Image Compression Manager Reference

QuickTime > Compression & Decompression

2006-05-23

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

Contents

Image Compression Manager Reference

Framework: Frameworks/QuickTime.framework

Declared in ImageCompression.h

Overview

Applications can use the QuickTime image compression APIs to compress and decompress sounds, images, and image sequences, as well as to transcode sounds and images between compression formats.

Functions by Task

Image Transcoder Support

`ImageTranscoderBeginSequence` (page 99)
Initiates an image transcoding sequence and specifies the input data format.

`ImageTranscoderConvert` (page 100)
Performs image transcoding operations.

`ImageTranscoderDisposeData` (page 101)
Disposes of transcoded data.

`ImageTranscoderEndSequence` (page 102)
Ends an image transcoding sequence.

Managing an ICM Compression Session

`ICMCompressionSessionCompleteFrames` (page 30)
Forces a compression session to complete encoding frames.

`ICMCompressionSessionCreate` (page 31)
Creates a compression session for a specified codec type.

`ICMCompressionSessionEncodeFrame` (page 33)
Presents video frames to a compression session.

`ICMCompressionSessionGetImageDescription` (page 34)
Retrieves the image description for a video compression session.

`ICMCompressionSessionGetPixelBufferPool` (page 35)
Returns a pool that can provide ideal source pixel buffers for a compression session.

`ICMCompressionSessionGetProperty` (page 36)
Retrieves the value of a specific property of a compression session.

Using the OpenGL Texture Context

`QTOpenGLTextureContextCreate` (page 117)
Creates a new OpenGL texture context for a specified OpenGL context and pixel format.

`QTVisualContextCopyImageForTime` (page 121)
Retrieves an image buffer from the visual context, indexed by the provided time.

`QTVisualContextGetAttribute` (page 121)
Returns a visual context attribute.

`QTVisualContextGetTypeID` (page 122)
Returns the CFTypeID for QTVisualContextRef.

`QTVisualContextIsNewImageAvailable` (page 122)
Queries whether a new image is available for a given time.

`QTVisualContextRelease` (page 123)
Releases a visual context object.

`QTVisualContextRetain` (page 124)
Retains a visual context object.

`QTVisualContextSetAttribute` (page 124)
Sets a visual context attribute.

`QTVisualContextSetImageAvailableCallback` (page 125)
Installs a user-defined callback to receive notifications when a new image becomes available.

`QTVisualContextTask` (page 125)
Causes visual context to release internally held resources for later re-use.

Supporting Functions

`DisposeICMAlignmentUPP` (page 17)
Disposes of an ICMAlignmentUPP pointer.

`DisposeICMCompletionUPP` (page 18)
Disposes of an ICMCompletionUPP pointer.

`DisposeICMConvertDataFormatUPP` (page 18)
Disposes of an ICMConvertDataFormatUPP pointer.

`DisposeICMCursorShieldedUPP` (page 19)
Disposes of an ICMCursorShieldedUPP pointer.

`DisposeICMDataUPP` (page 19)
Disposes of an ICMDataUPP pointer.

`DisposeICMFlushUPP` (page 20)
Disposes of an ICMFlushUPP pointer.

`DisposeICMMemoryDisposedUPP` (page 20)
Disposes of an ICMMemoryDisposedUPP pointer.

ICMEncodedFrameSetSimilarity (page 91)
Sets the similarity for an encoded frame.

ICMEncodedFrameSetValidTimeFlags (page 91)
Sets an encoded frame's flags that indicate which of its time stamps and durations are valid.

ICMImageDescriptionGetProperty (page 92)
Returns a particular property of a image description handle.

ICMImageDescriptionGetPropertyInfo (page 93)
Returns information about a particular property of a image description.

ICMImageDescriptionSetProperty (page 93)
Sets a particular property of a image description handle.

ICMMultiPassStorageCopyDataAtTimeStamp (page 94)
Called by a multipass-capable compressor to retrieve data at a given time stamp.

ICMMultiPassStorageCreateWithCallbacks (page 95)
Assembles a multipass storage mechanism from callbacks.

ICMMultiPassStorageCreateWithTemporaryFile (page 95)
Creates multipass storage using a temporary file.

ICMMultiPassStorageGetTimeStamp (page 96)
Called by a multipass-capable compressor to retrieve a time stamp for which a value is stored.

ICMMultiPassStorageGetTypeID (page 97)
Returns the type ID for the current multipass storage object.

ICMMultiPassStorageRelease (page 97)
Decrements the retain count of a multipass storage object.

ICMMultiPassStorageRetain (page 98)
Increments the retain count of a multipass storage object.

ICMMultiPassStorageSetDataAtTimeStamp (page 98)
Called by a multipass-capable compressor to store data at a given time stamp.

NewICMAlignmentUPP (page 102)
Allocates a Universal Procedure Pointer for the ICMAlignmentProc callback.

NewICMCompletionUPP (page 103)
Allocates a Universal Procedure Pointer for the ICMCompletionProc callback.

NewICMConvertDataFormatUPP (page 103)
Allocates a Universal Procedure Pointer for the ICMConvertDataFormatProc callback.

NewICMCursorShieldedUPP (page 104)
Allocates a Universal Procedure Pointer for the ICMCursorShieldedProc callback.

NewICMDataUPP (page 104)
Allocates a Universal Procedure Pointer for the ICMDataProc callback.

NewICMFlushUPP (page 105)
Allocates a Universal Procedure Pointer for the ICMFlushProc callback.

NewICMMemoryDisposedUPP (page 105)
Allocates a Universal Procedure Pointer for the ICMMemoryDisposedProc callback.

NewICMProgressUPP (page 106)
Allocates a Universal Procedure Pointer for the ICMProgressProc callback.

NewQDPixUPP (page 106)
Allocates a Universal Procedure Pointer for the QDPixProc callback.

Functions

DisposeICMAlignmentUPP

Disposes of an ICMAlignmentUPP pointer.

```
void DisposeICMAlignmentUPP (
   ICMAlignmentUPP userUPP
);
```

Parameters

userUPP
An `ICMAlignmentUPP` pointer. See `Universal Procedure Pointers`.

Return Value

You can access this function's error returns through `GetMoviesError` and `GetMoviesStickyError`.

Version Notes
Introduced in QuickTime 4.1.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`ImageCompression.h`

DisposeICMCompletionUPP

Disposes of an ICMCompletionUPP pointer.

```
void DisposeICMCompletionUPP (
   ICMCompletionUPP userUPP
);
```

Parameters

userUPP

An `ICMCompletionUPP` pointer. See `Universal Procedure Pointers`.

Return Value
You can access this function's error returns through `GetMoviesError` and `GetMoviesStickyError`.

Version Notes
Introduced in QuickTime 4.1.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
qtcompress
qtcompress.win

Declared In
`ImageCompression.h`

DisposeICMConvertDataFormatUPP

Disposes of an ICMConvertDataFormatUPP pointer.

```
void DisposeICMConvertDataFormatUPP (
   ICMConvertDataFormatUPP userUPP
);
```

Parameters

userUPP

An `ICMConvertDataFormatUPP` pointer. See `Universal Procedure Pointers`.

Return Value
You can access this function's error returns through `GetMoviesError` and `GetMoviesStickyError`.

Version Notes
Introduced in QuickTime 4.1.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`ImageCompression.h`

DisposeICMCursorShieldedUPP

Disposes of an ICMCursorShieldedUPP pointer.

```
void DisposeICMCursorShieldedUPP (
   ICMCursorShieldedUPP userUPP
);
```

Parameters

userUPP

An `ICMCursorShieldedUPP` pointer. See `Universal Procedure Pointers`.

Return Value
You can access this function's error returns through `GetMoviesError` and `GetMoviesStickyError`.

Version Notes
Introduced in QuickTime 4.1.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`ImageCompression.h`

DisposeICMDataUPP

Disposes of an ICMDataUPP pointer.

```
void DisposeICMDataUPP (
   ICMDataUPP userUPP
);
```

Parameters

userUPP

An `ICMDataUPP` pointer. See `Universal Procedure Pointers`.

Return Value
You can access this function's error returns through `GetMoviesError` and `GetMoviesStickyError`.

Version Notes
Introduced in QuickTime 4.1.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`ImageCompression.h`

DisposeICMFlushUPP

Disposes of an ICMFlushUPP pointer.

```
void DisposeICMFlushUPP (
   ICMFlushUPP userUPP
);
```

Parameters

userUPP

An `ICMFlushUPP` pointer. See `Universal Procedure Pointers`.

Return Value
You can access this function's error returns through `GetMoviesError` and `GetMoviesStickyError`.

Version Notes
Introduced in QuickTime 4.1.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`ImageCompression.h`

DisposeICMMemoryDisposedUPP

Disposes of an ICMMemoryDisposedUPP pointer.

```
void DisposeICMMemoryDisposedUPP (
   ICMMemoryDisposedUPP userUPP
);
```

Parameters

userUPP

An `ICMMemoryDisposedUPP` pointer. See `Universal Procedure Pointers`.

Return Value
You can access this function's error returns through `GetMoviesError` and `GetMoviesStickyError`.

Version Notes
Introduced in QuickTime 4.1.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`ImageCompression.h`

DisposeICMProgressUPP

Disposes of an ICMProgressUPP pointer.

```
void DisposeICMProgressUPP (
   ICMProgressUPP userUPP
);
```

Parameters

userUPP

An `ICMProgressUPP` pointer. See `Universal Procedure Pointers`.

Return Value
You can access this function's error returns through `GetMoviesError` and `GetMoviesStickyError`.

Version Notes
Introduced in QuickTime 4.1.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
qtdataexchange
qtdataexchange.win

Declared In
`ImageCompression.h`

DisposeQDPixUPP

Disposes of a QDPixUPP pointer.

```
void DisposeQDPixUPP (
   QDPixUPP userUPP
);
```

Parameters

userUPP

A `QDPixUPP` pointer. See `Universal Procedure Pointers`.

Return Value
You can access this function's error returns through `GetMoviesError` and `GetMoviesStickyError`.

Version Notes
Introduced in QuickTime 4.1.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`ImageCompression.h`

DisposeStdPixUPP

Disposes of a StdPixUPP pointer.

```
void DisposeStdPixUPP (
   StdPixUPP userUPP
);
```

Parameters

userUPP

A `StdPixUPP` pointer. See `Universal Procedure Pointers`.

Return Value
You can access this function's error returns through `GetMoviesError` and `GetMoviesStickyError`.

Version Notes
Introduced in QuickTime 4.1.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
Desktop Sprites
DesktopSprites
DesktopSprites.win

Declared In
`ImageCompression.h`

ICMCompressionFrameOptionsCreate

Creates a frame compression options object.

```
OSStatus ICMCompressionFrameOptionsCreate (
   CFAllocatorRef allocator,
   ICMCompressionSessionRef session,
   ICMCompressionFrameOptionsRef *options
);
```

Parameters

allocator

An allocator. Pass NULL to use the default allocator.

session

A compression session reference. This reference is returned by `ICMCompressionSessionCreate` (page 31).

options

On return, a reference to a new frame compression options object.

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

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

Declared In
`ImageCompression.h`

ICMCompressionFrameOptionsCreateCopy

Copies a frame compression options object.

```
OSStatus ICMCompressionFrameOptionsCreateCopy (
   CFAllocatorRef allocator,
   ICMCompressionFrameOptionsRef originalOptions,
   ICMCompressionFrameOptionsRef *copiedOptions
);
```

Parameters

allocator

An allocator. Pass NULL to use the default allocator.

originalOptions

A frame compression options reference. This reference is returned by `ICMCompressionFrameOptionsCreate`.

copiedOptions

On return, a reference to a copy of the frame compression options object passed in `originalOptions`.

Return Value

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

Availability

Available in Mac OS X v10.3 and later.

Declared In

`ImageCompression.h`

ICMCompressionFrameOptionsGetForceKeyFrame

Retrieves the force key frame flag.

```
Boolean ICMCompressionFrameOptionsGetForceKeyFrame (
   ICMCompressionFrameOptionsRef options
);
```

Parameters

options

A compression frame options reference. This reference is returned by `ICMCompressionFrameOptionsCreate`.

Return Value

Returns TRUE if frames are forced to be compressed as key frames, FALSE otherwise.

Availability

Available in Mac OS X v10.3 and later.

Related Sample Code

ExampleIPBCodec

Declared In

`ImageCompression.h`

ICMCompressionFrameOptionsGetFrameType

Retrieves the frame type setting.

```
ICMFrameType ICMCompressionFrameOptionsGetFrameType (
   ICMCompressionFrameOptionsRef options
);
```

Parameters

options

A compression frame options reference. This reference is returned by `ICMCompressionFrameOptionsCreate`.

Return Value
On return, one of the `frame` types listed below.

Discussion
This function can return one of these constants:

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

Related Sample Code
ExampleIPBCodec

Declared In
`ImageCompression.h`

ICMCompressionFrameOptionsGetProperty

Retrieves the value of a specific property of a compression frame options object.

```
OSStatus ICMCompressionFrameOptionsGetProperty (
   ICMCompressionFrameOptionsRef options,
   ComponentPropertyClass inPropClass,
   ComponentPropertyID inPropID,
   ByteCount inPropValueSize,
   ComponentValuePtr outPropValueAddress,
   ByteCount *outPropValueSizeUsed
);
```

Parameters

options

A compression frame options reference. This reference is returned by `ICMCompressionFrameOptionsCreate`.

inPropClass

Pass the following constant to define the property class: `kComponentPropertyClassPropertyInfo = 'pnfo'` The property information class. See these constants:

`kComponentPropertyClassPropertyInfo`

inPropID

Pass one of these constants to define the property ID: `kComponentPropertyInfoList` = `'list'` An array of `CFData` values, one for each property. `kComponentPropertyCacheSeed` = `'seed'` A property cache seed value. `kComponentPropertyCacheFlags` = `'flgs'` One of the `kComponentPropertyCache` flags: `kComponentPropertyCacheFlagNotPersistentProperty` metadata should not be saved in persistent cache. `kComponentPropertyCacheFlagIsDynamicProperty` metadata should not cached at all. `kComponentPropertyExtendedInfo` = `'meta'` A `CFDictionary` with extended property information. See these constants:

```
kComponentPropertyInfoList
kComponentPropertyCacheSeed
kComponentPropertyCacheFlags
kComponentPropertyExtendedInfo
```

outPropType

A pointer to the type of the returned property's value.

outPropValueAddress

A pointer to a variable to receive the returned property's value.

outPropValueSizeUsed

On return, a pointer to the number of bytes actually used to store the property.

Return Value

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

Availability

Available in Mac OS X v10.3 and later.

Declared In

`ImageCompression.h`

ICMCompressionFrameOptionsGetPropertyInfo

Retrieves information about properties of a compression frame options object.

```
OSStatus ICMCompressionFrameOptionsGetPropertyInfo (
   ICMCompressionFrameOptionsRef options,
   ComponentPropertyClass inPropClass,
   ComponentPropertyID inPropID,
   ComponentValueType *outPropType,
   ByteCount *outPropValueSize,
   UInt32 *outPropertyFlags
);
```

Parameters

options

A compression frame options reference. This reference is returned by `ICMCompressionFrameOptionsCreate`.

inPropClass

Pass the following constant to define the property class: `kComponentPropertyClassPropertyInfo` = `'pnfo'` The property information class. See these constants:

```
kComponentPropertyClassPropertyInfo
```

inPropID

Pass one of these constants to define the property ID: `kComponentPropertyInfoList` = `'list'` An array of `CFData` values, one for each property. `kComponentPropertyCacheSeed` = `'seed'` A property cache seed value. `kComponentPropertyCacheFlags` = `'flgs'` One of the `kComponentPropertyCache` flags: `kComponentPropertyCacheFlagNotPersistentProperty` metadata should not be saved in persistent cache. `kComponentPropertyCacheFlagIsDynamicProperty` metadata should not cached at all. `kComponentPropertyExtendedInfo` = `'meta'` A `CFDictionary` with extended property information. See these constants:

```
kComponentPropertyInfoList
kComponentPropertyCacheSeed
kComponentPropertyCacheFlags
kComponentPropertyExtendedInfo
```

outPropType

A pointer to the type of the returned property's value.

outPropValueSize

A pointer to the size of the returned property's value.

outPropFlags

On return, a pointer to flags representing the requested information about the property.

