A reference to a visual context object. If you pass `NULL`, nothing happens.

Discussion
When the retain count decreases to zero, the visual context is disposed.

Version Notes
Introduced in QuickTime 7 for Mac OS X and in QuickTime 7.1 for Windows.

Availability
Carbon status: Supported C interface file: ImageCompression.h

Related Sample Code
LiveVideoMixer3
QTCoreVideo102
QTCoreVideo201
QTCoreVideo301
QTQuartzPlayer

Declared In
`ImageCompression.h`

QTVisualContextRetain

Retains a visual context object.

```
QTVisualContextRef QTVisualContextRetain (
   QTVisualContextRef visualContext
);
```

Parameters

visualContext

A reference to a visual context object. If you pass `NULL`, nothing happens.

Return Value
On return, a reference to the same visual context object, for convenience.

Version Notes
Introduced in QuickTime 7 for Mac OS X and in QuickTime 7.1 for Windows.

Availability
Carbon status: Supported C interface file: ImageCompression.h

Related Sample Code
QTQuartzPlayer

Declared In
`ImageCompression.h`

QTVisualContextSetAttribute

Sets a visual context attribute.

```
OSStatus QTVisualContextSetAttribute (
   QTVisualContextRef visualContext,
   CFStringRef attributeKey,
   CFTypeRef attributeValue
);
```

Parameters

visualContext

The visual context.

attributeKey

Identifier of attribute to set.

attributeValue

The value of the attribute to set, or `NULL` to remove a value.

Return Value
An error code. Returns `noErr` if there is no error.

Version Notes
Introduced in QuickTime 7 for Mac OS X and in QuickTime 7.1 for Windows.

Availability
Carbon status: Supported C interface file: ImageCompression.h

Related Sample Code
CIVideoDemoGL
VideoViewer

Declared In
`ImageCompression.h`

QTVisualContextSetImageAvailableCallback

Installs a user-defined callback to receive notifications when a new image becomes available.

```
OSStatus QTVisualContextSetImageAvailableCallback (
   QTVisualContextRef visualContext,
   QTVisualContextImageAvailableCallback imageAvailableCallback,
   void *refCon
);
```

Parameters

visualContext
The visual context invoking the callback.

imageAvailableCallback
Time for which a new image has become available. May be `NULL`.

refCon
A user-defined value passed to `QTImageAvailableCallback`.

Return Value
An error code. Returns `noErr` if there is no error.

Discussion
Due to unpredictible activity, such as user seeks or the arrival of streaming video packets from a network, new images may become available for times supposedly occupied by previous images. Applications using the CoreVideo display link to drive rendering probably do not need to install a callback of this type, since they will already be checking for new images at a sufficient rate.

Version Notes
Introduced in QuickTime 7 for Mac OS X and in QuickTime 7.1 for Windows.

Availability
Carbon status: Supported C interface file: ImageCompression.h

Related Sample Code
QTPixelBufferVCToCGImage

Declared In
`ImageCompression.h`

QTVisualContextTask

Enables the visual context to release internally held resources for later use.

```
void QTVisualContextTask (
   QTVisualContextRef visualContext
);
```

Parameters

visualContext
The visual context.

Discussion
For optimal resource management, this function should be called in every rendering pass. It should be called after old images have been released, new images have been used and all rendering has been flushed to the screen. The call is not mandatory.

Version Notes
Introduced in QuickTime 7

Availability
Carbon status: Supported C interface file: ImageCompression.h

Related Sample Code
LiveVideoMixer3
QTCoreImage101
QTCoreVideo102
QTCoreVideo201
QTCoreVideo301

Declared In
`ImageCompression.h`

RemoveMovieApertureModeDimensions

Removes aperture mode dimension information from a movie.

```
OSErr RemoveMovieApertureModeDimensions (
   Movie movie
);
```

Parameters

movie
The movie.

Discussion
This function removes aperture mode dimension information from a movie's tracks. It does not attempt to modify sample descriptions, so it may not completely reverse the effect of `GenerateMovieApertureModeDimensions`. It sets the `kQTVisualPropertyID_HasApertureModeDimensions` property to `FALSE`.

Version Notes
Introduced in QuickTime 7.1

Availability
Carbon status: Supported C interface file: ImageCompression.h

Declared In
`Movies.h`

RemoveTrackApertureModeDimensions

Removes aperture mode dimension information from a track.

```
OSErr RemoveTrackApertureModeDimensions (
   Track track
);
```

Parameters

track

The track.

Discussion

This function removes aperture mode dimension information from a track. It does not attempt to modify sample descriptions, so it may not completely reverse the effect of `GenerateTrackApertureModeDimensions`. It sets the `kQTVisualPropertyID_HasApertureModeDimensions` property to `FALSE`.

Version Notes

Introduced in QuickTime 7.1.

Availability

Carbon status: Supported C interface file: ImageCompression.h

Declared In

`Movies.h`

SCAudioFillBuffer

Used is used to pull compressed frames from the StdAudio component in `kQTSCAudioPropertyID_BasicDescription` format.

```
ComponentResult SCAudioFillBuffer (
   ComponentInstance ci,
   SCAudioInputDataProc inInputDataProc,
   void *inInputDataProcRefCon,
   UInt32 *ioOutputDataPacketSize,
   AudioBufferList *outOutputData,
   AudioStreamPacketDescription *outPacketDescription
);
```

Parameters

ci

The client's connection to a StdAudio Compression component.

inInputDataProc

The proc address of the function that will be called to supply data in the `kQTSCAudioPropertyID_InputBasicDescription` format to SCAudio.

inInputDataProcRefCon

The client refcon that will be passed to the user-provided `SCAudioInputDataProc` function.

ioOutputDataPacketSize

On input, the number of desired packets. On output, the actual number of packets delivered (can be fewer than the input desired packets).

outOutputData

An `AudioBufferList` providing sufficiently large buffers to hold the requested number of packets.

outPacketDescription

An array of AudioStreamPacketDescriptions. If the requested output format requires external framing information––that is, a VBR format such as AAC–– allocate and pass an array of packet descriptions as large as the number of packets you are requesting.