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

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

Declared In
`ImageCompression.h`

ICMCompressionFrameOptionsGetTypeID

Returns the type ID for the current frame compression options object.

```
CFTypeID ICMCompressionFrameOptionsGetTypeID (
   void
);
```

Return Value
A `CFTypeID` value.

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

Declared In
`ImageCompression.h`

ICMCompressionFrameOptionsRelease

Decrements the retain count of a frame compression options object.

```
void ICMCompressionFrameOptionsRelease (
   ICMCompressionFrameOptionsRef options
);
```

Parameters

options

A reference to a frame compression options object. This reference is returned by `ICMCompressionFrameOptionsCreate`. If you pass NULL, nothing happens.

Discussion
If the retain count drops to 0, the object is disposed.

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

Declared In
`ImageCompression.h`

ICMCompressionFrameOptionsRetain

Increments the retain count of a frame compression options object.

```
ICMCompressionFrameOptionsRef ICMCompressionFrameOptionsRetain (
   ICMCompressionFrameOptionsRef options
);
```

Parameters

options

A reference to a frame compression options object. This reference is returned by `ICMCompressionFrameOptionsCreate`. If you pass NULL, nothing happens.

Return Value
A copy of the object reference passed in options, for convenience.

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

Declared In
`ImageCompression.h`

ICMCompressionFrameOptionsSetForceKeyFrame

Forces frames to be compressed as key frames.

```
OSStatus ICMCompressionFrameOptionsSetForceKeyFrame (
   ICMCompressionFrameOptionsRef options,
   Boolean forceKeyFrame
);
```

Parameters

options

A compression frame options reference. This reference is returned by `ICMCompressionFrameOptionsCreate`.

forceKeyFrame

Pass TRUE to force frames to be compressed as key frames, FALSE otherwise.

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

Discussion
The compressor must obey this flag if set. By default it is set FALSE.

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

Declared In
`ImageCompression.h`

ICMCompressionFrameOptionsSetFrameType

Requests a frame be compressed as a particular frame type.

```
OSStatus ICMCompressionFrameOptionsSetFrameType (
   ICMCompressionFrameOptionsRef options,
   ICMFrameType frameType
);
```

Parameters

options

A compression frame options reference. This reference is returned by `ICMCompressionFrameOptionsCreate`.

frameType

A constant that identifies a frame type. Pass one of the following but do not assume that there are no other frame types: `kICMFrameType_I` = `'I'` An I frame. `kICMFrameType_P` = `'P'` A P frame. `kICMFrameType_B` = `'B'` A B frame. `kICMFrameType_Unknown` = 0 A frame of unknown type. See these constants:

`kICMFrameType_I`

`kICMFrameType_P`

`kICMFrameType_B`

`kICMFrameType_Unknown`

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

Discussion
The frame type setting may be ignored by the compressor if it is not appropriate. By default it is set to `kICMFrameType_Unknown`.

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

Declared In
`ImageCompression.h`

ICMCompressionFrameOptionsSetProperty

Sets the value of a specific property of a compression frame options object.

```
OSStatus ICMCompressionFrameOptionsSetProperty (
   ICMCompressionFrameOptionsRef options,
   ComponentPropertyClass inPropClass,
   ComponentPropertyID inPropID,
   ByteCount inPropValueSize,
   ConstComponentValuePtr inPropValueAddress
);
```

Parameters

options

A compression frame options reference. This reference is returned by `ICMCompressionFrameOptionsCreate`.

inPropClass

Pass the following constant to define the property class: `kComponentPropertyClassPropertyInfo` = `'pnfo'` The property information class. See these constants:

`kComponentPropertyClassPropertyInfo`

inPropID

Pass one of these constants to define the property ID: `kComponentPropertyInfoList` = `'list'` An array of `CFData` values, one for each property. `kComponentPropertyCacheSeed` = `'seed'` A property cache seed value. `kComponentPropertyCacheFlags` = `'flgs'` One of the `kComponentPropertyCache` flags: `kComponentPropertyCacheFlagNotPersistentProperty` metadata should not be saved in persistent cache. `kComponentPropertyCacheFlagIsDynamicProperty` metadata should not cached at all. `kComponentPropertyExtendedInfo` = `'meta'` A `CFDictionary` with extended property information. See these constants:

`kComponentPropertyInfoList`

`kComponentPropertyCacheSeed`

`kComponentPropertyCacheFlags`

`kComponentPropertyExtendedInfo`

inPropValueSize

The size of the property value to be set.

inPropValueAddress

A pointer to the value of the property to be set.

Return Value

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

Availability

Available in Mac OS X v10.3 and later.

Declared In

`ImageCompression.h`

ICMCompressionSessionBeginPass

Announces the start of a specific compression pass.

```
OSStatus ICMCompressionSessionBeginPass (
   ICMCompressionSessionRef session,
   ICMCompressionPassModeFlags passModeFlags,
   UInt32 flags
);
```

Parameters

session

A compression session reference. This reference is returned by `ICMCompressionSessionCreate` (page 31).

passModeFlags

Flags that describe how the compressor should behave in this pass of multipass encoding: `kICMCompressionPassMode_OutputEncodedFrames` = 1L<<0 Output encoded frames. `kICMCompressionPassMode_NoSourceFrames` = 1L<<1 The client need not provide source frame buffers. `kICMCompressionPassMode_WriteToMultiPassStorage` = 1L<<2 The compressor may write private data to multipass storage. `kICMCompressionPassMode_ReadFromMultiPassStorage` = 1L<<3 The compressor may read private data from multipass storage. See these constants:

```
kICMCompressionPassMode_OutputEncodedFrames
kICMCompressionPassMode_NoSourceFrames
kICMCompressionPassMode_WriteToMultiPassStorage
kICMCompressionPassMode_ReadFromMultiPassStorage
```

flags

Reserved. Set to 0.

Return Value

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

Discussion

The source frames and frame options for each display time should be the same across passes. During multipass compression, valid `displayTimeStamp` values must be passed to `ICMCompressionSessionEncodeFrame` (page 33), because they are used to index the compressor's stored state.

Availability

Available in Mac OS X v10.3 and later.

Declared In

`ImageCompression.h`

ICMCompressionSessionCompleteFrames

Forces a compression session to complete encoding frames.

```
OSStatus ICMCompressionSessionCompleteFrames (
   ICMCompressionSessionRef session,
   Boolean completeAllFrames,
   TimeValue64 completeUntilDisplayTimeStamp,
   TimeValue64 nextDisplayTimeStamp
);
```

Parameters

session

A reference to a video compression session, returned by a previous call to `ICMCompressionSessionCreate` (page 31).

completeAllFrames

Pass TRUE to direct the session to complete all pending frames.

completeUntilDisplayTimeStamp

A 64-bit time value that represents the display time up to which to complete frames. This value is ignored if `completeAllFrames` is TRUE.

nextDisplayTimeStamp

A 64-bit time value that represents the display time of the next frame that should be passed to `EncodeFrame`. This value is ignored unless `ICMCompressionSessionOptionsSetDurationsNeeded` set TRUE and `kICMValidTime_DisplayDurationIsValid` was 0 in `validTimeFlags` in the last call to `ICMCompressionSessionEncodeFrame` (page 33).

Return Value

Returns an error code, or 0 if there is no error. The function may return before frames are completed if the encoded frame callback routine returns an error.

Discussion

Call this function to force a compression session to complete encoding frames. Set `completeAllFrames` to direct the session to complete all pending frames. If `completeAllFrames` is false, only frames with display time stamps up to and including the time passed in `completeUntilDisplayTimeStamp` will be encoded. If `ICMCompressionSessionOptionsSetDurationsNeeded` set TRUE and you are passing valid display timestamps but not display durations to `ICMCompressionSessionEncodeFrame` (page 33), pass in `nextDisplayTimeStamp` the display timestamp of the next frame that would be passed to `EncodeFrame`.

Availability

Available in Mac OS X v10.3 and later.

Related Sample Code

CaptureAndCompressIPBMovie

OpenGLCaptureToMovie

Quartz Composer QCTV

Declared In

`ImageCompression.h`

ICMCompressionSessionCreate

Creates a compression session for a specified codec type.

```
OSStatus ICMCompressionSessionCreate (
   CFAllocatorRef allocator,
   int width,
   int height,
   CodecType cType,
   TimeScale timescale,
   ICMCompressionSessionOptionsRef compressionOptions,
   CFDictionaryRef sourcePixelBufferAttributes,
   ICMEncodedFrameOutputRecord *encodedFrameOutputRecord,
   ICMCompressionSessionRef *compressionSessionOut
);
```

Parameters

allocator

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

width

The width of frames. Pass 0 to let the compressor control the width.

height

The height of frames. Pass 0 to let the compressor control the height.

cType

The codec type.

timescale

The timescale to be used for all time stamps and durations used in the session.

compressionOptions

A reference to a settings object that configures the session. You create such an object by calling `ICMCompressionSessionOptionsCreate`. You can then use these constants to set its properties:
`kICMUnlimitedFrameDelayCount` No limit on the number of frames in the compression window.
`kICMUnlimitedFrameDelayTime` No time limit on the frames in the compression window.
`kICMUnlimitedCPUTimeBudget` No CPU time limit on compression.

sourcePixelBufferAttributes

Required attributes for source pixel buffers, used when creating a pixel buffer pool for source frames. If you do not want the ICM to create one for you, pass NULL. Using pixel buffers not allocated by the ICM may increase the chance that it will be necessary to copy image data.

encodedFrameOutputRecord

The callback that will receive encoded frames.

compressionSessionOut

Points to a variable to receive the created session object.

Return Value

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

Discussion

Some compressors do not support arbitrary source dimensions, and may override the suggested width and height.

Availability

Available in Mac OS X v10.3 and later.

Related Sample Code

CaptureAndCompressIPBMovie

OpenGLCaptureToMovie

Quartz Composer QCTV

Declared In
`ImageCompression.h`

ICMCompressionSessionEncodeFrame

Presents video frames to a compression session.

```
OSStatus ICMCompressionSessionEncodeFrame (
   ICMCompressionSessionRef session,
   CVPixelBufferRef pixelBuffer,
   TimeValue64 displayTimeStamp,
   TimeValue64 displayDuration,
   ICMValidTimeFlags validTimeFlags,
   ICMCompressionFrameOptionsRef frameOptions,
   ICMSourceTrackingCallbackRecord *sourceTrackingCallback,
   void *sourceFrameRefCon
);
```

Parameters

session

A reference to a video compression session, returned by a previous call to `ICMCompressionSessionCreate` (page 31).

pixelBuffer

A reference to a buffer containing a source image to be compressed, which must have a nonzero reference count. The session will retain it as long as necessary. The client should not modify the pixel buffer's pixels until the pixel buffer release callback is called. In a multipass encoding session pass, where the compressor suggested the flag `kICMCompressionPassMode_NoSourceFrames`, you may pass NULL in this parameter.

displayTimeStamp

A 64-bit time value that represents the display time of the frame, using the time scale passed to `ICMCompressionSessionCreate` (page 31). If you pass a valid value, set the `kICMValidTime_DisplayTimeStampIsValid` flag in the `validTimeFlags` parameter (below).

displayDuration

A 64-bit time value that represents the display duration of the frame, using the time scale passed to `ICMCompressionSessionCreate` (page 31). If you pass a valid value, set the `kICMValidTime_DisplayDurationIsValid` flag in the `validTimeFlags` parameter (below).

validTimeFlags

Flags to indicate which of the values passed in `displayTimeStamp` and `displayDuration` are valid: `kICMValidTime_DisplayTimeStampIsValid` The time value passed in `displayTimeStamp` is valid. `kICMValidTime_DisplayDurationIsValid` The time value passed in `displayDuration` is valid. See these constants:

`kICMValidTime_DisplayTimeStampIsValid`

`kICMValidTime_DisplayDurationIsValid`

frameOptions

Options for this frame. Currently not used; pass NULL.

sourceTrackingCallback

A pointer to a callback to be notified about the status of this source frame. Pass NULL if you do not require notification.

sourceFrameRefCon

A reference constant to be passed to your callback. Use this parameter to point to a data structure containing any information your callback needs.

Return Value
Returns an error code, or 0 if there is no error. Encoded frames may or may not be output before the function returns.

Discussion
The session will retain the pixel buffer as long as necessary, and the client should not modify the pixel data until the session releases it. The most practical way to deal with this is by allocating pixel buffers from a pool. The client may fill in both, either, or neither of `displayTimeStamp` and `displayDuration`, but should set the appropriate flags to indicate which are valid. If the client needs to track the progress of a source frame, it should provide a source tracking callback. If multipass compression is enabled, calls to this function must be bracketed by calls to `ICMCompressionSessionBeginPass` and `ICMCompressionSessionEndPass`.

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

Related Sample Code
CaptureAndCompressIPBMovie
OpenGLCaptureToMovie
Quartz Composer QCTV

Declared In
`ImageCompression.h`

ICMCompressionSessionEndPass

Announces the end of a pass.

```
OSStatus ICMCompressionSessionEndPass (
   ICMCompressionSessionRef session
);
```

Parameters

session

A compression session reference. This reference is returned by `ICMCompressionSessionCreate` (page 31).

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

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

Declared In
`ImageCompression.h`

ICMCompressionSessionGetImageDescription

Retrieves the image description for a video compression session.

```
OSStatus ICMCompressionSessionGetImageDescription (
   ICMCompressionSessionRef session,
   ImageDescriptionHandle *imageDescOut
);
```

Parameters

session

A reference to a video compression session, returned by a previous call to `ICMCompressionSessionCreate` (page 31).

imageDescOut

A handle to an `ImageDescription` structure. The caller must not dispose of this handle; the ICM will dispose of it when the compression session is disposed.

Return Value

Returns an error code, or 0 if there is no error. For some codecs, this function may fail if called before the first frame is compressed.

Discussion

Multiple calls to this function return the same handle.

Availability

Available in Mac OS X v10.3 and later.

Declared In

`ImageCompression.h`

ICMCompressionSessionGetPixelBufferPool

Returns a pool that can provide ideal source pixel buffers for a compression session.

```
CVPixelBufferPoolRef ICMCompressionSessionGetPixelBufferPool (
   ICMCompressionSessionRef session
);
```

Parameters

session

A compression session reference. This reference is returned by `ICMCompressionSessionCreate` (page 31).

Return Value

A reference to a pool of pixel buffers. The compression session creates this pixel buffer pool based on the compressor's pixel buffer attributes and any pixel buffer attributes passed to `ICMCompressionSessionCreate` (page 31).

Discussion

A new compression session builds this pixel buffer pool based on the compressor's pixel buffer attributes and any pixel buffer attributes passed in to `ICMCompressionSessionCreate` (page 31). If the source pixel buffer attributes and the compressor pixel buffer attributes cannot be reconciled, the pool is based on the source pixel buffer attributes and the ICM converts each pixel buffer internally.