Discussion
`SCAudioFillBuffer` preserves the same style API as the Core Audio `AudioConverterFillComplexBuffer` API, which is part of the AudioToolbox framework and resides in `AudioConverter.h`.

This function has the same parameters as `AudioConverterFillComplexBuffer`, but internally it has an audio converter, a matrix mixer, and another audio converter. It is based on a "pull" model: You pull on the audio converter for some output, and specify the buffer in which you want to put it. You have already configured the format that it is going to return to you. You also provide it with an input proc and it will call back that input proc and specify however many samples of audio you have. For example, if you want five packets of AAC audio coming out the back end, and at the front end you happen to have PCM de-interleaved, the routine will ask you for the samples. The difference is you can now have different numbers of channels going in or out. For example, at the same time you are doing the compression from PCM to AAC, you can also go from 5.1 to stereo.

`SCAudioFillBuffer` is used to pull compressed frames from the StdAudio component in `kQTSCAudioPropertyID_BasicDescription` format. The StdAudio component can perform any combination of decompression/mixing/compression, combining the facilities of Core Audio AudioConverters and Matrix Mixer AudioUnits. The behavior of the `SCAudioFillBuffer` call (signalling end of data, and so on) is identical to the AudioConverter's `AudioConverterFillComplexBuffer` API.

One important difference between the AudioConverter and SCAudio component is that the SCAudio compression APIs can do mixing as well as n -> n channels conversion. If you want to compress, decompress, or transcode audio using the `SCAudioFillBuffer` interface, you configure the `StandardCompressionSubTypeAudio` component with the desired input and output formats (or use the `SCRequestImageSettings` API to present a dialog and let a user select an output format), then you call `SCAudioFillBuffer`, providing an `SCAudioInputDataProc` callback, which is called for audio in the specified source format.

Version Notes
Introduced in QuickTime 7.1 for Windows and QuickTime 7 for Mac OS X.

Availability
Carbon status: Supported C interface file: QuickTimeComponents.h

Related Sample Code
SCAudioCompress

Declared In
`QuickTimeComponents.h`

SetMovieVisualContext

Targets a movie to render into a visual context.

```
OSStatus SetMovieVisualContext (
   Movie movie,
   QTVisualContextRef visualContext
);
```

Parameters

movie

The movie.

visualContext

The visual context that the movie will render into. May be NULL.

Return Value

An error code. Returns noErr if there is no error. Returns memFullErr if memory cannot be allocated. Returns kQTVisualContextNotAllowed if the movie is not able to render using a visual context. Returns paramErr if the movie is NULL.

Discussion

When SetMovieVisualContext succeeds, it will retain the QTVisualContext object for its own use. If *visualContext* is NULL, the movie will not render any visual media. SetMovieVisualContext will fail if a different movie is already using the visual context, so you should first disassociate the other movie by calling SetMovieVisualContext with a NULL *visualContext*.

Special Considerations

Note that calling SetMovieGWorld on a movie that is connected to a visual context will work, but it may still keep a reference to the visual context. If you wish to completely disconnect the visual context, make sure to first call SetMovieVisualContext with a NULL *visualContext*.

Version Notes

Introduced in QuickTime 7 for Mac OS X and in QuickTime 7.1 for Windows.

Availability

Carbon status: Supported C interface file: Movies.h

Related Sample Code

QTCoreImage101

QTCoreVideo102

QTCoreVideo201

QTCoreVideo202

QTCoreVideo301

Declared In

Movies.h

SetTrackApertureModeDimensionsUsingSampleDescription

Sets a track's aperture mode dimensions using values calculated using a sample description.

```
ComponentResult ADD_MEDIA_BASENAME()
SetTrackApertureModeDimensionsUsingSampleDescription
```

Parameters

track

The track.

sampleDesc

The sample description handle.

Discussion
This function should be used to add information needed to support aperture modes to newly created tracks. This information is calculated using the given sample description. If sampleDesc is `NULL`, the track's first sample description is used.

Version Notes
Introduced in QuickTime 7.1.

Availability
Carbon status: Supported C interface file: ImageCompression.h

Declared In
`Movies.h`

Types and Constants

Data Types

TrackCleanApertureDimensions

```
struct TrackCleanApertureDimensions {
    long          flags;  /* 1 byte of version / 3 bytes of flags */
    FixedPoint    cleanApertureDimensions;
};
```

Declared In
MoviesFormat.h.

TrackProductionApertureDimensions

```
struct TrackProductionApertureDimensions {
    long          flags;  /* 1 byte of version / 3 bytes of flags */
    FixedPoint    productionApertureDimensions;
};
```

Declared In
MoviesFormat.h.

TrackEncodedPixelsDimensions

```
struct TrackEncodedPixelsDimensions {
    long           flags;  /* 1 byte of version / 3 bytes of flags */
    FixedPoint     encodedPixelsDimensions;
};
```

Declared In
MoviesFormat.h.

Constants

Visual Context Constants

The following are values for `kQTVisualContextTypeKey`, a read-only `CFStringRef` that defines the type of the visual context:

Constants

`kQTVisualContextExpectedReadAheadKey`
A `CFNumberRef` that defines the number of seconds ahead of real time that the client expects to pull images out of a visual context. Applications using the Core Video display link should set this attribute according to the value returned by `CVDisplayLinkGetOutputVideoLatency`.

`kQTVisualContextPixelBufferAttributesKey`
A `CFDictionaryRef` that defines the dictionary containing pixel buffer attributes. See `kICMCompressionSessionPropertyID_PixelBufferPool` in ICM Compression Session Properties.

`kQTVisualContextTargetDimensionsKey`
A `CFDictionaryRef` that defines the dictionary containing `kQTVisualContextTargetDimensions_WidthKey` and `kQTVisualContextTargetDimensions_HeightKey` values. This key is used as a hint to optimize certain media types, such as text, that can be rendered at any resolution. If this attribute is not set, the movie will be rendered at its native resolution.

`kQTVisualContextTargetDimensions_WidthKey`
A `CFNumberRef` that defines the width, in pixels, of the rendering target.

`kQTVisualContextTargetDimensions_HeightKey`
A `CFNumberRef` that defines the height, in pixels, of the rendering target.

`kQTVisualContextType_Direct3DTexture`
The `CFString` value for use in dictionaries for identifying Direct3DTexture visual contexts.

Declared In

`ImageCompression.h`

kQTPropertyClass_ImageDescription

The following are aperture mode properties of image descriptions in QuickTime 7.1.

```
enum {
    kQTPropertyClass_ImageDescription  = 'idsc',
    kICMImageDescriptionPropertyID_EncodedWidth = 'encw',
    kICMImageDescriptionPropertyID_EncodedHeight = 'ench',
    kICMImageDescriptionPropertyID_CleanAperture = 'clap',
    kICMImageDescriptionPropertyID_PixelAspectRatio = 'pasp',
    kICMImageDescriptionPropertyID_CleanApertureDisplayDimensions = 'cadi',
    kICMImageDescriptionPropertyID_ProductionApertureDisplayDimensions = 'prdi',
    kICMImageDescriptionPropertyID_EncodedPixelsDimensions = 'endi',
    kICMImageDescriptionPropertyID_DisplayWidth = 'disw',
    kICMImageDescriptionPropertyID_DisplayHeight = 'dish',
    kICMImageDescriptionPropertyID_ProductionDisplayWidth = 'pdsw',
    kICMImageDescriptionPropertyID_ProductionDisplayHeight = 'pdsh',
    kICMImageDescriptionPropertyID_NCLCColorInfo = 'nclc',
    kICMImageDescriptionPropertyID_FieldInfo = 'fiel',
    kICMImageDescriptionPropertyID_ClassicTrackWidth = 'claw',
    kICMImageDescriptionPropertyID_ClassicTrackHeight = 'clah',
    kICMImageDescriptionPropertyID_GammaLevel = 'gama',
    kICMImageDescriptionPropertyID_RowBytes = 'rowb',
    kICMImageDescriptionPropertyID_StepDuration = 'step',
    kICMImageDescriptionPropertyID_CleanApertureClipRect = 'cacr',
    kICMImageDescriptionPropertyID_CleanApertureMatrix = 'camx',
    kICMImageDescriptionPropertyID_ProductionApertureMatrix = 'pamx',
    kICMImageDescriptionPropertyID_SummaryString = 'isum',
};
```

Constants

`kQTPropertyClass_ImageDescription`

Class identifier for image description properties.

Available in Mac OS X v10.3 and later.

Declared in `ImageCompression.h`.

`kICMImageDescriptionPropertyID_EncodedWidth`

The width of the encoded image. Usually, but not always, this is the ImageDescription's width field.

Available in Mac OS X v10.3 and later.

Declared in `ImageCompression.h`.

`kICMImageDescriptionPropertyID_EncodedHeight`

The height of the encoded image. Usually, but not always, this is the ImageDescription's height field.

Available in Mac OS X v10.3 and later.

Declared in `ImageCompression.h`.

`kICMImageDescriptionPropertyID_CleanAperture`

Describes the clean aperture of the buffer. If not specified explicitly in the image description, the default clean aperture (full encoded width and height) will be returned.

Available in Mac OS X v10.3 and later.

Declared in `ImageCompression.h`.

`kICMImageDescriptionPropertyID_PixelAspectRatio`

Describes the pixel aspect ratio. If not specified explicitly in the image description, a square (1:1) pixel aspect ratio will be returned.

Available in Mac OS X v10.3 and later.

Declared in `ImageCompression.h`.

`kICMImageDescriptionPropertyID_CleanApertureDisplayDimensions`
: The dimensions at which the image can be displayed on a square-pixel display, generally calculated using the clean aperture and pixel aspect ratio. Note that this value is returned as a FixedPoint; the width and height can also be read separately as rounded SInt32s via `kICMImageDescriptionPropertyID_CleanApertureDisplayWidth` and `kICMImageDescriptionPropertyID_CleanApertureDisplayHeight`.

Available in Mac OS X v10.3 and later.

Declared in `ImageCompression.h`.

`kICMImageDescriptionPropertyID_ProductionApertureDisplayDimensions`
: The dimensions at which the image could be displayed on a square-pixel display, disregarding any clean aperture but honoring the pixel aspect ratio. This may be useful for authoring applications that want to expose the edge-processing region. For general viewing, use `kICMImageDescriptionPropertyID_CleanApertureDimensions` instead. Note that this value is returned as a FixedPoint; the width and height can also be read separately as rounded SInt32s via `kICMImageDescriptionPropertyID_ProductionApertureDisplayWidth` and `kICMImageDescriptionPropertyID_ProductionApertureDisplayHeight`.

Available in Mac OS X v10.3 and later.

Declared in `ImageCompression.h`.

`kICMImageDescriptionPropertyID_EncodedPixelsDimensions`
: Describes the dimensions of the encoded image. Note that this value is returned as a FixedPoint for convenience; the width and height can also be read separately as SInt32s via `kICMImageDescriptionPropertyID_EncodedWidth` and `kICMImageDescriptionPropertyID_EncodedHeight`.

Available in Mac OS X v10.3 and later.

Declared in `ImageCompression.h`.

`kICMImageDescriptionPropertyID_CleanApertureDisplayWidth`
: A width at which the buffer's image could be displayed on a square-pixel display, possibly calculated using the clean aperture and pixel aspect ratio.

Available in Mac OS X v10.3 and later.

Declared in `ImageCompression.h`.

`kICMImageDescriptionPropertyID_CleanApertureDisplayHeight`
: A height at which the buffer's image could be displayed on a square-pixel display, possibly calculated using the clean aperture and pixel aspect ratio.