Availability

Available in Mac OS X v10.3 and later.

Declared In

`ImageCompression.h`

ICMCompressionSessionGetProperty

Retrieves the value of a specific property of a compression session.

```
OSStatus ICMCompressionSessionGetProperty (
   ICMCompressionSessionRef session,
   ComponentPropertyClass inPropClass,
   ComponentPropertyID inPropID,
   ByteCount inPropValueSize,
   ComponentValuePtr outPropValueAddress,
   ByteCount *outPropValueSizeUsed
);
```

Parameters

session

A compression session reference. This reference is returned by `ICMCompressionSessionCreate` (page 31).

inPropClass

Pass the following constant to define the property class: `kComponentPropertyClassPropertyInfo` = `'pnfo'` The property information class. See these constants:

`kComponentPropertyClassPropertyInfo`

inPropID

Pass one of these constants to define the property ID: `kComponentPropertyInfoList` = `'list'` An array of `CFData` values, one for each property. `kComponentPropertyCacheSeed` = `'seed'` A property cache seed value. `kComponentPropertyCacheFlags` = `'flgs'` One of the `kComponentPropertyCache` flags: `kComponentPropertyCacheFlagNotPersistentProperty` metadata should not be saved in persistent cache. `kComponentPropertyCacheFlagIsDynamicProperty` metadata should not cached at all. `kComponentPropertyExtendedInfo` = `'meta'` A `CFDictionary` with extended property information. See these constants:

`kComponentPropertyInfoList`

`kComponentPropertyCacheSeed`

`kComponentPropertyCacheFlags`

`kComponentPropertyExtendedInfo`

outPropType

A pointer to the type of the returned property's value.

outPropValueAddress

A pointer to a variable to receive the returned property's value.

outPropValueSizeUsed

On return, a pointer to the number of bytes actually used to store the property.

Return Value

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

Availability

Available in Mac OS X v10.3 and later.

Declared In

`ImageCompression.h`

ICMCompressionSessionGetPropertyInfo

Retrieves information about properties of a compression session.

```
OSStatus ICMCompressionSessionGetPropertyInfo (
   ICMCompressionSessionRef session,
   ComponentPropertyClass inPropClass,
   ComponentPropertyID inPropID,
   ComponentValueType *outPropType,
   ByteCount *outPropValueSize,
   UInt32 *outPropertyFlags
);
```

Parameters

session

A compression session reference. This reference is returned by `ICMCompressionSessionCreate` (page 31).

inPropClass

Pass the following constant to define the property class: `kComponentPropertyClassPropertyInfo` = `'pnfo'` The property information class. See these constants:

`kComponentPropertyClassPropertyInfo`

inPropID

Pass one of these constants to define the property ID: `kComponentPropertyInfoList` = `'list'` An array of `CFData` values, one for each property. `kComponentPropertyCacheSeed` = `'seed'` A property cache seed value. `kComponentPropertyCacheFlags` = `'flgs'` One of the `kComponentPropertyCache` flags: `kComponentPropertyCacheFlagNotPersistentProperty` metadata should not be saved in persistent cache. `kComponentPropertyCacheFlagIsDynamicProperty` metadata should not cached at all. `kComponentPropertyExtendedInfo` = `'meta'` A `CFDictionary` with extended property information. See these constants:

`kComponentPropertyInfoList`

`kComponentPropertyCacheSeed`

`kComponentPropertyCacheFlags`

`kComponentPropertyExtendedInfo`

outPropType

A pointer to the type of the returned property's value.

outPropValueSize

A pointer to the size of the returned property's value.

outPropFlags

On return, a pointer to flags representing the requested information about the property.

Return Value

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

Availability

Available in Mac OS X v10.3 and later.

Declared In

`ImageCompression.h`

ICMCompressionSessionGetTimeScale

Retrieves the time scale for a compression session.

```
TimeScale ICMCompressionSessionGetTimeScale (
   ICMCompressionSessionRef session
);
```

Parameters

session

A compression session reference. This reference is returned by `ICMCompressionSessionCreate` (page 31).

Return Value

The time scale for the compression session.

Availability

Available in Mac OS X v10.3 and later.

Related Sample Code

OpenGLCaptureToMovie

Quartz Composer QCTV

Declared In

`ImageCompression.h`

ICMCompressionSessionGetTypeID

Returns the type ID for the current compression session.

```
CFTypeID ICMCompressionSessionGetTypeID (
   void
);
```

Return Value

A `CFTypeID` value.

Availability

Available in Mac OS X v10.3 and later.

Declared In

`ImageCompression.h`

ICMCompressionSessionOptionsCreate

Creates a compression session options object.

```
OSStatus ICMCompressionSessionOptionsCreate (
   CFAllocatorRef allocator,
   ICMCompressionSessionOptionsRef *options
);
```

Parameters

allocator

An allocator. Pass NULL to use the default allocator.

options

On return, a reference to a new compression session options object.

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

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

Related Sample Code
CaptureAndCompressIPBMovie
Quartz Composer QCTV

Declared In
`ImageCompression.h`

ICMCompressionSessionOptionsCreateCopy

Copies a compression session options object.

```
OSStatus ICMCompressionSessionOptionsCreateCopy (
   CFAllocatorRef allocator,
   ICMCompressionSessionOptionsRef originalOptions,
   ICMCompressionSessionOptionsRef *copiedOptions
);
```

Parameters

allocator

An allocator. Pass NULL to use the default allocator.

originalOptions

A compression session options reference. This reference is returned by `ICMCompressionSessionOptionsCreate`.

copiedOptions

On return, a reference to a copy of the compression session options object passed in `originalOptions`.

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

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

Declared In
`ImageCompression.h`

ICMCompressionSessionOptionsGetAllowFrameReordering

Retrieves the allow frame reordering flag.

```
Boolean ICMCompressionSessionOptionsGetAllowFrameReordering (
   ICMCompressionSessionOptionsRef options
);
```

Parameters

options

A compression session options reference. This reference is returned by `ICMCompressionSessionOptionsCreate`.

Return Value
Returns TRUE if frame reordering is allowed, FALSE otherwise.

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

Related Sample Code
ExampleIPBCodec

Declared In
`ImageCompression.h`

ICMCompressionSessionOptionsGetAllowFrameTimeChanges

Retrieves the allow frame time changes flag.

```
Boolean ICMCompressionSessionOptionsGetAllowFrameTimeChanges (
   ICMCompressionSessionOptionsRef options
);
```

Parameters

options

A compression session options reference. This reference is returned by `ICMCompressionSessionOptionsCreate`.

Return Value
Returns TRUE if the compressor is allowed to modify frame times, FALSE otherwise.

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

Declared In
`ImageCompression.h`

ICMCompressionSessionOptionsGetAllowTemporalCompression

Retrieves the allow temporal compression flag.

```
Boolean ICMCompressionSessionOptionsGetAllowTemporalCompression (
   ICMCompressionSessionOptionsRef options
);
```

Parameters

options

A compression session options reference. This reference is returned by `ICMCompressionSessionOptionsCreate`.

Return Value
Returns TRUE if temporal compression is allowed, FALSE otherwise.

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

Related Sample Code
ExampleIPBCodec

Declared In
`ImageCompression.h`

ICMCompressionSessionOptionsGetDurationsNeeded

Retrieves the durations needed flag.

```
Boolean ICMCompressionSessionOptionsGetDurationsNeeded (
   ICMCompressionSessionOptionsRef options
);
```

Parameters

options

A compression session options reference. This reference is returned by `ICMCompressionSessionOptionsCreate`.

Return Value
Returns TRUE if the durations of outputted frames must be calculated, FALSE otherwise.

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

Declared In
`ImageCompression.h`

ICMCompressionSessionOptionsGetMaxKeyFrameInterval

Retrieves the maximum key frame interval.

```
SInt32 ICMCompressionSessionOptionsGetMaxKeyFrameInterval (
   ICMCompressionSessionOptionsRef options
);
```

Parameters

options

A compression session options reference. This reference is returned by `ICMCompressionSessionOptionsCreate`.

Return Value

Returns the maximum key frame interval.

Availability

Available in Mac OS X v10.3 and later.

Related Sample Code

ExampleIPBCodec

Declared In

`ImageCompression.h`

ICMCompressionSessionOptionsGetProperty

Retrieves the value of a specific property of a compression session options object.

```
OSStatus ICMCompressionSessionOptionsGetProperty (
   ICMCompressionSessionOptionsRef options,
   ComponentPropertyClass inPropClass,
   ComponentPropertyID inPropID,
   ByteCount inPropValueSize,
   ComponentValuePtr outPropValueAddress,
   ByteCount *outPropValueSizeUsed
);
```

Parameters

options

A compression session options reference. This reference is returned by `ICMCompressionSessionOptionsCreate`.

inPropClass

Pass the following constant to define the property class: `kComponentPropertyClassPropertyInfo = 'pnfo'` The property information class. See these constants:

`kComponentPropertyClassPropertyInfo`

inPropID

Pass one of these constants to define the property ID: `kComponentPropertyInfoList` = `'list'` An array of `CFData` values, one for each property. `kComponentPropertyCacheSeed` = `'seed'` A property cache seed value. `kComponentPropertyCacheFlags` = `'flgs'` One of the `kComponentPropertyCache` flags: `kComponentPropertyCacheFlagNotPersistentProperty` metadata should not be saved in persistent cache. `kComponentPropertyCacheFlagIsDynamicProperty` metadata should not cached at all. `kComponentPropertyExtendedInfo` = `'meta'` A `CFDictionary` with extended property information. See these constants:

```
kComponentPropertyInfoList
kComponentPropertyCacheSeed
kComponentPropertyCacheFlags
kComponentPropertyExtendedInfo
```

outPropType

A pointer to the type of the returned property's value.

outPropValueAddress

A pointer to a variable to receive the returned property's value.

outPropValueSizeUsed

On return, a pointer to the number of bytes actually used to store the property.

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

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

Related Sample Code
ExampleIPBCodec

Declared In
`ImageCompression.h`

ICMCompressionSessionOptionsGetPropertyInfo

Retrieves information about properties of a compression session options object.

```
OSStatus ICMCompressionSessionOptionsGetPropertyInfo (
   ICMCompressionSessionOptionsRef options,
   ComponentPropertyClass inPropClass,
   ComponentPropertyID inPropID,
   ComponentValueType *outPropType,
   ByteCount *outPropValueSize,
   UInt32 *outPropertyFlags
);
```

Parameters

options

A compression session options reference. This reference is returned by `ICMCompressionSessionOptionsCreate`.

inPropClass

Pass the following constant to define the property class: kComponentPropertyClassPropertyInfo = 'pnfo' The property information class. See these constants:

kComponentPropertyClassPropertyInfo

inPropID

Pass one of these constants to define the property ID: kComponentPropertyInfoList = 'list' An array of CFData values, one for each property. kComponentPropertyCacheSeed = 'seed' A property cache seed value. kComponentPropertyCacheFlags = 'flgs' One of the kComponentPropertyCache flags: kComponentPropertyCacheFlagNotPersistentProperty metadata should not be saved in persistent cache. kComponentPropertyCacheFlagIsDynamicProperty metadata should not cached at all. kComponentPropertyExtendedInfo = 'meta' A CFDictionary with extended property information. See these constants:

kComponentPropertyInfoList

kComponentPropertyCacheSeed

kComponentPropertyCacheFlags

kComponentPropertyExtendedInfo

outPropType

A pointer to the type of the returned property's value.

outPropValueSize

A pointer to the size of the returned property's value.

outPropFlags

On return, a pointer to flags representing the requested information about the property.

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

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

Declared In
ImageCompression.h

ICMCompressionSessionOptionsGetTypeID

Returns the type ID for the current compression session options object.

```
CFTypeID ICMCompressionSessionOptionsGetTypeID (
   void
);
```

Return Value
A CFTypeID value.

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

Declared In
ImageCompression.h

ICMCompressionSessionOptionsRelease

Decrements the retain count of a compression session options object.

```
void ICMCompressionSessionOptionsRelease (
   ICMCompressionSessionOptionsRef options
);
```

Parameters

options

A reference to a compression session options object. This reference is returned by `ICMCompressionSessionOptionsCreate`. If you pass NULL, nothing happens.

Discussion
If the retain count drops to 0, the object is disposed.

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

Related Sample Code
CaptureAndCompressIPBMovie
ExampleIPBCodec

Declared In
`ImageCompression.h`

ICMCompressionSessionOptionsRetain

Increments the retain count of a compression session options object.

```
ICMCompressionSessionOptionsRef ICMCompressionSessionOptionsRetain (
   ICMCompressionSessionOptionsRef options
);
```

Parameters

options

A reference to a compression session options object. This reference is returned by `ICMCompressionSessionOptionsCreate`. If you pass NULL, nothing happens.

Return Value
A copy of the object reference passed in options, for convenience.

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

Related Sample Code
ExampleIPBCodec

Declared In
`ImageCompression.h`

ICMCompressionSessionOptionsSetAllowFrameReordering

Enables frame reordering.

```
OSStatus ICMCompressionSessionOptionsSetAllowFrameReordering (
   ICMCompressionSessionOptionsRef options,
   Boolean allowFrameReordering
);
```

Parameters

options

A compression session options reference. This reference is returned by `ICMCompressionSessionOptionsCreate`.

allowFrameReordering

Pass TRUE to enable frame reordering, FALSE to disable it.

Return Value

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

Discussion

To encode B-frames a compressor must reorder frames, which means that the order in which they will be emitted and stored (the decode order) is different from the order in which they were presented to the compressor (the display order). By default, frame reordering is disabled. To encode using B-frames, you must call this function, passing TRUE.

Availability

Available in Mac OS X v10.3 and later.

Related Sample Code

CaptureAndCompressIPBMovie

OpenGLCaptureToMovie

Declared In

`ImageCompression.h`

ICMCompressionSessionOptionsSetAllowFrameTimeChanges

Allows the compressor to modify frame times.

```
OSStatus ICMCompressionSessionOptionsSetAllowFrameTimeChanges (
   ICMCompressionSessionOptionsRef options,
   Boolean allowFrameTimeChanges
);
```

Parameters

options

A compression session options reference. This reference is returned by `ICMCompressionSessionOptionsCreate`.

allowFrameTimeChanges

Pass TRUE to let the compressor to modify frame times, FALSE to prohibit it.

Return Value

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

Discussion
Some compressors are able to identify and coalesce runs of identical frames and output single frames with longer durations, or output frames at a different frame rate from the original. This feature is controlled by the allow frame time changes flag. By default, this flag is set to false, which forces compressors to emit one encoded frame for every source frame and preserve frame display times.

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

Related Sample Code
CaptureAndCompressIPBMovie
OpenGLCaptureToMovie

Declared In
`ImageCompression.h`

ICMCompressionSessionOptionsSetAllowTemporalCompression

Enables temporal compression.

```
OSStatus ICMCompressionSessionOptionsSetAllowTemporalCompression (
   ICMCompressionSessionOptionsRef options,
   Boolean allowTemporalCompression
);
```

Parameters

options

A compression session options reference. This reference is returned by `ICMCompressionSessionOptionsCreate`.

allowTemporalCompression

Pass TRUE to enable temporal compression, FALSE to disable it.

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

Discussion
By default, temporal compression is disabled. If you want temporal compression for P-frames or B-frames you must call this function and pass TRUE.

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

Related Sample Code
CaptureAndCompressIPBMovie
OpenGLCaptureToMovie

Declared In
`ImageCompression.h`

ICMCompressionSessionOptionsSetDurationsNeeded

Indicates that the durations of outputted frames must be calculated.

```
OSStatus ICMCompressionSessionOptionsSetDurationsNeeded (
   ICMCompressionSessionOptionsRef options,
   Boolean decodeDurationsNeeded
);
```

Parameters

options

A compression session options reference. This reference is returned by `ICMCompressionSessionOptionsCreate`.

decodeDurationsNeeded

Pass TRUE to indicate that durations must be calculated, FALSE otherwise.

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

Discussion
If this flag is set and source frames are provided with times but not durations, then frames will be delayed so that durations can be calculated as the difference between one frame's time stamp and the next frame's time stamp. By default this flag is 0, so frames will not be delayed in order to calculate durations.

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

Related Sample Code
CaptureAndCompressIPBMovie
OpenGLCaptureToMovie

Declared In
`ImageCompression.h`

ICMCompressionSessionOptionsSetMaxKeyFrameInterval

Sets the maximum interval between key frames.

```
OSStatus ICMCompressionSessionOptionsSetMaxKeyFrameInterval (
   ICMCompressionSessionOptionsRef options,
   SInt32 maxKeyFrameInterval
);
```

Parameters

options

A compression session options reference. This reference is returned by `ICMCompressionSessionOptionsCreate`.

maxKeyFrameInterval

The maximum interval between key frames, also known as the key frame rate.

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

Discussion
Compressors are allowed to generate key frames more frequently if this would result in more efficient compression. The default key frame interval is 0, which indicates that the compressor should choose where to place all key frames.

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

Related Sample Code
CaptureAndCompressIPBMovie
OpenGLCaptureToMovie

Declared In
`ImageCompression.h`

ICMCompressionSessionOptionsSetProperty

Sets the value of a specific property of a compression session options object.

```
OSStatus ICMCompressionSessionOptionsSetProperty (
   ICMCompressionSessionOptionsRef options,
   ComponentPropertyClass inPropClass,
   ComponentPropertyID inPropID,
   ByteCount inPropValueSize,
   ConstComponentValuePtr inPropValueAddress
);
```

Parameters

`options`

A compression session options reference. This reference is returned by `ICMCompressionSessionOptionsCreate`.

`inPropClass`

Pass the following constant to define the property class: `kComponentPropertyClassPropertyInfo` = `'pnfo'` The property information class. See these constants:

`kComponentPropertyClassPropertyInfo`

`inPropID`

Pass one of these constants to define the property ID: `kComponentPropertyInfoList` = `'list'` An array of `CFData` values, one for each property. `kComponentPropertyCacheSeed` = `'seed'` A property cache seed value. `kComponentPropertyCacheFlags` = `'flgs'` One of the `kComponentPropertyCache` flags: `kComponentPropertyCacheFlagNotPersistentProperty` metadata should not be saved in persistent cache. `kComponentPropertyCacheFlagIsDynamicProperty` metadata should not cached at all. `kComponentPropertyExtendedInfo` = `'meta'` A `CFDictionary` with extended property information. See these constants:

`kComponentPropertyInfoList`
`kComponentPropertyCacheSeed`
`kComponentPropertyCacheFlags`
`kComponentPropertyExtendedInfo`

`inPropValueSize`

The size of the property value to be set.

`inPropValueAddress`

A pointer to the value of the property to be set.

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

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

Related Sample Code
CaptureAndCompressIPBMovie
OpenGLCaptureToMovie
Quartz Composer QCTV

Declared In
`ImageCompression.h`

ICMCompressionSessionProcessBetweenPasses

Lets the compressor perform processing between passes.

```
OSStatus ICMCompressionSessionProcessBetweenPasses (
   ICMCompressionSessionRef session,
   UInt32 flags,
   Boolean *interpassProcessingDoneOut,
   ICMCompressionPassModeFlags *requestedNextPassModeFlagsOut
);
```

Parameters

session

A compression session reference. This reference is returned by `ICMCompressionSessionCreate` (page 31).

flags

Reserved. Set to 0.

interpassProcessingDoneOut

A pointer to a Boolean that will be set to FALSE if this function should be called again, TRUE if not.

requestedNextPassModeFlagsOut

A pointer to `ICMCompressionPassModeFlags` that will be set to the codec's recommended mode flags for the next pass. `kICMCompressionPassMode_OutputEncodedFrames` will be set only if it recommends that the next pass be the final one:
`kICMCompressionPassMode_OutputEncodedFrames` = 1L<<0 Output encoded frames.
`kICMCompressionPassMode_NoSourceFrames` = 1L<<1 The client need not provide source frame buffers. `kICMCompressionPassMode_WriteToMultiPassStorage` = 1L<<2 The compressor may write private data to multipass storage. `kICMCompressionPassMode_ReadFromMultiPassStorage` = 1L<<3 The compressor may read private data from multipass storage. See these constants:

`kICMCompressionPassMode_OutputEncodedFrames`
`kICMCompressionPassMode_NoSourceFrames`
`kICMCompressionPassMode_WriteToMultiPassStorage`
`kICMCompressionPassMode_ReadFromMultiPassStorage`

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

Discussion
Call this function repeatedly until the compressor sets `interpassProcessingDoneOut` to TRUE to indicate that it is done with this round of interpass processing. When done, the compressor will indicate its preferred mode for the next pass. At this point the client may choose to begin an encoding pass, by OR-combining the `kICMCompressionPassMode_OutputEncodedFrames` flag, regardless of the compressor's request.

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

Declared In
`ImageCompression.h`

ICMCompressionSessionRelease

Decrements the retain count of a compression session.

```
void ICMCompressionSessionRelease (
   ICMCompressionSessionRef session
);
```

Parameters

session

A compression session reference. This reference is returned by `ICMCompressionSessionCreate` (page 31). If you pass NULL, nothing happens.

Discussion
If the retain count drops to 0, the session is disposed.

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

Related Sample Code
CaptureAndCompressIPBMovie
OpenGLCaptureToMovie
Quartz Composer QCTV

Declared In
`ImageCompression.h`

ICMCompressionSessionRetain

Increments the retain count of a compression session.

```
ICMCompressionSessionRef ICMCompressionSessionRetain (
   ICMCompressionSessionRef session
);
```

Parameters

session

A compression session reference. This reference is returned by `ICMCompressionSessionCreate` (page 31). If you pass NULL, nothing happens.

Return Value
A reference to the object passed in session, for convenience.

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

Declared In
`ImageCompression.h`

ICMCompressionSessionSetProperty

Sets the value of a specific property of a compression session.

```
OSStatus ICMCompressionSessionSetProperty (
   ICMCompressionSessionRef session,
   ComponentPropertyClass inPropClass,
   ComponentPropertyID inPropID,
   ByteCount inPropValueSize,
   ConstComponentValuePtr inPropValueAddress
);
```

Parameters

session

A compression session reference. This reference is returned by `ICMCompressionSessionCreate` (page 31).

inPropClass

Pass the following constant to define the property class: `kComponentPropertyClassPropertyInfo` = `'pnfo'` The property information class. See these constants:

`kComponentPropertyClassPropertyInfo`

inPropID

Pass one of these constants to define the property ID: `kComponentPropertyInfoList` = `'list'` An array of `CFData` values, one for each property. `kComponentPropertyCacheSeed` = `'seed'` A property cache seed value. `kComponentPropertyCacheFlags` = `'flgs'` One of the `kComponentPropertyCache` flags: `kComponentPropertyCacheFlagNotPersistentProperty` metadata should not be saved in persistent cache. `kComponentPropertyCacheFlagIsDynamicProperty` metadata should not cached at all. `kComponentPropertyExtendedInfo` = `'meta'` A `CFDictionary` with extended property information. See these constants:

`kComponentPropertyInfoList`
`kComponentPropertyCacheSeed`
`kComponentPropertyCacheFlags`
`kComponentPropertyExtendedInfo`

inPropValueSize

The size of the property value to be set.

inPropValueAddress

A pointer to the value of the property to be set.

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

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

Declared In
`ImageCompression.h`

ICMCompressionSessionSupportsMultiPassEncoding

Queries whether a compression session supports multipass encoding.

```
Boolean ICMCompressionSessionSupportsMultiPassEncoding (
   ICMCompressionSessionRef session,
   UInt32 multiPassStyleFlags,
   ICMCompressionPassModeFlags *firstPassModeFlagsOut
);
```

Parameters

session
: A compression session reference. This reference is returned by `ICMCompressionSessionCreate` (page 31).

multiPassStyleFlags
: Reserved; set to 0.

firstPassModeFlagsOut
: A pointer to a variable to receive the session's requested mode flags for the first pass. The client may modify these flags, but should not set `kICMCompressionPassMode_NoSourceFrames`. Pass NULL if you do not want this information.

Return Value
Returns TRUE if the compression session supports multipass encoding, FALSE otherwise.

Discussion
Even if this function returns FALSE, if you passed TRUE to `ICMCompressionSessionOptionsSetMultiPass`, you must call `ICMCompressionSessionBeginPass` and `ICMCompressionSessionEndPass`.

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

Declared In
`ImageCompression.h`

ICMCompressorSessionDropFrame

Called by a compressor to notify the ICM that a source frame has been dropped and will not contribute to any encoded frames.

```
OSStatus ICMCompressorSessionDropFrame (
   ICMCompressorSessionRef session,
   ICMCompressorSourceFrameRef sourceFrame
);
```

Parameters

session
: A reference to the compression session between the ICM and an image compressor component.

sourceFrame

A reference to a frame that has been passed in `sourceFrameRefCon` to `ICMCompressionSessionEncodeFrame` (page 33). If you pass NULL, nothing happens.

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

Discussion
Calling this function does not automatically release the source frame; if the compressor called `ICMCompressorSourceFrameRetain` it should still call `ICMCompressorSourceFrameRelease`.

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

Declared In
`ImageCompression.h`

ICMCompressorSessionEmitEncodedFrame

Called by a compressor to output an encoded frame corresponding to one or more source frames.

```
OSStatus ICMCompressorSessionEmitEncodedFrame (
   ICMCompressorSessionRef session,
   ICMMutableEncodedFrameRef encodedFrame,
   long numberOfSourceFrames,
   ICMCompressorSourceFrameRef sourceFrames[]
);
```

Parameters

session

A reference to the compression session between the ICM and an image compressor component.

encodedFrame

A reference to an encoded frame object with write capabilities.

numberOfSourceFrames

The number of source frames encoded in the encoded frame.

sourceFrames

References to frames that have been passed in `sourceFrameRefCon` to `ICMCompressionSessionEncodeFrame` (page 33).

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

Discussion
Encoded frames may correspond to more than one source frame only if `allowFrameTimeChanges` is set in the compression session's `compressionSessionOptions`.

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

Related Sample Code
ExampleIPBCodec

Declared In
`ImageCompression.h`

ICMCompressorSourceFrameGetDisplayNumber

Retrieves a source frames display number.

```
long ICMCompressorSourceFrameGetDisplayNumber (
   ICMCompressorSourceFrameRef sourceFrame
);
```

Parameters

sourceFrame

A reference to a frame that has been passed in `sourceFrameRefCon` to `ICMCompressionSessionEncodeFrame` (page 33).

Return Value

The display number of the source frame.

Discussion

The ICM tags source frames with display numbers in the order that they are passed to `ICMCompressionSessionEncodeFrame` (page 33). The first display number is 1. Compressors may compare these numbers to work out whether `prediction` is forward or backward, even when display times are not provided.

Availability

Available in Mac OS X v10.3 and later.

Related Sample Code

ExampleIPBCodec

Declared In

`ImageCompression.h`

ICMCompressorSourceFrameGetDisplayTimeStampAndDuration

Retrieves the display time stamp and duration of a source frame.

```
OSStatus ICMCompressorSourceFrameGetDisplayTimeStampAndDuration (
   ICMCompressorSourceFrameRef sourceFrame,
   TimeValue64 *displayTimeStampOut,
   TimeValue64 *displayDurationOut,
   TimeScale *timeScaleOut,
   ICMValidTimeFlags *validTimeFlagsOut
);
```

Parameters

sourceFrame

A reference to a frame that has been passed in `sourceFrameRefCon` to `ICMCompressionSessionEncodeFrame` (page 33).

displayTimeStampOut

A pointer to the source frame's display time stamp.

displayDurationOut

A pointer to the source frame's display duration.

timeScaleOut

A pointer to the source frame's display time scale.

validTimeFlagsOut

A pointer to one of these display time flags for the source frame:
kICMValidTime_DisplayTimeStampIsValid = 1L<<0 The value of displayTimeStamp is valid.
kICMValidTime_DisplayDurationIsValid = 1L<<1 The value of displayDuration is valid.
See these constants:

kICMValidTime_DisplayTimeStampIsValid

kICMValidTime_DisplayDurationIsValid

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

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

Related Sample Code
ExampleIPBCodec

Declared In
ImageCompression.h

ICMCompressorSourceFrameGetFrameOptions

Retrieves the frame compression options for a source frame.

```
ICMCompressionFrameOptionsRef ICMCompressorSourceFrameGetFrameOptions (
   ICMCompressorSourceFrameRef sourceFrame
);
```

Parameters

sourceFrame

A reference to a frame that has been passed in sourceFrameRefCon to ICMCompressionSessionEncodeFrame (page 33).

Return Value
A compression session frame options reference representing options for this frame. A frame options object is created by ICMCompressionFrameOptionsCreate.

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

Related Sample Code
ExampleIPBCodec

Declared In
ImageCompression.h

ICMCompressorSourceFrameGetPixelBuffer

Retrieves a source frames pixel buffer.

```
CVPixelBufferRef ICMCompressorSourceFrameGetPixelBuffer (
   ICMCompressorSourceFrameRef sourceFrame
);
```

Parameters

sourceFrame

A reference to a frame that has been passed in `sourceFrameRefCon` to `ICMCompressionSessionEncodeFrame` (page 33).

Return Value
A reference to the pixel buffer containing the source frame's image being compressed.

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

Related Sample Code
ExampleIPBCodec

Declared In
`ImageCompression.h`

ICMCompressorSourceFrameGetTypeID

Returns the type ID for the current source frame object.

```
CFTypeID ICMCompressorSourceFrameGetTypeID (
   void
);
```

Return Value
A `CFTypeID` value.

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

Declared In
`ImageCompression.h`

ICMCompressorSourceFrameRelease

Decrements the retain count of a source frame object.

```
void ICMCompressorSourceFrameRelease (
   ICMCompressorSourceFrameRef sourceFrame
);
```

Parameters

sourceFrame

A reference to a frame that has been passed in `sourceFrameRefCon` to `ICMCompressionSessionEncodeFrame` (page 33). If you pass NULL, nothing happens.

Discussion
If the retain count drops to 0, the object is disposed.

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

Related Sample Code
ExampleIPBCodec

Declared In
`ImageCompression.h`

ICMCompressorSourceFrameRetain

Increments the retain count of a source frame object.

```
ICMCompressorSourceFrameRef ICMCompressorSourceFrameRetain (
   ICMCompressorSourceFrameRef sourceFrame
);
```

Parameters

sourceFrame

A reference to a frame that has been passed in `sourceFrameRefCon` to `ICMCompressionSessionEncodeFrame` (page 33). If you pass NULL, nothing happens.

Return Value
A reference to the object passed in `sourceFrame`, for convenience.

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

Related Sample Code
ExampleIPBCodec

Declared In
`ImageCompression.h`

ICMDecompressionFrameOptionsCreate

Creates a frame decompression options object.

```
OSStatus ICMDecompressionFrameOptionsCreate (
   CFAllocatorRef allocator,
   ICMDecompressionFrameOptionsRef *options
);
```

Parameters

allocator

An allocator. Pass NULL to use the default allocator.

options

On return, a reference to a frame decompression options object.