Available in Mac OS X v10.3 and later.

Declared in `ImageCompression.h`.

`kICMImageDescriptionPropertyID_ProductionApertureDisplayWidth`
: A width at which the image could be displayed on a square-pixel display, disregarding any clean aperture but honoring the pixel aspect ratio. This may be useful for authoring applications that want to expose the edge processing region. For general viewing, use `kICMImageDescriptionPropertyID_DisplayWidth` instead.

Available in Mac OS X v10.3 and later.

Declared in `ImageCompression.h`.

`kICMImageDescriptionPropertyID_ProductionApertureDisplayHeight`
: A height at which the image could be displayed on a square-pixel display, disregarding any clean aperture but honoring the pixel aspect ratio. This may be useful for authoring applications that want to expose the edge processing region. For general viewing, use `kICMImageDescriptionPropertyID_DisplayHeight` instead.

  Available in Mac OS X v10.3 and later.

  Declared in `ImageCompression.h`.

`kICMImageDescriptionPropertyID_DisplayWidth`
: Synonym for `kICMImageDescriptionPropertyID_CleanApertureDisplayHeight`.

  Available in Mac OS X v10.3 and later.

  Declared in `ImageCompression.h`.

`kICMImageDescriptionPropertyID_DisplayHeight`
: Synonym for `kICMImageDescriptionPropertyID_CleanApertureDisplayHeight`.

  Available in Mac OS X v10.3 and later.

  Declared in `ImageCompression.h`.

`kICMImageDescriptionPropertyID_ProductionDisplayWidth`
: Synonym for `kICMImageDescriptionPropertyID_ProductionApertureDisplayWidth`.

  Available in Mac OS X v10.3 and later.

  Declared in `ImageCompression.h`.

`kICMImageDescriptionPropertyID_ProductionDisplayHeight`
: Synonym for `kICMImageDescriptionPropertyID_ProductionApertureDisplayHeight`.

  Available in Mac OS X v10.3 and later.

  Declared in `ImageCompression.h`.

`kICMImageDescriptionPropertyID_NCLCColorInfo`
: Color information, if available in the `NCLCColorInfoImageDescriptionExtension` format.

  Available in Mac OS X v10.3 and later.

  Declared in `ImageCompression.h`.

`kICMImageDescriptionPropertyID_CGColorSpace`
: A CGColorSpaceRef for the colorspace described by the image description, constructed from video color information or ICC Profile. It is important to note that the YCbCr matrix from the video color info is *not* represented in the `CGColorSpaceRef`. The caller of `GetProperty` is responsible for releasing this, for example, by calling `CGColorSpaceRelease`. Only supported on Mac OS X.

  Available in Mac OS X v10.3 and later.

  Declared in `ImageCompression.h`.

`kICMImageDescriptionPropertyID_ICCProfile`
: A CFDataRef containing the serialized ICC profile described by the image description. The caller of `GetProperty` is responsible for releasing this, for example, by calling `CFRelease`.

  Available in Mac OS X v10.3 and later.

  Declared in `ImageCompression.h`.

`kICMImageDescriptionPropertyID_FieldInfo`
: Information about the number and order of fields, if available.

  Available in Mac OS X v10.3 and later.

  Declared in `ImageCompression.h`.

`kICMImageDescriptionPropertyID_ClassicTrackWidth`
: A track width suitable for passing to `NewMovieTrack` when creating a new track to hold this image data.

  Available in Mac OS X v10.3 and later.

  Declared in `ImageCompression.h`.

`kICMImageDescriptionPropertyID_ClassicTrackHeight`
: A track height suitable for passing to `NewMovieTrack` when creating a new track to hold this image data.

  Available in Mac OS X v10.3 and later.

  Declared in `ImageCompression.h`.

`kICMImageDescriptionPropertyID_GammaLevel`
: A Fixed for the gammma level described by the image description.

  Available in Mac OS X v10.3 and later.

  Declared in `ImageCompression.h`.

`kICMImageDescriptionPropertyID_RowBytes`
: The offset in bytes from the start of one row to the next. Only valid if the codec type is a chunky pixel format.

  Available in Mac OS X v10.3 and later.

  Declared in `ImageCompression.h`.

`kICMImageDescriptionPropertyID_StepDuration`
: Defines a duration for quantizing time. This is applicable for cases where a single media sample generates visual output that varies continuously through its duration. By interpreting this property, such a sample may be considered to have internal "step points" at multiples of the stepping duration. This can be used to throttle frame generation during playback, and when stepping using InterestingTime APIs. Setting a step duration with value zero removes any current step duration.

  Available in Mac OS X v10.3 and later.

  Declared in `ImageCompression.h`.

`kICMImageDescriptionPropertyID_CleanApertureClipRect`
: The clean aperture as a FixedRect in source coordinates, within the rectangle defined by the image description width and height, suitable for use as a source rectangle in a decompression sequence. For historical reasons, the DVCPROHD codecs store the production aperture display dimensions in the image description width and height; the actual encoded dimensions are smaller. For DVCPROHD, the clip rect will be relative to the image description width and height, not the encoded dimensions.

  Available in Mac OS X v10.3 and later.

  Declared in `ImageCompression.h`.

`kICMImageDescriptionPropertyID_CleanApertureMatrix`
: A matrix transforming the clean aperture clip rect to the origin, scaled to the clean aperture display dimensions. For historical reasons, the DVCPROHD codecs store the production aperture display dimensions in the image description width and height; the actual encoded dimensions are smaller. For DVCPROHD, the matrix will be relative to the image description width and height, not the encoded dimensions.

  Available in Mac OS X v10.3 and later.

  Declared in `ImageCompression.h`.

`kICMImageDescriptionPropertyID_ProductionApertureMatrix`
: A matrix transforming the image to the origin, scaled to the production aperture display dimensions. For historical reasons, the DVCPROHD codecs store the production aperture display dimensions in the image description width and height; the actual encoded dimensions are smaller. For DVCPROHD, the matrix will be relative to the image description width and height, not the encoded dimensions.