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

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

Declared In
ImageCompression.h

ICMDecompressionFrameOptionsCreateCopy

Copies a frame decompression options object.

```
OSStatus ICMDecompressionFrameOptionsCreateCopy (
   CFAllocatorRef allocator,
   ICMDecompressionFrameOptionsRef originalOptions,
   ICMDecompressionFrameOptionsRef *copiedOptions
);
```

Parameters

allocator
An allocator. Pass NULL to use the default allocator.

originalOptions
A reference to a frame decompression options object. You can create this object by calling ICMDecompressionFrameOptionsCreate.

copiedOptions
On return, a reference to a copy of the frame decompression options object passed in originalOptions.

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

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

Declared In
ImageCompression.h

ICMDecompressionFrameOptionsGetProperty

Retrieves the value of a specific property of a decompression frame options object.

```
OSStatus ICMDecompressionFrameOptionsGetProperty (
   ICMDecompressionFrameOptionsRef options,
   ComponentPropertyClass inPropClass,
   ComponentPropertyID inPropID,
   ByteCount inPropValueSize,
   ComponentValuePtr outPropValueAddress,
   ByteCount *outPropValueSizeUsed
);
```

Parameters

options
A decompression frame options reference. This reference is returned by ICMDecompressionFrameOptionsCreate.

inPropClass

Pass the following constant to define the property class: `kComponentPropertyClassPropertyInfo` = `'pnfo'` The property information class. See these constants:

`kComponentPropertyClassPropertyInfo`

inPropID

Pass one of these constants to define the property ID: `kComponentPropertyInfoList` = `'list'` An array of `CFData` values, one for each property. `kComponentPropertyCacheSeed` = `'seed'` A property cache seed value. `kComponentPropertyCacheFlags` = `'flgs'` One of the `kComponentPropertyCache` flags: `kComponentPropertyCacheFlagNotPersistentProperty` metadata should not be saved in persistent cache. `kComponentPropertyCacheFlagIsDynamicProperty` metadata should not cached at all. `kComponentPropertyExtendedInfo` = `'meta'` A `CFDictionary` with extended property information. See these constants:

`kComponentPropertyInfoList`

`kComponentPropertyCacheSeed`

`kComponentPropertyCacheFlags`

`kComponentPropertyExtendedInfo`

outPropType

A pointer to the type of the returned property's value.

outPropValueAddress

A pointer to a variable to receive the returned property's value.

outPropValueSizeUsed

On return, a pointer to the number of bytes actually used to store the property.

Return Value

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

Availability

Available in Mac OS X v10.3 and later.

Declared In

`ImageCompression.h`

ICMDecompressionFrameOptionsGetPropertyInfo

Retrieves information about properties of a decompression frame options object.

```
OSStatus ICMDecompressionFrameOptionsGetPropertyInfo (
   ICMDecompressionFrameOptionsRef options,
   ComponentPropertyClass inPropClass,
   ComponentPropertyID inPropID,
   ComponentValueType *outPropType,
   ByteCount *outPropValueSize,
   UInt32 *outPropertyFlags
);
```

Parameters

options

A decompression frame options reference. This reference is returned by `ICMDecompressionFrameOptionsCreate`.

inPropClass

Pass the following constant to define the property class: `kComponentPropertyClassPropertyInfo` = `'pnfo'` The property information class. See these constants:

`kComponentPropertyClassPropertyInfo`

inPropID

Pass one of these constants to define the property ID: `kComponentPropertyInfoList` = `'list'` An array of `CFData` values, one for each property. `kComponentPropertyCacheSeed` = `'seed'` A property cache seed value. `kComponentPropertyCacheFlags` = `'flgs'` One of the `kComponentPropertyCache` flags: `kComponentPropertyCacheFlagNotPersistentProperty` metadata should not be saved in persistent cache. `kComponentPropertyCacheFlagIsDynamicProperty` metadata should not cached at all. `kComponentPropertyExtendedInfo` = `'meta'` A `CFDictionary` with extended property information. See these constants:

`kComponentPropertyInfoList`

`kComponentPropertyCacheSeed`

`kComponentPropertyCacheFlags`

`kComponentPropertyExtendedInfo`

outPropType

A pointer to the type of the returned property's value.

outPropValueSize

A pointer to the size of the returned property's value.

outPropFlags

On return, a pointer to flags representing the requested information about the frame option's property.

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

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

Declared In
`ImageCompression.h`

ICMDecompressionFrameOptionsGetTypeID

Returns the type ID for the current frame decompression options object.

```
CFTypeID ICMDecompressionFrameOptionsGetTypeID (
   void
);
```

Return Value
A `CFTypeID` value.

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

Declared In
`ImageCompression.h`

ICMDecompressionFrameOptionsRelease

Decrements the retain count of a frame decompression options object.

```
void ICMDecompressionFrameOptionsRelease (
   ICMDecompressionFrameOptionsRef options
);
```

Parameters

options

A reference to a frame decompression options object. You can create this object by calling `ICMDecompressionFrameOptionsCreate`. If you pass NULL, nothing happens.

Discussion

If the retain count drops to 0, the object is disposed.

Availability

Available in Mac OS X v10.3 and later.

Declared In

`ImageCompression.h`

ICMDecompressionFrameOptionsRetain

Increments the retain count of a frame decompression options object.

```
ICMDecompressionFrameOptionsRef ICMDecompressionFrameOptionsRetain (
   ICMDecompressionFrameOptionsRef options
);
```

Parameters

options

A reference to a frame decompression options object. You can create this object by calling `ICMDecompressionFrameOptionsCreate`. If you pass NULL, nothing happens.

Return Value

A reference to the frame decompression options object passed in options, for convenience.

Availability

Available in Mac OS X v10.3 and later.

Declared In

`ImageCompression.h`

ICMDecompressionFrameOptionsSetProperty

Sets the value of a specific property of a decompression frame options object.

```
OSStatus ICMDecompressionFrameOptionsSetProperty (
   ICMDecompressionFrameOptionsRef options,
   ComponentPropertyClass inPropClass,
   ComponentPropertyID inPropID,
   ByteCount inPropValueSize,
   ConstComponentValuePtr inPropValueAddress
);
```

Parameters

options

A decompression frame options reference. This reference is returned by `ICMDecompressionFrameOptionsCreate`.

inPropClass

Pass the following constant to define the property class: `kComponentPropertyClassPropertyInfo` = `'pnfo'` The property information class. See these constants:

`kComponentPropertyClassPropertyInfo`

inPropID

Pass one of these constants to define the property ID: `kComponentPropertyInfoList` = `'list'` An array of `CFData` values, one for each property. `kComponentPropertyCacheSeed` = `'seed'` A property cache seed value. `kComponentPropertyCacheFlags` = `'flgs'` One of the `kComponentPropertyCache` flags: `kComponentPropertyCacheFlagNotPersistentProperty` metadata should not be saved in persistent cache. `kComponentPropertyCacheFlagIsDynamicProperty` metadata should not cached at all. `kComponentPropertyExtendedInfo` = `'meta'` A `CFDictionary` with extended property information. See these constants:

`kComponentPropertyInfoList`

`kComponentPropertyCacheSeed`

`kComponentPropertyCacheFlags`

`kComponentPropertyExtendedInfo`

inPropValueSize

The size of the property value to be set.

inPropValueAddress

A pointer to the value of the property to be set.

Return Value

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

Availability

Available in Mac OS X v10.3 and later.

Declared In

`ImageCompression.h`

ICMDecompressionSessionCreate

Creates a session for decompressing video frames.

```
OSStatus ICMDecompressionSessionCreate (
   CFAllocatorRef allocator,
   ImageDescriptionHandle desc,
   ICMDecompressionSessionOptionsRef decompressionOptions,
   CFDictionaryRef destinationPixelBufferAttributes,
   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

A decompression session options reference. This reference is returned by `ICMDecompressionSessionOptionsCreate`. The session will retain the object. You may change some options during the session by modifying the object. You may also pass NULL.

destinationPixelBufferAttributes

Requirements for emitted pixel buffers. You may pass NULL.

trackingCallback

A pointer to a structure that designates a callback to be called for information about queued frames and pixel buffers containing decompressed frames. See `ICMDecompressionTrackingCallbackRecord` and `ICMDecompressionTrackingCallbackProc`.

decompressionSessionOut

A pointer to a variable to receive a reference to the new decompression session.

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

Discussion
Frames are returned through calls to the callback pointed to by `trackingCallback`.

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

Related Sample Code
CaptureAndCompressIPBMovie
MovieVideoChart
WhackedTV

Declared In
`ImageCompression.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.

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

Related Sample Code
QTQuartzPlayer

Declared In
`ImageCompression.h`

ICMDecompressionSessionDecodeFrame

Queues a frame for decompression.

```
OSStatus ICMDecompressionSessionDecodeFrame (
   ICMDecompressionSessionRef session,
   const UInt8 *data,
   ByteCount dataSize,
   ICMDecompressionFrameOptionsRef frameOptions,
   const ICMFrameTimeRecord *frameTime,
   void *sourceFrameRefCon
);
```

Parameters

session

A decompression session reference. This reference is returned by `ICMDecompressionSessionCreate`.

data

A pointer to the compressed data for this frame. The data must remain in this location until `ICMDecompressionTrackingCallbackProc` is called with the `kICMDecompressionTracking_ReleaseSourceData` flag set in `decompressionTrackingFlags`.

dataSize

The number of bytes of compressed data. You may not pass 0 in this parameter.

frameOptions

A reference to a frame decompression options object containing options for this frame. You can create this object by calling `ICMDecompressionFrameOptionsCreate`.

frameTime

A pointer to a structure describing the frame's timing information.

sourceFrameRefCon

Your reference value for the frame.

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

Related Sample Code
CaptureAndCompressIPBMovie
MovieVideoChart
QTQuartzPlayer
WhackedTV

Declared In
`ImageCompression.h`

ICMDecompressionSessionFlush

Flushes the frames queued for a decompression session.

```
OSStatus ICMDecompressionSessionFlush (
   ICMDecompressionSessionRef session
);
```

Parameters

session

A decompression session reference. This reference is returned by `ICMDecompressionSessionCreate`.

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

Discussion
The tracking callback will be called for each frame with the result -1.

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

Declared In
`ImageCompression.h`

ICMDecompressionSessionGetProperty

Retrieves the value of a specific property of a decompression session.

```
OSStatus ICMDecompressionSessionGetProperty (
   ICMDecompressionSessionRef session,
   ComponentPropertyClass inPropClass,
   ComponentPropertyID inPropID,
   ByteCount inPropValueSize,
   ComponentValuePtr outPropValueAddress,
   ByteCount *outPropValueSizeUsed
);
```

Parameters

session

A decompression session reference. This reference is returned by `ICMDecompressionSessionCreate`.

inPropClass

Pass the following constant to define the property class: `kComponentPropertyClassPropertyInfo` = `'pnfo'` The property information class. See these constants:

`kComponentPropertyClassPropertyInfo`

inPropID

Pass one of these constants to define the property ID: `kComponentPropertyInfoList` = `'list'` An array of `CFData` values, one for each property. `kComponentPropertyCacheSeed` = `'seed'` A property cache seed value. `kComponentPropertyCacheFlags` = `'flgs'` One of the `kComponentPropertyCache` flags: `kComponentPropertyCacheFlagNotPersistentProperty` metadata should not be saved in persistent cache.
`kComponentPropertyCacheFlagIsDynamicProperty` metadata should not cached at all.
`kComponentPropertyExtendedInfo` = `'meta'` A `CFDictionary` with extended property information. See these constants:

`kComponentPropertyInfoList`
`kComponentPropertyCacheSeed`
`kComponentPropertyCacheFlags`
`kComponentPropertyExtendedInfo`

outPropType

A pointer to the type of the returned property's value.

outPropValueAddress

A pointer to a variable to receive the returned property's value.

outPropValueSizeUsed

On return, a pointer to the number of bytes actually used to store the property.

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

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

Declared In
`ImageCompression.h`

ICMDecompressionSessionGetPropertyInfo

Retrieves information about the properties of a decompression session.

```
OSStatus ICMDecompressionSessionGetPropertyInfo (
   ICMDecompressionSessionRef session,
   ComponentPropertyClass inPropClass,
   ComponentPropertyID inPropID,
   ComponentValueType *outPropType,
   ByteCount *outPropValueSize,
   UInt32 *outPropertyFlags
);
```

Parameters

session
A decompression session reference. This reference is returned by `ICMDecompressionSessionCreate`.

inPropClass
Pass the following constant to define the property class: `kComponentPropertyClassPropertyInfo` = `'pnfo'` The property information class. See these constants:
`kComponentPropertyClassPropertyInfo`

inPropID
Pass one of these constants to define the property ID: `kComponentPropertyInfoList` = `'list'` An array of `CFData` values, one for each property. `kComponentPropertyCacheSeed` = `'seed'` A property cache seed value. `kComponentPropertyCacheFlags` = `'flgs'` One of the `kComponentPropertyCache` flags: `kComponentPropertyCacheFlagNotPersistentProperty` metadata should not be saved in persistent cache.
`kComponentPropertyCacheFlagIsDynamicProperty` metadata should not cached at all.
`kComponentPropertyExtendedInfo` = `'meta'` A `CFDictionary` with extended property information. See these constants:
`kComponentPropertyInfoList`
`kComponentPropertyCacheSeed`
`kComponentPropertyCacheFlags`
`kComponentPropertyExtendedInfo`

outPropType
A pointer to the type of the returned property's value.

outPropValueSize
A pointer to the size of the returned property's value.

outPropFlags
On return, a pointer to flags representing the requested information about the property.

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

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

Declared In
`ImageCompression.h`

ICMDecompressionSessionGetTypeID

Returns the type ID for the current decompression session.

```
CFTypeID ICMDecompressionSessionGetTypeID (
   void
);
```

Return Value
A `CFTypeID` value.

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

Declared In
`ImageCompression.h`

ICMDecompressionSessionOptionsCreate

Creates a decompression session options object.

```
OSStatus ICMDecompressionSessionOptionsCreate (
   CFAllocatorRef allocator,
   ICMDecompressionSessionOptionsRef *options
);
```

Parameters

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

`options`
On return, a reference to a decompression session options object.

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

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

Related Sample Code
WhackedTV

Declared In
`ImageCompression.h`

ICMDecompressionSessionOptionsCreateCopy

Copies a decompression session options object.

```
OSStatus ICMDecompressionSessionOptionsCreateCopy (
   CFAllocatorRef allocator,
   ICMDecompressionSessionOptionsRef originalOptions,
   ICMDecompressionSessionOptionsRef *copiedOptions
);
```

Parameters

allocator

An allocator. Pass NULL to use the default allocator.

originalOptions

A decompression session options reference. This reference is returned by `ICMDecompressionSessionOptionsCreate`.

copiedOptions

On return, a reference to a copy of the decompression session options object passed in `originalOptions`.

Return Value

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

Availability

Available in Mac OS X v10.3 and later.

Declared In

`ImageCompression.h`

ICMDecompressionSessionOptionsGetProperty

Retrieves the value of a specific property of a decompression session options object.