Available in Mac OS X v10.3 and later.

Declared in `ImageCompression.h`.

`kICMImageDescriptionPropertyID_SummaryString`
: A localized, human readable string summarizing the image as a CFString, that is, "Apple DV, 720 x 480 (640 x 480), Millions". The elements are: the codec name, the encoded pixels dimensions, then parenthetically the clean aperture mode dimensions, but only if they are different from the encoded pixels dimensions; then the depth. The codec name shall be from the localized decompressor component name string if exactly one decompressor with the correct cType is available; otherwise the string in the image description shall be used. The caller of GetProperty is responsible for releasing this CFString, for example, by calling CFRelease.

Available in Mac OS X v10.3 and later.

Declared in `ImageCompression.h`.

Declared In
`ImageCompression.h`

kQTPropertyClass_DVCompressor

The following are DV Compressor component properties in QuickTime 7.1.

```
enum {
    kQTPropertyClass_DVCompressor = 'dvco',
    kDVCompressorPropertyID_ProgressiveScan = 'prog',
    kDVCompressorPropertyID_AspectRatio16x9 = '16x9',
};
```

Constants
`kQTPropertyClass_DVCompressor`
: The property class for DV compressors. (Applicable to DV25, DV50, NTSC, PAL, PROPAL.)

Available in Mac OS X v10.3 and later.

Declared in `ImageCodec.h`.

`kDVCompressorPropertyID_ProgressiveScan`
: If set, indicates that the compressed frames should be marked as progressive-scan. By default, this flag is clear, meaning that frames should be marked as interlaced.

Available in Mac OS X v10.3 and later.

Declared in `ImageCodec.h`.

`kDVCompressorPropertyID_AspectRatio16x9`
: If set, indicates that the compressor should use a 16:9 picture aspect ratio. If clear, the compressor will use the default 4:3 picture aspect ratio.

Available in Mac OS X v10.3 and later.

Declared in `ImageCodec.h`.

Declared In
`ImageCodec.h`

kQTVisualPropertyID_ApertureMode

The following are visual properties of movies for aperture modes in QuickTime 7.1.

```
enum {
    kQTVisualPropertyID_ApertureMode = 'apmd',
};
```

Constants

`kQTVisualPropertyID_ApertureMode`

You can set the aperture mode property on a movie to indicate whether aspect ratio and clean aperture correction should be performed (`kQTPropertyClass_Visual` / `kQTVisualPropertyID_ApertureMode`). When a movie is in clean, production or encoded pixels aperture mode, each track's dimensions are overriden 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. You can set the aperture mode property on a decompression session options object to indicate whether pixel buffers should be tagged to enable aspect ratio and clean aperture correction (`kQTPropertyClass_ICMDecompressionSessionOptions` / `kICMDecompressionSessionOptionsPropertyID_ApertureMode`).

Available in Mac OS X v10.3 and later.

Declared in `Movies.h`.

Declared In

`Movies.h`

Aperture Mode Properties

You can set the aperture mode property on a movie to indicate whether aspect ratio and clean aperture correction should be performed (`kQTPropertyClass_Visual/kQTVisualPropertyID_ApertureMode`). When a movie is in clean, production or encoded pixels aperture mode, each track's dimensions are overriden 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. You can set the aperture mode property on a decompression session options object to indicate whether pixel buffers should be tagged to enable aspect ratio and clean aperture correction (`kQTPropertyClass_ICMDecompressionSessionOptions` / `kICMDecompressionSessionOptionsPropertyID_ApertureMode`). The movie will be recalculated (laid out again) after one of the following happens: the movie's aperture mode is changed; a track property corresponding to the current aperture mode is changed (that is, `kQTVisualPropertyID_EncodedWidth` for `kQTApertureMode_EncodedPixels`). Track and movie matrices will stay constant unless explicitly changed. The aperture mode of a movie can be set during `NewMovieFromProperties`, and set/get using `SetMovieProperties` and `GetMovieProperties`. `NewMovieFromProperties` defaults to `kQTApertureMode_Classic`. Decompression session options also support this property.

```
enum {
    kQTApertureMode_Classic = 'clas',
    kQTApertureMode_CleanAperture = 'clea',
    kQTApertureMode_ProductionAperture = 'prod',
    kQTApertureMode_EncodedPixels = 'enco'
};
```

Constants

`kQTApertureMode_Classic`

An aperture mode which gives compatibility with behavior in QuickTime 7.0.x and earlier. A movie in classic aperture mode uses track dimensions as set in `NewMovieTrack` and `SetTrackDimensions`. A decompression session in classic aperture mode does not set the clean aperture or pixel aspect ratio attachments on emitted pixel buffers. Movies default to classic aperture mode. If you call `SetTrackDimensions` on a track, the movie is automatically switched into classic aperture mode.

Available in Mac OS X v10.3 and later.

Declared in `ImageCompression.h`.

`kQTApertureMode_CleanAperture`

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 its `kQTVisualPropertyID_CleanApertureDimensions`. A decompression session in clean aperture mode sets the clean aperture and pixel aspect ratio attachments on emitted pixel buffers based on the image description.

Available in Mac OS X v10.3 and later.

Declared in `ImageCompression.h`.

`kQTApertureMode_ProductionAperture`

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 its `kQTVisualPropertyID_ProductionApertureDimensions`. A decompression session in production aperture mode sets the pixel aspect ratio attachments on emitted pixel buffers based on the image description.

Available in Mac OS X v10.3 and later.

Declared in `ImageCompression.h`.

`kQTApertureMode_EncodedPixels`

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 its `kQTVisualPropertyID_EncodedPixelsDimensions`. A decompression session in encoded pixels aperture mode does not set the clean aperture or pixel aspect ratio attachments on emitted pixel buffers.