```
OSStatus ICMDecompressionSessionOptionsGetProperty (
   ICMDecompressionSessionOptionsRef options,
   ComponentPropertyClass inPropClass,
   ComponentPropertyID inPropID,
   ByteCount inPropValueSize,
   ComponentValuePtr outPropValueAddress,
   ByteCount *outPropValueSizeUsed
);
```

Parameters

options

A decompression session options reference. This reference is returned by `ICMDecompressionSessionOptionsCreate`.

inPropClass

Pass the following constant to define the property class: `kComponentPropertyClassPropertyInfo = 'pnfo'` The property information class. See these constants:

`kComponentPropertyClassPropertyInfo`

inPropID

Pass one of these constants to define the property ID: `kComponentPropertyInfoList` = `'list'` An array of `CFData` values, one for each property. `kComponentPropertyCacheSeed` = `'seed'` A property cache seed value. `kComponentPropertyCacheFlags` = `'flgs'` One of the `kComponentPropertyCache` flags: `kComponentPropertyCacheFlagNotPersistentProperty` metadata should not be saved in persistent cache. `kComponentPropertyCacheFlagIsDynamicProperty` metadata should not cached at all. `kComponentPropertyExtendedInfo` = `'meta'` A `CFDictionary` with extended property information. See these constants:

```
kComponentPropertyInfoList
kComponentPropertyCacheSeed
kComponentPropertyCacheFlags
kComponentPropertyExtendedInfo
```

inPropValueSize

The size of the property value to be retrieved.

outPropValueAddress

A pointer to a variable to hold the value of the property.

outPropValueSizeUsed

On return, a pointer to the number of bytes actually used to store the property value.

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

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

Declared In
`ImageCompression.h`

ICMDecompressionSessionOptionsGetPropertyInfo

Retrieves information about properties of a decompression session options object.

```
OSStatus ICMDecompressionSessionOptionsGetPropertyInfo (
   ICMDecompressionSessionOptionsRef options,
   ComponentPropertyClass inPropClass,
   ComponentPropertyID inPropID,
   ComponentValueType *outPropType,
   ByteCount *outPropValueSize,
   UInt32 *outPropertyFlags
);
```

Parameters

options

A decompression session options reference. This reference is returned by `ICMDecompressionSessionOptionsCreate`.

inPropClass

Pass the following constant to define the property class: `kComponentPropertyClassPropertyInfo` = `'pnfo'` The property information class. See these constants:

```
kComponentPropertyClassPropertyInfo
```

inPropID

Pass one of these constants to define the property ID: `kComponentPropertyInfoList` = `'list'` An array of `CFData` values, one for each property. `kComponentPropertyCacheSeed` = `'seed'` A property cache seed value. `kComponentPropertyCacheFlags` = `'flgs'` One of the `kComponentPropertyCache` flags: `kComponentPropertyCacheFlagNotPersistentProperty` metadata should not be saved in persistent cache. `kComponentPropertyCacheFlagIsDynamicProperty` metadata should not cached at all. `kComponentPropertyExtendedInfo` = `'meta'` A `CFDictionary` with extended property information. See these constants:

```
kComponentPropertyInfoList
kComponentPropertyCacheSeed
kComponentPropertyCacheFlags
kComponentPropertyExtendedInfo
```

outPropType

A pointer to the type of the returned property's value.

outPropValueSize

A pointer to the size of the returned property's value.

outPropFlags

On return, a pointer to flags representing the requested information about the property.

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

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

Declared In
`ImageCompression.h`

ICMDecompressionSessionOptionsGetTypeID

Returns the type ID for the current decompression session options object.

```
CFTypeID ICMDecompressionSessionOptionsGetTypeID (
   void
);
```

Return Value
A `CFTypeID` value.

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

Declared In
`ImageCompression.h`

ICMDecompressionSessionOptionsRelease

Decrements the retain count of a decompression session options object.

```
void ICMDecompressionSessionOptionsRelease (
   ICMDecompressionSessionOptionsRef options
);
```

Parameters

options

A reference to a decompression session options object. This reference is returned by `ICMDecompressionSessionOptionsCreate`. If you pass NULL, nothing happens.

Discussion
If the retain count drops to 0, the object is disposed.

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

Related Sample Code
CaptureAndCompressIPBMovie
WhackedTV

Declared In
`ImageCompression.h`

ICMDecompressionSessionOptionsRetain

Increments the retain count of a decompression session options object.

```
ICMDecompressionSessionOptionsRef ICMDecompressionSessionOptionsRetain (
   ICMDecompressionSessionOptionsRef options
);
```

Parameters

options

A reference to a decompression session options object. This reference is returned by `ICMDecompressionSessionOptionsCreate`. If you pass NULL, nothing happens.

Return Value
A copy of the object reference passed in options, for convenience.

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

Declared In
`ImageCompression.h`

ICMDecompressionSessionOptionsSetProperty

Sets the value of a specific property of a decompression session options object.

```
OSStatus ICMDecompressionSessionOptionsSetProperty (
   ICMDecompressionSessionOptionsRef options,
   ComponentPropertyClass inPropClass,
   ComponentPropertyID inPropID,
   ByteCount inPropValueSize,
   ConstComponentValuePtr inPropValueAddress
);
```

Parameters

options

A decompression session options reference. This reference is returned by `ICMDecompressionSessionOptionsCreate`.

inPropClass

Pass the following constant to define the property class: `kComponentPropertyClassPropertyInfo` = `'pnfo'` The property information class. See these constants:

`kComponentPropertyClassPropertyInfo`

inPropID

Pass one of these constants to define the property ID: `kComponentPropertyInfoList` = `'list'` An array of `CFData` values, one for each property. `kComponentPropertyCacheSeed` = `'seed'` A property cache seed value. `kComponentPropertyCacheFlags` = `'flgs'` One of the `kComponentPropertyCache` flags: `kComponentPropertyCacheFlagNotPersistentProperty` metadata should not be saved in persistent cache. `kComponentPropertyCacheFlagIsDynamicProperty` metadata should not cached at all. `kComponentPropertyExtendedInfo` = `'meta'` A `CFDictionary` with extended property information. See these constants:

`kComponentPropertyInfoList`

`kComponentPropertyCacheSeed`

`kComponentPropertyCacheFlags`

`kComponentPropertyExtendedInfo`

inPropValueSize

The size of the property value to be set.

inPropValueAddress

A pointer to the value of the property to be set.

Return Value

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

Availability

Available in Mac OS X v10.3 and later.

Related Sample Code

WhackedTV

Declared In

`ImageCompression.h`

ICMDecompressionSessionRelease

Decrements the retain count of a decompression session.

```
void ICMDecompressionSessionRelease (
   ICMDecompressionSessionRef session
);
```

Parameters

session

A decompression session reference. This reference is returned by `ICMDecompressionSessionCreate`. If you pass NULL, nothing happens.

Discussion
If the retain count drops to 0, the object is disposed.

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

Related Sample Code
CaptureAndCompressIPBMovie
MovieVideoChart
QTQuartzPlayer
WhackedTV

Declared In
`ImageCompression.h`

ICMDecompressionSessionRetain

Increments the retain count of a decompression session.

```
ICMDecompressionSessionRef ICMDecompressionSessionRetain (
   ICMDecompressionSessionRef session
);
```

Parameters

session

A decompression session reference. This reference is returned by `ICMDecompressionSessionCreate`. If you pass NULL, nothing happens.

Return Value
A copy of the reference passed in session, for convenience.

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

Declared In
`ImageCompression.h`

ICMDecompressionSessionSetNonScheduledDisplayDirection

Sets the direction for non-scheduled display time.

```
OSStatus ICMDecompressionSessionSetNonScheduledDisplayDirection (
   ICMDecompressionSessionRef session,
   Fixed rate
);
```

Parameters

session

A decompression session reference. This reference is returned by `ICMDecompressionSessionCreate`.

rate

The display direction. Negative values represent backward display and positive values represent forward display.

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

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

Declared In
`ImageCompression.h`

ICMDecompressionSessionSetNonScheduledDisplayTime

Sets the display time for a decompression session, and requests display of the non-scheduled queued frame at that display time, if there is one.

```
OSStatus ICMDecompressionSessionSetNonScheduledDisplayTime (
   ICMDecompressionSessionRef session,
   TimeValue64 displayTime,
   TimeScale displayTimeScale,
   UInt32 flags
);
```

Parameters

session

A decompression session reference. This reference is returned by `ICMDecompressionSessionCreate`.

displayTime

A display time. Usually this is the display time of a non-scheduled queued frame.

displayTimeScale

The timescale according to which `displayTime` should be interpreted.

flags

Reserved; set to 0.

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

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

Related Sample Code
CaptureAndCompressIPBMovie
QTQuartzPlayer
WhackedTV

Declared In
ImageCompression.h

ICMDecompressionSessionSetProperty

Sets the value of a specific property of a decompression session.

```
OSStatus ICMDecompressionSessionSetProperty (
   ICMDecompressionSessionRef session,
   ComponentPropertyClass inPropClass,
   ComponentPropertyID inPropID,
   ByteCount inPropValueSize,
   ConstComponentValuePtr inPropValueAddress
);
```

Parameters

session

A decompression session reference. This reference is returned by ICMDecompressionSessionCreate.

inPropClass

Pass the following constant to define the property class: kComponentPropertyClassPropertyInfo = 'pnfo' The property information class. See these constants:

kComponentPropertyClassPropertyInfo

inPropID

Pass one of these constants to define the property ID: kComponentPropertyInfoList = 'list' An array of CFData values, one for each property. kComponentPropertyCacheSeed = 'seed' A property cache seed value. kComponentPropertyCacheFlags = 'flgs' One of the kComponentPropertyCache flags: kComponentPropertyCacheFlagNotPersistentProperty metadata should not be saved in persistent cache. kComponentPropertyCacheFlagIsDynamicProperty metadata should not cached at all. kComponentPropertyExtendedInfo = 'meta' A CFDictionary with extended property information. See these constants:

kComponentPropertyInfoList

kComponentPropertyCacheSeed

kComponentPropertyCacheFlags

kComponentPropertyExtendedInfo

inPropValueSize

The size in bytes of the property's value.

inPropValueAddress

A pointer to the property value to be set.

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

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

Declared In
ImageCompression.h

ICMEncodedFrameCreateMutable

Called by a compressor to create an encoded-frame token corresponding to a given source frame.

```
OSStatus ICMEncodedFrameCreateMutable (
   ICMCompressorSessionRef session,
   ICMCompressorSourceFrameRef sourceFrame,
   ByteCount bufferSize,
   ICMMutableEncodedFrameRef *frameOut
);
```

Parameters

session

A reference to the compression session between the ICM and an image compressor component.

sourceFrame

A reference to a frame that has been passed in `sourceFrameRefCon` to `ICMCompressionSessionEncodeFrame` (page 33).

bufferSize

The size of the frame buffer in bytes.

frameOut

On return, a reference to an encoded frame object with write capabilities.

Return Value

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

Discussion

The encoded frame will initially show 0 for `mediaSampleFlags`; if the frame is not a key frame, the compressor must call `ICMEncodedFrameSetMediaSampleFlags` to set `mediaSampleNotSync`. If the frame is droppable, the compressor should set `mediaSampleDroppable`. If the frame is a partial key frame, the compressor should set `mediaSamplePartialSync`.

Availability

Available in Mac OS X v10.3 and later.

Related Sample Code

ExampleIPBCodec

Declared In

`ImageCompression.h`

ICMEncodedFrameGetBufferSize

Gets the size of an encoded frame's data buffer.

```
ByteCount ICMEncodedFrameGetBufferSize (
   ICMEncodedFrameRef frame
);
```

Parameters

frame

A reference to an encoded frame object.

Return Value

The physical size in bytes of the encoded frame's data buffer.

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

Declared In
`ImageCompression.h`

ICMEncodedFrameGetDataPtr

Gets the data buffer for an encoded frame.

```
UInt8 * ICMEncodedFrameGetDataPtr (
   ICMEncodedFrameRef frame
);
```

Parameters

frame

A reference to an encoded frame object.

Return Value
A pointer to the object's data buffer.

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

Related Sample Code
CaptureAndCompressIPBMovie
ExampleIPBCodec
Quartz Composer QCTV

Declared In
`ImageCompression.h`

ICMEncodedFrameGetDataSize

Gets the data size of the compressed frame in an encoded frame's buffer.

```
ByteCount ICMEncodedFrameGetDataSize (
   ICMEncodedFrameRef frame
);
```

Parameters

frame

A reference to an encoded frame object.

Return Value
The logical size in bytes of the encoded frame's data buffer, which may be less than the physical size of the buffer.

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

Related Sample Code
CaptureAndCompressIPBMovie
Quartz Composer QCTV

Declared In
`ImageCompression.h`

ICMEncodedFrameGetDecodeDuration

Retrieves an encoded frame's decode duration.

```
TimeValue64 ICMEncodedFrameGetDecodeDuration (
   ICMEncodedFrameRef frame
);
```

Parameters

frame

A reference to an encoded frame object.

Return Value
The encoded frame's decode duration.

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

Related Sample Code
CaptureAndCompressIPBMovie
OpenGLCaptureToMovie
Quartz Composer QCTV

Declared In
`ImageCompression.h`

ICMEncodedFrameGetDecodeNumber

Retrieves the decode number of an encoded frame.

```
UInt32 ICMEncodedFrameGetDecodeNumber (
   ICMEncodedFrameRef frame
);
```

Parameters

frame

A reference to an encoded frame object.

Return Value
The decode number of the encoded frame.

Discussion
The ICM automatically stamps ascending decode numbers on frames after the compressor emits them. The first decode number in session is 1. Compressors should not call this function.

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

Declared In
`ImageCompression.h`

ICMEncodedFrameGetDecodeTimeStamp

Retrieves an encoded frame's decode time stamp.

```
TimeValue64 ICMEncodedFrameGetDecodeTimeStamp (
   ICMEncodedFrameRef frame
);
```

Parameters

frame

A reference to an encoded frame object.

Return Value
The encoded frame's decode time stamp.

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

Related Sample Code
CaptureAndCompressIPBMovie

Declared In
`ImageCompression.h`

ICMEncodedFrameGetDisplayDuration

Retrieves an encoded frame's display duration.

```
TimeValue64 ICMEncodedFrameGetDisplayDuration (
   ICMEncodedFrameRef frame
);
```

Parameters

frame

A reference to an encoded frame object.

Return Value
The encoded frame's display duration.

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

Declared In
`ImageCompression.h`

ICMEncodedFrameGetDisplayOffset

Retrieves an encoded frame's display offset.

```
TimeValue64 ICMEncodedFrameGetDisplayOffset (
   ICMEncodedFrameRef frame
);
```

Parameters

frame

A reference to an encoded frame object.

Return Value
The encoded frame's display offset. This is the time offset from decode time stamp to display time stamp.

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

Related Sample Code
CaptureAndCompressIPBMovie

Declared In
`ImageCompression.h`

ICMEncodedFrameGetDisplayTimeStamp

Retrieves an encoded frame's display time stamp.

```
TimeValue64 ICMEncodedFrameGetDisplayTimeStamp (
   ICMEncodedFrameRef frame
);
```

Parameters

frame

A reference to an encoded frame object.

Return Value
The encoded frame's display time stamp.

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

Related Sample Code
CaptureAndCompressIPBMovie

Declared In
`ImageCompression.h`

ICMEncodedFrameGetFrameType

Retrieves the frame type for an encoded frame.

```
ICMFrameType ICMEncodedFrameGetFrameType (
   ICMEncodedFrameRef frame
);
```

Parameters

frame

A reference to an encoded frame object.

Return Value
The encoded frame's frame type (see below).

Discussion
This function returns one of these values:

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

Declared In
`ImageCompression.h`

ICMEncodedFrameGetImageDescription

Retrieves the image description of an encoded frame.

```
OSStatus ICMEncodedFrameGetImageDescription (
   ICMEncodedFrameRef frame,
   ImageDescriptionHandle *imageDescOut
);
```

Parameters

frame

A reference to an encoded frame object.

imageDescOut

A pointer to a handle containing the encoded frame's image description. The caller should not dispose of this handle.

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

Discussion
This function returns the same image description handle as `ICMCompressionSessionGetImageDescription` (page 34).

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

Related Sample Code
CaptureAndCompressIPBMovie

Declared In
`ImageCompression.h`

ICMEncodedFrameGetMediaSampleFlags

Retrieves the media sample flags for an encoded frame.

```
MediaSampleFlags ICMEncodedFrameGetMediaSampleFlags (
   ICMEncodedFrameRef frame
);
```

Parameters

frame

A reference to an encoded frame object.

Return Value

The object's media sample flags. These flags are listed in the header file `Movies.h`.

Availability

Available in Mac OS X v10.3 and later.

Related Sample Code

CaptureAndCompressIPBMovie

Declared In

`ImageCompression.h`

ICMEncodedFrameGetSimilarity

Retrieves the similarity value for an encoded frame.

```
Float32 ICMEncodedFrameGetSimilarity (
   ICMEncodedFrameRef frame
);
```

Parameters

frame

A reference to an encoded frame object.

Return Value

The encoded frame's similarity value. 1.0 means identical; 0.0 means not at all alike. The default value is -1.0, which means unknown.

Availability

Available in Mac OS X v10.3 and later.

Declared In

`ImageCompression.h`

ICMEncodedFrameGetSourceFrameRefCon

Retrieves the reference value of an encoded frame's source frame.

```
void * ICMEncodedFrameGetSourceFrameRefCon (
   ICMEncodedFrameRef frame
);
```

Parameters

frame

A reference to an encoded frame object.

Discussion

The source frame's reference value is copied from the session's `sourceFrameRefCon` parameter that was passed to `ICMCompressionSessionEncodeFrame` (page 33).

Availability

Available in Mac OS X v10.3 and later.

Declared In

`ImageCompression.h`

ICMEncodedFrameGetTimeScale

Retrieves the timescale of an encoded frame.

```
TimeScale ICMEncodedFrameGetTimeScale (
   ICMEncodedFrameRef frame
);
```

Parameters

frame

A reference to an encoded frame object.

Return Value

The time scale of an encoded frame. This is always the same as the time scale of the compression session.

Availability

Available in Mac OS X v10.3 and later.

Related Sample Code

CaptureAndCompressIPBMovie

Declared In

`ImageCompression.h`

ICMEncodedFrameGetTypeID

Returns the type ID for the current encoded frame object.

```
CFTypeID ICMEncodedFrameGetTypeID (
   void
);
```

Return Value

A `CFTypeID` value.

Availability

Available in Mac OS X v10.3 and later.

Declared In
ImageCompression.h

ICMEncodedFrameGetValidTimeFlags

Retrieves an encoded frame's flags indicating which of its time stamps and durations are valid.

```
ICMValidTimeFlags ICMEncodedFrameGetValidTimeFlags (
   ICMEncodedFrameRef frame
);
```

Parameters

frame

A reference to an encoded frame object.

Return Value
One of the constants listed below.

Discussion
This function returns one of these values:

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

Related Sample Code
CaptureAndCompressIPBMovie

Declared In
ImageCompression.h

ICMEncodedFrameRelease

Decrements the retain count of an encoded frame object.

```
void ICMEncodedFrameRelease (
   ICMEncodedFrameRef frame
);
```

Parameters

frame

A reference to an encoded frame object. If you pass NULL, nothing happens.

Discussion
If the retain count drops to 0, the object is disposed.

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

Related Sample Code
ExampleIPBCodec

Declared In
ImageCompression.h

ICMEncodedFrameRetain

Increments the retain count of an encoded frame object.

```
ICMEncodedFrameRef ICMEncodedFrameRetain (
   ICMEncodedFrameRef frame
);
```

Parameters

frame

A reference to an encoded frame object. If you pass NULL, nothing happens.

Return Value
A reference to the object passed in frame, for convenience.

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

Declared In
`ImageCompression.h`

ICMEncodedFrameSetDataSize

Sets the data size of the compressed frame in an encoded frame's buffer.

```
OSStatus ICMEncodedFrameSetDataSize (
   ICMMutableEncodedFrameRef frame,
   ByteCount dataSize
);
```

Parameters

frame

A reference to an encoded frame object with write capabilities.

dataSize

The data size of the compressed frame in the encoded frame object's buffer.

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

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

Related Sample Code
ExampleIPBCodec

Declared In
`ImageCompression.h`

ICMEncodedFrameSetDecodeDuration

Sets an encoded frame's decode duration.

```
OSStatus ICMEncodedFrameSetDecodeDuration (
   ICMMutableEncodedFrameRef frame,
   TimeValue64 decodeDuration
);
```

Parameters

frame

A reference to an encoded frame object with write capabilities.

decodeDuration

The encoded frame's decode duration.

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

Discussion
This function automatically sets the `kICMValidTime_DecodeDurationIsValid` flag.

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

Declared In
`ImageCompression.h`

ICMEncodedFrameSetDecodeTimeStamp

Sets an encoded frame's decode time stamp.

```
OSStatus ICMEncodedFrameSetDecodeTimeStamp (
   ICMMutableEncodedFrameRef frame,
   TimeValue64 decodeTimeStamp
);
```

Parameters

frame

A reference to an encoded frame object with write capabilities.

decodeTimeStamp

The encoded frame's decode time stamp.

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

Discussion
This function automatically sets the `kICMValidTime_DecodeTimeStampIsValid` flag. If the display time stamp is valid, it also sets the `kICMValidTime_DisplayOffsetIsValid` flag.

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

Declared In
`ImageCompression.h`

ICMEncodedFrameSetDisplayDuration

Sets an encoded frame's display duration.

```
OSStatus ICMEncodedFrameSetDisplayDuration (
   ICMMutableEncodedFrameRef frame,
   TimeValue64 displayDuration
);
```

Parameters

frame

A reference to an encoded frame object with write capabilities.

displayDuration

The encoded frame's display duration.

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

Discussion
This function automatically sets the `kICMValidTime_DisplayDurationIsValid` flag.

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

Declared In
`ImageCompression.h`

ICMEncodedFrameSetDisplayTimeStamp

Sets an encoded frame's display time stamp.

```
OSStatus ICMEncodedFrameSetDisplayTimeStamp (
   ICMMutableEncodedFrameRef frame,
   TimeValue64 displayTimeStamp
);
```

Parameters

frame

A reference to an encoded frame object with write capabilities.

displayTimeStamp

The encoded frame's display time stamp.

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

Discussion
This function automatically sets the `kICMValidTime_DisplayTimeStampIsValid` flag. If the decode time stamp is valid, it also sets the `kICMValidTime_DisplayOffsetIsValid` flag.

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

Declared In
`ImageCompression.h`

ICMEncodedFrameSetFrameType

Sets the frame type for an encoded frame.

```
OSStatus ICMEncodedFrameSetFrameType (
   ICMMutableEncodedFrameRef frame,
   ICMFrameType frameType
);
```

Parameters

frame

A reference to an encoded frame object with write capabilities.

frameType

The frame type to be set: `kICMFrameType_I` = `'I'` An I frame. `kICMFrameType_P` = `'P'` A P frame. `kICMFrameType_B` = `'B'` A B frame. `kICMFrameType_Unknown` = 0 A frame of unknown type. See these constants:

`kICMFrameType_I`

`kICMFrameType_P`

`kICMFrameType_B`

`kICMFrameType_Unknown`

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

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

Related Sample Code
ExampleIPBCodec

Declared In
`ImageCompression.h`

ICMEncodedFrameSetMediaSampleFlags

Sets the media sample flags for an encoded frame.

```
OSStatus ICMEncodedFrameSetMediaSampleFlags (
   ICMMutableEncodedFrameRef frame,
   MediaSampleFlags mediaSampleFlags
);
```

Parameters

frame

A reference to an encoded frame object with write capabilities.

mediaSampleFlags

The object's media sample flags. These flags are listed in the header file `Movies.h`.

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

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

Related Sample Code
ExampleIPBCodec

Declared In
`ImageCompression.h`

ICMEncodedFrameSetSimilarity

Sets the similarity for an encoded frame.

```
OSStatus ICMEncodedFrameSetSimilarity (
   ICMMutableEncodedFrameRef frame,
   Float32 similarity
);
```

Parameters

frame

A reference to an encoded frame object with write capabilities.

similarity

The encoded frame's similarity value to be set. 1.0 means identical; 0.0 means not at all alike. The default value is -1.0, which means unknown.

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

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

Declared In
`ImageCompression.h`

ICMEncodedFrameSetValidTimeFlags

Sets an encoded frame's flags that indicate which of its time stamps and durations are valid.

```
OSStatus ICMEncodedFrameSetValidTimeFlags (
   ICMMutableEncodedFrameRef frame,
   ICMValidTimeFlags validTimeFlags
);
```

Parameters

frame

A reference to an encoded frame object with write capabilities.

validTimeFlags

One of the following constants: `kICMValidTime_DisplayTimeStampIsValid` = 1L<<0 The value of `displayTimeStamp` is valid. `kICMValidTime_DisplayDurationIsValid` = 1L<<1 The value of `displayDuration` is valid. See these constants:

`kICMValidTime_DisplayTimeStampIsValid`

`kICMValidTime_DisplayDurationIsValid`

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

Discussion
Setting an encoded frame's decode or display time stamp or duration automatically sets the corresponding valid time flags. For example, calling `ICMEncodedFrameSetDecodeTimeStamp` sets `kICMValidTime_DisplayTimeStampIsValid`. If both the encoded frame's decode time stamp and display time stamp are valid, `kICMValidTime_DisplayOffsetIsValid` is automatically set.

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

Declared In
`ImageCompression.h`

ICMImageDescriptionGetProperty

Returns a particular property of a image description handle.

```
OSStatus ICMImageDescriptionGetProperty (
   ImageDescriptionHandle inDesc,
   ComponentPropertyClass inPropClass,
   ComponentPropertyID inPropID,
   ByteCount inPropValueSize,
   ComponentValuePtr outPropValueAddress,
   ByteCount *outPropValueSizeUsed
);
```

Parameters

inDesc
The image description handle being interrogated.

inPropClass
The class of property being requested.

inPropID
The ID of the property being requested.

inPropValueSize
The size of the property value buffer.

outPropValueAddress
Points to the buffer to receive the property value.

outPropValueSizeUsed
Points to a variable to receive the actual size of returned property value. (This can be NULL).

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

Discussion
This routine returns a particular property of a image description handle.

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

Related Sample Code
CaptureAndCompressIPBMovie
WhackedTV

Declared In
`ImageCompression.h`

ICMImageDescriptionGetPropertyInfo

Returns information about a particular property of a image description.

```
OSStatus ICMImageDescriptionGetPropertyInfo (
   ImageDescriptionHandle inDesc,
   ComponentPropertyClass inPropClass,
   ComponentPropertyID inPropID,
   ComponentValueType *outPropType,
   ByteCount *outPropValueSize,
   UInt32 *outPropertyFlags
);
```

Parameters

inDesc
The image description handle being interrogated.

inPropClass
The class of property being requested.

inPropID
The ID of the property being requested.

outPropType
The type of property is returned here. (This can be NULL).

outPropValueSize
The size of property is returned here. (This can be NULL).

outPropertyFlags
The property flags are returned here. (This can be NULL).

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

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

Declared In
`ImageCompression.h`

ICMImageDescriptionSetProperty

Sets a particular property of a image description handle.

```
OSStatus ICMImageDescriptionSetProperty (
   ImageDescriptionHandle inDesc,
   ComponentPropertyClass inPropClass,
   ComponentPropertyID inPropID,
   ByteCount inPropValueSize,
   ConstComponentValuePtr inPropValueAddress
);
```

Parameters

inDesc

The image description handle being modified.

inPropClass

The class of property being set.

inPropID

The ID of the property being set.

inPropValueSize

The size of property value.

inPropValueAddress

Points to the property value buffer.

Return Value

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

Availability

Available in Mac OS X v10.3 and later.

Related Sample Code

ExampleIPBCodec

SoftVDigX

WhackedTV

Declared In

`ImageCompression.h`

ICMMultiPassStorageCopyDataAtTimeStamp

Called by a multipass-capable compressor to retrieve data at a given time stamp.

```
OSStatus ICMMultiPassStorageCopyDataAtTimeStamp (
   ICMMultiPassStorageRef multiPassStorage,
   TimeValue64 timeStamp,
   long index,
   CFMutableDataRef *dataOut
);
```

Parameters

multiPassStorage

The multipass storage object.

timeStamp

The time stamp at which the value should be retrieved.

index

An index by which multiple values may be stored at a time stamp. The meaning of individual indexes is private to the compressor.

dataOut

A pointer to memory to receive the data at the time stamp.

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

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

Declared In
`ImageCompression.h`

ICMMultiPassStorageCreateWithCallbacks

Assembles a multipass storage mechanism from callbacks.

```
OSStatus ICMMultiPassStorageCreateWithCallbacks (
   CFAllocatorRef allocator,
   ICMMultiPassStorageCallbacks *callbacks,
   ICMMultiPassStorageRef *multiPassStorageOut
);
```

Parameters

allocator

An allocator for this task. Pass NULL to use the default allocator.

callbacks

A structure containing a collection of callbacks for creating a custom multipass storage object. See `ICMMultiPassStorageCallbacks`.

multiPassStorageOut

A reference to the new multipass storage object.

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

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

Declared In
`ImageCompression.h`

ICMMultiPassStorageCreateWithTemporaryFile

Creates multipass storage using a temporary file.

```
OSStatus ICMMultiPassStorageCreateWithTemporaryFile (
   CFAllocatorRef allocator,
   FSRef *directoryRef,
   CFStringRef fileName,
   ICMMultiPassStorageCreationFlags flags,
   ICMMultiPassStorageRef *multiPassStorageOut
);
```

Parameters

allocator

An allocator for this task. Pass NULL to use the default allocator.

directoryRef

A reference to a file directory. If you pass NULL, the ICM will use the user's Temporary Items folder.

fileName

A file name to use for the storage. If you pass NULL, the ICM will pick a unique name. If you pass the name of a file that already exists, the ICM will assume you are continuing a previous multipass session where you left off. This file will be deleted when the multipass storage is released, unless you set the `kICMMultiPassStorage_DoNotDeleteWhenDone` flag.

flags

Flag controlling this process: `kICMMultiPassStorage_DoNotDeleteWhenDone` = 1L<<0 The temporary file should not be deleted when the multipass storage is released. See these constants:

`kICMMultiPassStorage_DoNotDeleteWhenDone`

multiPassStorageOut

A reference to the new multipass storage.

Return Value

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

Availability

Available in Mac OS X v10.3 and later.

Declared In

`ImageCompression.h`

ICMMultiPassStorageGetTimeStamp

Called by a multipass-capable compressor to retrieve a time stamp for which a value is stored.

```
OSStatus ICMMultiPassStorageGetTimeStamp (
   ICMMultiPassStorageRef multiPassStorage,
   TimeValue64 fromTimeStamp,
   ICMMultiPassStorageStep step,
   TimeValue64 *timeStampOut
);
```

Parameters

multiPassStorage

The multipass storage object.

fromTimeStamp

The initial time stamp. This value is ignored for some values of step.

step

Indicates the kind of time stamp search to perform: `kICMMultiPassStorage_GetFirstTimeStamp` = 1 Requests the first time stamp at which a value is stored. `kICMMultiPassStorage_GetPreviousTimeStamp` = 2 Requests the previous time stamp before the time stamp specified in `fromTimeStamp` at which a value is stored. `kICMMultiPassStorage_GetNextTimeStamp` = 3 Requests the next time stamp after the time stamp specified in `fromTimeStamp` at which a value is stored. `kICMMultiPassStorage_GetLastTimeStamp` = 4 Requests the last time stamp at which a value is stored. See these constants:

```
kICMMultiPassStorage_GetFirstTimeStamp
kICMMultiPassStorage_GetPreviousTimeStamp
kICMMultiPassStorage_GetNextTimeStamp
kICMMultiPassStorage_GetLastTimeStamp
```

timeStampOut

A pointer to a TimeValue64 value to receive the found time stamp. It will be set to -1 if no time stamp is found.

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

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

Declared In
`ImageCompression.h`

ICMMultiPassStorageGetTypeID

Returns the type ID for the current multipass storage object.