Available in Mac OS X v10.3 and later.

Declared in `ImageCompression.h`.

Declared In

`ImageCompression.h`

Track Aperture Mode Dimension Properties

The following are the vsual properties of tracks for aperture modes. A track's dimensions may vary depending on the movie's aperture mode. The dimensions for a given aperture mode may be accessed using these properties.

```
enum {
    kQTVisualPropertyID_ClassicDimensions = 'cldi',
    kQTVisualPropertyID_CleanApertureDimensions = 'cadi',
    kQTVisualPropertyID_ProductionApertureDimensions = 'prdi',
    kQTVisualPropertyID_EncodedPixelsDimensions = 'endi',
    kQTVisualPropertyID_HasApertureModeDimensions = 'hamd',
};
```

Constants

`kQTVisualPropertyID_ClassicDimensions`
The track dimensions used in QuickTime 7.0.x and earlier. Setting this property is equivalent to calling SetTrackDimensions, except that SetTrackDimensions also changes the aperture mode to `kQTApertureMode_Classic`, and setting this property does not.

Available in Mac OS X v10.3 and later.

Declared in `Movies.h`.

`kQTVisualPropertyID_CleanApertureDimensions`
The track dimensions to use in clean aperture mode.

Available in Mac OS X v10.3 and later.

Declared in `Movies.h`.

`kQTVisualPropertyID_ProductionApertureDimensions`
The track dimensions to use in production aperture mode.

Available in Mac OS X v10.3 and later.

Declared in `Movies.h`.

`kQTVisualPropertyID_EncodedPixelsDimensions`
The track dimensions to use in encoded pixels aperture mode.

Available in Mac OS X v10.3 and later.

Declared in `Movies.h`.

`kQTVisualPropertyID_HasApertureModeDimensions`
True if aperture mode dimensions have been set on this movie, even if they are all identical to the classic dimensions (as is the case for content with square pixels and no edge processing region). This property can also be tested on a movie, where it is true if any track has aperture mode dimensions.

Available in Mac OS X v10.3 and later.

Declared in `Movies.h`.

Declared In
`Movies.h`

kCharacteristicSupportsApertureModes

Defines media characteristics for aperture modes.

```
enum {
    kCharacteristicSupportsApertureModes = 'apmd',
}
```

Constants

`kCharacteristicSupportsApertureModes`

Indicates that a media handler supports aperture modes, which enable video to be automatically scaled and cropped to compensate for non-square pixel aspect ratios and to trim possibly-dirty edge processing regions. The dimensions of such a track may change when the movie's aperture mode is changed.

Available in Mac OS X v10.3 and later.

Declared in `Movies.h`.

Declared In

`Movies.h`.

kQTSpecialScalingMode_FitWithinDimensions

Defines scaling modes.

```
enum {
    kQTSpecialScalingMode_FitWithinDimensions = 'fit '
};
```

Constants

`kQTSpecialScalingMode_FitWithinDimensions`

Adjusts destination dimensions so that the source fits within the dimensions specified with `kQTSettingsImageWidth` and `kQTSettingsImageHeight` by fitting to the shortest side, and scales the source to the destination. Internally, the default scaling mode, which is based on the source aperture mode, is used for compression session, instead of this scaling mode..

Available in Mac OS X v10.3 and later.

Declared in `QuickTimeComponents.h`.

Declared In

`QuickTimeComponents.h`

QT Settings for Video, Image, Clean Aperture, and Pixel Aspect Ratio

Defines settings for video and image sizes, clean aperture and pixel aspect ratios.

```
enum {
     kQTSettingsVideoSize          = 'isiz'
     kQTSettingsImageWidth         = 'iwdt'
     QTSettingsImageHeight         = 'ihgt'
     kQTSettingsCleanAperture      = 'clap'
     kQTSettingsPixelAspectRatio   = 'pasp'
     kQTSettingsScalingMode        = 'scam'
     kQTSettingsUseCodecEnforcedDimensions = 'uenf'
     kQTSettingsDeinterlaceSource  = 'dint'
};
```

Constants

`kQTSettingsVideoSize`

The video size-related container.

Available in Mac OS X v10.3 and later.

Declared in `QuickTimeComponents.h`.

`kQTSettingsImageWidth`

The destination width. If this is zero, it means the source width.

Available in Mac OS X v10.3 and later.

Declared in `QuickTimeComponents.h`.

`QTSettingsImageHeight`

The destination height. If this is zero, it means the source height.

`kQTSettingsCleanAperture`

The clean aperture for compression sessions. If this is all zeros, it means no clean aperture (that is, full width and height).

Available in Mac OS X v10.3 and later.

Declared in `QuickTimeComponents.h`.

`kQTSettingsPixelAspectRatio`

The pixel aspect ratio for compression sessions. If this is all zeros, it means square pixels (that is, 1:1).

Available in Mac OS X v10.3 and later.

Declared in `QuickTimeComponents.h`.

`kQTSettingsScalingMode`

The scaling mode for compression sessions. If this is zero, it means scaling mode based on the source aperture mode.

Available in Mac OS X v10.3 and later.

Declared in `QuickTimeComponents.h`.

`kQTSettingsUseCodecEnforcedDimensions`

If `TRUE`, compressor's enforced dimension overrides the image size settings.

Available in Mac OS X v10.3 and later.

Declared in `QuickTimeComponents.h`.

`kQTSettingsDeinterlaceSource`

If `TRUE`, deinterlacing is applied to source frames.

Available in Mac OS X v10.3 and later.

Declared in `QuickTimeComponents.h`.

Declared In

`QuickTimeComponents.h`

kICMDecompressionSessionOptionsPropertyID_ApertureMode

Defines properties of decompression session options objects.

```
enum {
    kICMDecompressionSessionOptionsPropertyID_ApertureMode = 'apmd', // OSType,
Read/Write
};
```

Constants

`kICMDecompressionSessionOptionsPropertyID_ApertureMode`

You can set the aperture mode property on a decompression session options object to indicate whether pixel buffers should be tagged to enable aspect ratio and clean aperture correction. The default aperture mode for a decompression session is clean aperture mode.

Available in Mac OS X v10.3 and later.

Declared in `ImageCompression.h`.

Declared In

`ImageCompression.h`

kICMCompressionSessionOptionsPropertyID_ExtraAspectRatioStretchFactor

Defines properties of compression sessions options objects.

```
enum {
    kICMCompressionSessionOptionsPropertyID_ExtraAspectRatioStretchFactor =
'exsf', // Fixed, Default fixed1, Read/Write
};
```

Constants

`kICMCompressionSessionOptionsPropertyID_ExtraAspectRatioStretchFactor`

Requests additional distortion to be applied to the aspect ratio in the `kICMScalingMode_Letterbox` and `kICMScalingMode_Trim` scaling modes. Values greater than fixed1 mean wider, values less than fixed1 mean narrower. For example, a value of X2Fix(2.0) would make the picture aspect ratio twice as wide.

Available in Mac OS X v10.3 and later.

Declared in `ImageCompression.h`.

Declared In

`ImageCompression.h`

Aperture scaling modes

These constants indicate how source frames to a compression session should be scaled if the dimensions and/or display aspect ratio do not match.

```
enum {
    kICMScalingMode_StretchCleanAperture = 'sc2c',
    kICMScalingMode_StretchProductionAperture = 'sp2p',
    kICMScalingMode_Letterbox = 'lett',
    kICMScalingMode_Trim = 'trim'
};
```

Constants

`kICMScalingMode_StretchCleanAperture`
The clean aperture of the source frames is scaled to the clean aperture of the destination.

Available in Mac OS X v10.3 and later.

Declared in `ImageCompression.h`.

`kICMScalingMode_StretchProductionAperture`
The full width and height of source frames is scaled to the full width and height of the destination. This is the default if no other scaling mode is specified.

Available in Mac OS X v10.3 and later.

Declared in `ImageCompression.h`.

`kICMScalingMode_Letterbox`
The clean aperture of the source frames is scaled to fit inside the clean aperture of the destination, preserving the original display aspect ratio. If the display aspect ratios are different, the source frames are centered with black bars above and below, or to the left and right.

Available in Mac OS X v10.3 and later.

Declared in `ImageCompression.h`.

`kICMScalingMode_Trim`
The clean aperture of the source frames is scaled to cover the clean aperture of the destination, preserving the original display aspect ratio. If the display aspect ratios are different, the source frames are centered and cropped.

Available in Mac OS X v10.3 and later.

Declared in `ImageCompression.h`.

Declared In
`ImageCompression.h`

TrackApertureModeDimensionsAID

The type of atom that stores information for video correction in the movie resource. A child of the `'track'` atom.

```
 enum {
    TrackApertureModeDimensionsAID          = 'tapt',
    TrackCleanApertureDimensionsAID         = 'clef',
    TrackProductionApertureDimensionsAID    = 'prof',
    TrackEncodedPixelsDimensionsAID         = 'enof',
};
```

Constants

`TrackApertureModeDimensionsAID`
The container atom.

Available in Mac OS X v10.3 and later.

Declared in `MoviesFormat.h`.

`TrackCleanApertureDimensionsAID`
The container atom.

Available in Mac OS X v10.3 and later.

Declared in `MoviesFormat.h`.

`TrackProductionApertureDimensionsAID`
The container atom.

Available in Mac OS X v10.3 and later.

Declared in `MoviesFormat.h`.

`TrackEncodedPixelsDimensionsAID`
The container atom.

Available in Mac OS X v10.3 and later.

Declared in `MoviesFormat.h`.

Declared In
`MoviesFormat.h`.

kQTPropertyClass_ImageCompressor

The following is a property for image compressor components.

```
enum {
kQTPropertyClass_ImageCompressor = 'imco',
};
```

Constants
`kQTPropertyClass_ImageCompressor`
Property class for image compressor components.

Available in Mac OS X v10.3 and later.

Declared in `ImageCodec.h`.

Declared In
`ImageCodec.h`

movieExportSourceApertureMode

The following sets the aperture mode on the decompression session.

```
enum {
 movieExportSourceApertureMode = 'srap',
};
```

Constants
`movieExportSourceApertureMode`
A pointer to an OSType. The source movie's aperture mode.

Available in Mac OS X v10.3 and later.

Declared in `QuickTimeComponents.h`.

Declared In
`QuickTimeComponents.h`

kICMImageCompressorPropertyID_Enforced

The following are enforced properties (introduced in QuickTime 7.0.4) for image compressor components. Image compressors that sometimes or always restrict image dimensions, clean aperture and/or pixel aspect ratio should support these properties. If these properties can change dynamically for a compressor (for example, in response to user interaction) then the properties should be listenable, and the compressor should call the listeners whenever the properties change. (In this case, the component's `GetComponentPropertyInfo` function should set the `kComponentPropertyFlagWillNotifyListeners` flag.) If a compressor has a mode in which these properties are flexible, then when the component is in that mode, (a) the component's `GetComponentProperty` function should return `kQTPropertyAskLaterErr` for these properties, and (b) the component's `GetComponentPropertyInfo` function should set the `kComponentPropertyFlagCanGetLater` flag for these properties.

```
enum {
    kICMImageCompressorPropertyID_EnforcedEncodedWidth    = 'enwi',
    kICMImageCompressorPropertyID_EnforcedEncodedHeight   = 'enhe',
    kICMImageCompressorPropertyID_EnforcedCleanAperture   = 'encl',
    kICMImageCompressorPropertyID_EnforcedPixelAspectRatio = 'enpa',
    kICMImageCompressorPropertyID_EnforcedFieldInfo       = 'enfi',
};
```

Constants

`kICMImageCompressorPropertyID_EnforcedEncodedWidth`
The encoded width enforced for compressed frames.
Available in Mac OS X v10.3 and later.
Declared in `ImageCodec.h`.

`kICMImageCompressorPropertyID_EnforcedEncodedHeight`
The encoded height enforced for compressed frames.
Available in Mac OS X v10.3 and later.
Declared in `ImageCodec.h`.

`kICMImageCompressorPropertyID_EnforcedCleanAperture`
The clean aperture enforced for compressed frames.
Available in Mac OS X v10.3 and later.
Declared in `ImageCodec.h`.

`kICMImageCompressorPropertyID_EnforcedPixelAspectRatio`
The pixel aspect ratio enforced for compressed frames.
Available in Mac OS X v10.3 and later.
Declared in `ImageCodec.h`.

`kICMImageCompressorPropertyID_EnforcedFieldInfo`
The number and order of fields enforced for compressed frames.
Available in Mac OS X v10.3 and later.
Declared in `ImageCodec.h`.

Declared In
`ImageCodec.h`

Audio Properties

The following is a list of the new audio properties available in QuickTime 7.1.

```
enum {
    kQTAudioRenderQuality_PlaybackDefault = 0x8000,
    kQTAudioPropertyID_RenderQuality = 'qual',
    kQTMovieAudioExtractionAudioPropertyID_RenderQuality = 'qual',
    kQTAudioPropertyID_DeviceASBD = 'dasd',
    kQTAudioPropertyID_SummaryASBD = 'sasd',
    kQTAudioPropertyID_RateChangesPreservePitch = 'aucp',
    kQTAudioPropertyID_Pitch  = 'pitc',
    kQTSCAudioPropertyID_RenderQuality = 'qlty',
};
```

Constants

`kQTAudioPropertyID_DeviceASBD`

This is a get-only property and returns the `AudioStreamBasicDescription` of the device the movie is playing to. The interesting fields are the sample rate, which reflects device's current state, and the number of channels, which matches what is reported by `kQTAudioPropertyID_DeviceChannelLayout`.

`kQTAudioPropertyID_SummaryASBD`

Get-only. Returns the `AudioStreamBasicDescription` corresponding to the Summary Mix of a movie. This describes non-interleaved, `Float32` linear PCM data, with a sample rate equal to the highest audio sample rate found among the sound tracks contributing to the AudioContext mix, and a number of channels that matches what is reported by `kQTAudioPropertyID_SummaryChannelLayout`.

`kQTAudioPropertyID_RateChangesPreservePitch`

This property was introduced in QuickTime 7 and must be set in order for pitch changes to take effect. 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 may be used, which alters the speed without changing the pitch. Because this is more computationally expensive, this property may be silently ignored on some slow CPUs. Media handlers may query this movie property and honor it when performing Scaled Edits. This property can be specified as a property to the `NewMovieFromProperties` API. Currently, it has no effect when set on an open movie.

`kQTAudioPropertyID_Pitch`

The movie pitch adjustment. This adjusts the pitch of all audio tracks that contribute to the AudioContext mix. Pitch control takes effect only if `kQTAudioPropertyID_RateChangesPreservePitch` is in effect; otherwise, returns `kQTMessageNotHandledErr`. The `Float32` value is specified in cents: 0.0 == no change, 1.0 == one cent up, 100.0 == one semi-tone up, -1.0 == one cent down. The most useful ranges for pitch are +/- 1200, that is, one octave.

`kQTAudioPropertyID_RenderQuality`

Movie audio render quality takes effect for movie playback. `UInt32` values are as defined in AudioUnit/AudioUnitProperties.h and vary from 0x00 (`kRenderQuality_Min`) to 0x7F (`kRenderQuality_Max`). A special value `kQTAudioRenderQuality_PlaybackDefault`) is also defined which resets the quality settings of the playback processing chain to values that are chosen to be an optimal balance of performance and quality.

`kQTMovieAudioExtractionAudioPropertyID_RenderQuality`

Sets the render quality to be used for this audio extraction session. `UInt32` values are as defined in AudioUnit/AudioUnitProperties.h and vary from 0x00(`kRenderQuality_Min`) to 0x7F (`kRenderQuality_Max`). A special value (`kQTAudioRenderQuality_PlaybackDefault`) is also defined which resets the quality settings to the same values that were chosen by default for playback.

`kQTSCAudioPropertyID_RenderQuality`
: Specifies the quality with which QuickTime should render the audio stream during the compression/decompression/transcode operation. Accepted constants are those used by AudioUnits (defined in AudioUnitProperties.h): `kRenderQuality_Max`, `kRenderQuality_High`, `kRenderQuality_Medium`, `kRenderQuality_Low`, `kRenderQuality_Min`. Available in QuickTimeComponents.h.

Declared In
`Movies.h`

Metadata Constants

The following is a list of new metadata constants available in QuickTime 7.1:

```
enum {
    kQTMetaDataCommonKeyAlbum     = 'albm',
    kQTMetaDataCommonKeyArtist    = 'arts',
    kQTMetaDataCommonKeyArtwork   = 'artw',
    kQTMetaDataCommonKeyChapterName = 'chap',
    kQTMetaDataCommonKeyComposer  = 'comp',
    kQTMetaDataCommonKeyDescription = 'desc',
    kQTMetaDataCommonKeyGenre     = 'genr',
    kQTMetaDataCommonKeyOriginalFormat = 'orif',
    kQTMetaDataCommonKeyOriginalSource = 'oris',
    kQTMetaDataCommonKeyPerformers = 'perf',
    kQTMetaDataCommonKeySoftware  = 'soft',
    kQTMetaDataCommonKeyWriter    = 'wrtr'
};
```

Constants
`kQTMetaDataCommonKeyAlbum`
: Information to come.

Declared In
`Movies.h.`

Document Revision History

This table describes the changes to *QuickTime 7.1 Update Reference*.

Date	Notes
2006-08-14	New document that describes the new functions available in QuickTime 7.1.

Document Revision History

Index

M

Q

R

S

T

V