```
CFTypeID ICMMultiPassStorageGetTypeID (
   void
);
```

Return Value
A `CFTypeID` value.

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

Declared In
`ImageCompression.h`

ICMMultiPassStorageRelease

Decrements the retain count of a multipass storage object.

```
void ICMMultiPassStorageRelease (
   ICMMultiPassStorageRef multiPassStorage
);
```

Parameters

multiPassStorageOut

A reference to a multipass storage object. You can create this object using `ICMMultiPassStorageCreateWithTemporaryFile` or `ICMMultiPassStorageCreateWithCallbacks`. If you pass NULL, nothing happens.

Discussion

If the retain count drops to 0, the object is disposed.

Availability

Available in Mac OS X v10.3 and later.

Declared In

`ImageCompression.h`

ICMMultiPassStorageRetain

Increments the retain count of a multipass storage object.

```
ICMMultiPassStorageRef ICMMultiPassStorageRetain (
   ICMMultiPassStorageRef multiPassStorage
);
```

Parameters

multiPassStorageOut

A reference to a multipass storage object. You can create this object using `ICMMultiPassStorageCreateWithTemporaryFile` or `ICMMultiPassStorageCreateWithCallbacks`. If you pass NULL, nothing happens.

Return Value

A reference to the object passed in `multiPassStorage`, for convenience.

Availability

Available in Mac OS X v10.3 and later.

Declared In

`ImageCompression.h`

ICMMultiPassStorageSetDataAtTimeStamp

Called by a multipass-capable compressor to store data at a given time stamp.

```
OSStatus ICMMultiPassStorageSetDataAtTimeStamp (
   ICMMultiPassStorageRef multiPassStorage,
   TimeValue64 timeStamp,
   long index,
   CFDataRef data
);
```

Parameters

multiPassStorage

The multipass storage object.

timeStamp

The time stamp at which the value should be stored.

index

An index by which multiple values may be stored at a time stamp. The meaning of individual indexes is private to the compressor.

data

The data to be stored, or NULL to delete the value.

Return Value

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

Discussion

The new data replaces any previous data held at that time stamp. If the value of data is NULL, the data for that time stamp is deleted. The format of the data is private to the compressor.

Availability

Available in Mac OS X v10.3 and later.

Declared In

`ImageCompression.h`

ImageTranscoderBeginSequence

Initiates an image transcoding sequence and specifies the input data format.

```
ComponentResult ImageTranscoderBeginSequence (
   ImageTranscoderComponent itc,
   ImageDescriptionHandle srcDesc,
   ImageDescriptionHandle *dstDesc,
   void *data,
   long dataSize
);
```

Parameters

itc

The image transcoder component.

srcDesc

The `ImageDescription` structure for the source compressed image data.

dstDesc

On return, a new `ImageDescription` structure.

data

First frame of data to be transcoded (may be `NIL`).

dataSize
Size of compressed image data pointed to by the data.

Return Value
See Error Codes. Returns noErr if there is no error.

Discussion
This function specifies the format of source compressed image data in the srcDesc parameter. The image transcoder should allocate a new ImageDescription structure and return it in the dstDesc parameter. The new ImageDescription structure should be a completely filled out image description which is sufficient for correctly decompressing the data generated by subsequent calls to ImageTranscoderConvert (page 100).

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
ImageCompression.h

ImageTranscoderConvert

Performs image transcoding operations.

```
ComponentResult ImageTranscoderConvert (
   ImageTranscoderComponent itc,
   void *srcData,
   long srcDataSize,
   void **dstData,
   long *dstDataSize
);
```

Parameters

itc
The image transcoder component.

srcData
A pointer to the source compressed image data to transcode.

srcDataSize
The size of the source image data, in bytes.

dstData
On return, a pointer to the transcoded data.

dstDataSize
On return, the size of the transcoded data in bytes.

Return Value
See Error Codes. Returns noErr if there is no error.

Discussion
The image transcoder component is responsible for allocating storage for the transcoded data, transcoding the data, and returning a pointer to the transcoded data in the dstData parameter. The size of the transcoded data in bytes should be returned in the dstDataSize parameter. The caller is responsible for disposing of the transcoded data using ImageTranscoderDisposeData (page 101).

The memory allocated to store the transcoded image data must not be in an unlocked handle. Even if the image transcoding operation can be performed in place, the transcoded data must be placed in a separate block of memory from the source data. The image transcoder component must not write back into the source image data.

The responsibility for allocating the buffer for the transcoded data has been placed in the transcoder with the intent that some hardware manufacturers may find it useful to place the transcoded data directly into on-board memory on their video board. If the transcoding operation is being performed on a QuickTime movie, the transcoded data pointer will be almost immediately passed on to a decompressor. If the decompressor is implemented in hardware, performance may be increased because the transcoded data is already loaded onto the decompression hardware.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`ImageCompression.h`

ImageTranscoderDisposeData

Disposes of transcoded data.

```
ComponentResult ImageTranscoderDisposeData (
   ImageTranscoderComponent itc,
   void *dstData
);
```

Parameters

itc
The image transcoder component.

dstData
A pointer to the transcoded data.

Return Value
See `Error Codes`. Returns `noErr` if there is no error.

Discussion
When the client of the image transcoder component is done with a piece of transcoded data, this function must be called with a pointer to the transcoded data. The image transcoder component should not make any assumptions about the maximum number of outstanding pieces of transcoded data or the order in which the transcoding data will be disposed.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`ImageCompression.h`

ImageTranscoderEndSequence

Ends an image transcoding sequence.

```
ComponentResult ImageTranscoderEndSequence (
   ImageTranscoderComponent itc
);
```

Parameters

itc

The image transcoder component whose transcoder sequence is ending.

Return Value

See `Error Codes`. Returns `noErr` if there is no error.

Discussion

`ImageTranscoderEndSequence` is called when there are no more frames of data to be transcoded using the parameters specified in the previous call to `ImageTranscoderBeginSequence` (page 99). After calling this function, the component will either be closed or receive another call to `ImageTranscoderBeginSequence` with a different `ImageDescription` structure. For example, the dimensions of the source image may be different.

Version Notes

Introduced in QuickTime 3 or earlier.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`ImageCompression.h`

NewICMAlignmentUPP

Allocates a Universal Procedure Pointer for the ICMAlignmentProc callback.

```
ICMAlignmentUPP NewICMAlignmentUPP (
   ICMAlignmentProcPtr userRoutine
);
```

Parameters

userRoutine

A pointer to your application-defined function.

Return Value

A new UPP; see `Universal Procedure Pointers`.

Discussion

This function is used with Macintosh PowerPC systems. See *Inside Macintosh: PowerPC System Software*.

Version Notes

Introduced in QuickTime 4.1. Replaces `NewICMAlignmentProc`.

Availability

Available in Mac OS X v10.0 and later.

Declared In
`ImageCompression.h`

NewICMCompletionUPP

Allocates a Universal Procedure Pointer for the ICMCompletionProc callback.

```
ICMCompletionUPP NewICMCompletionUPP (
   ICMCompletionProcPtr userRoutine
);
```

Parameters

`userRoutine`
A pointer to your application-defined function.

Return Value
A new UPP; see `Universal Procedure Pointers`.

Discussion
This function is used with Macintosh PowerPC systems. See *Inside Macintosh: PowerPC System Software*.

Version Notes
Introduced in QuickTime 4.1. Replaces `NewICMCompletionProc`.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`ImageCompression.h`

NewICMConvertDataFormatUPP

Allocates a Universal Procedure Pointer for the ICMConvertDataFormatProc callback.

```
ICMConvertDataFormatUPP NewICMConvertDataFormatUPP (
   ICMConvertDataFormatProcPtr userRoutine
);
```

Parameters

`userRoutine`
A pointer to your application-defined function.

Return Value
A new UPP; see `Universal Procedure Pointers`.

Discussion
This function is used with Macintosh PowerPC systems. See *Inside Macintosh: PowerPC System Software*.

Version Notes
Introduced in QuickTime 4.1. Replaces `NewICMConvertDataFormatProc`.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`ImageCompression.h`

NewICMCursorShieldedUPP

Allocates a Universal Procedure Pointer for the ICMCursorShieldedProc callback.

```
ICMCursorShieldedUPP NewICMCursorShieldedUPP (
   ICMCursorShieldedProcPtr userRoutine
);
```

Parameters

`userRoutine`
A pointer to your application-defined function.

Return Value
A new UPP; see `Universal Procedure Pointers`.

Discussion
This function is used with Macintosh PowerPC systems. See *Inside Macintosh: PowerPC System Software*.

Version Notes
Introduced in QuickTime 4.1. Replaces `NewICMCursorShieldedProc`.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`ImageCompression.h`

NewICMDataUPP

Allocates a Universal Procedure Pointer for the ICMDataProc callback.

```
ICMDataUPP NewICMDataUPP (
   ICMDataProcPtr userRoutine
);
```

Parameters

`userRoutine`
A pointer to your application-defined function.

Return Value
A new UPP; see `Universal Procedure Pointers`.

Discussion
This function is used with Macintosh PowerPC systems. See *Inside Macintosh: PowerPC System Software*.

Version Notes
Introduced in QuickTime 4.1. Replaces `NewICMDataProc`.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`ImageCompression.h`

NewICMFlushUPP

Allocates a Universal Procedure Pointer for the ICMFlushProc callback.

```
ICMFlushUPP NewICMFlushUPP (
   ICMFlushProcPtr userRoutine
);
```

Parameters

`userRoutine`
A pointer to your application-defined function.

Return Value
A new UPP; see `Universal Procedure Pointers`.

Discussion
This function is used with Macintosh PowerPC systems. See *Inside Macintosh: PowerPC System Software*.

Version Notes
Introduced in QuickTime 4.1. Replaces `NewICMFlushProc`.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`ImageCompression.h`

NewICMMemoryDisposedUPP

Allocates a Universal Procedure Pointer for the ICMMemoryDisposedProc callback.

```
ICMMemoryDisposedUPP NewICMMemoryDisposedUPP (
   ICMMemoryDisposedProcPtr userRoutine
);
```

Parameters

`userRoutine`
A pointer to your application-defined function.

Return Value
A new UPP; see `Universal Procedure Pointers`.

Discussion
This function is used with Macintosh PowerPC systems. See *Inside Macintosh: PowerPC System Software*.

Version Notes
Introduced in QuickTime 4.1. Replaces `NewICMMemoryDisposedProc`.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`ImageCompression.h`

NewICMProgressUPP

Allocates a Universal Procedure Pointer for the ICMProgressProc callback.

```
ICMProgressUPP NewICMProgressUPP (
   ICMProgressProcPtr userRoutine
);
```

Parameters

`userRoutine`
A pointer to your application-defined function.

Return Value
A new UPP; see `Universal Procedure Pointers`.

Discussion
This function is used with Macintosh PowerPC systems. See *Inside Macintosh: PowerPC System Software*.

Version Notes
Introduced in QuickTime 4.1. Replaces `NewICMProgressProc`.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
qtdataexchange
qtdataexchange.win
ThreadsExporter
ThreadsImporter

Declared In
`ImageCompression.h`

NewQDPixUPP

Allocates a Universal Procedure Pointer for the QDPixProc callback.

```
QDPixUPP NewQDPixUPP (
   QDPixProcPtr userRoutine
);
```

Parameters

`userRoutine`
A pointer to your application-defined function.

Return Value
A new UPP; see `Universal Procedure Pointers`.

Discussion
This function is used with Macintosh PowerPC systems. See *Inside Macintosh: PowerPC System Software*.

Version Notes
Introduced in QuickTime 4.1. Replaces `NewQDPixProc`.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`ImageCompression.h`

NewStdPixUPP

Allocates a Universal Procedure Pointer for the StdPixProc callback.

```
StdPixUPP NewStdPixUPP (
   StdPixProcPtr userRoutine
);
```

Parameters

`userRoutine`
A pointer to your application-defined function.

Return Value
A new UPP; see `Universal Procedure Pointers`.

Discussion
This function is used with Macintosh PowerPC systems. See *Inside Macintosh: PowerPC System Software*.

Version Notes
Introduced in QuickTime 4.1. Replaces `NewStdPixProc`.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
Desktop Sprites
DesktopSprites
DesktopSprites.win

Declared In
`ImageCompression.h`

QTAddComponentPropertyListener

Installs a callback to monitor a component property.

```
ComponentResult QTAddComponentPropertyListener (
   ComponentInstance inComponent,
   ComponentPropertyClass inPropClass,
   ComponentPropertyID inPropID,
   QTComponentPropertyListenerUPP inDispatchProc,
   void *inUserData
);
```

Parameters

inComponent

A component instance, which you can get by calling `OpenComponent` or `OpenDefaultComponent`.

inPropClass

A value (see below) of type `OSType` that specifies a property class: `kComponentPropertyClassPropertyInfo` (`'pnfo'`) A `QTComponentPropertyInfo` structure that defines a property information class. `kComponentPropertyInfoList` (`'list'`) An array of `QTComponentPropertyInfo` structures, one for each property. `kComponentPropertyCacheSeed` (`'seed'`) A component property cache seed value. `kComponentPropertyExtendedInfo` (`'meta'`) A `CFDictionary` with extended property information. `kComponentPropertyCacheFlags` (`'flgs'`) One of the following two flags: `kComponentPropertyCacheFlagNotPersistent` Property metadata should not be saved in persistent cache. `kComponentPropertyCacheFlagIsDynamic` Property metadata should not be cached at all. See these constants:

```
kComponentPropertyClassPropertyInfo
kComponentPropertyInfoList
kComponentPropertyCacheSeed
kComponentPropertyExtendedInfo
kComponentPropertyCacheFlags
kComponentPropertyCacheFlagNotPersistent
kComponentPropertyCacheFlagIsDynamic
```

inPropID

A value of type `OSType` that specifies a property ID.

inDispatchProc

A Universal Procedure Pointer to a `QTComponentPropertyListenerProc` callback.

inUserData

A pointer to user data that will be passed to the callback. You may pass NULL in this parameter.

Return Value

See `Error Codes` in the QuickTime API Reference. Returns `noErr` if there is no error.

Version Notes

Introduced in QuickTime 6.4.

Availability

Available in Mac OS X v10.3 and later.

Related Sample Code

WhackedTV

Declared In

`ImageCompression.h`

QTComponentPropertyListenerCollectionAddListener

Adds a listener callback for a specified property class and ID to a property listener collection.

```
OSStatus QTComponentPropertyListenerCollectionAddListener (
   QTComponentPropertyListenersRef inCollection,
   ComponentPropertyClass inPropClass,
   ComponentPropertyID inPropID,
   QTComponentPropertyListenerUPP inListenerProc,
   const void *inListenerProcRefCon
);
```

Parameters

inCollection

A property listener collection created by a previous call to `QTComponentPropertyListenerCollectionCreate`.

inPropClass

A value (see below) of type `OSType` that specifies a property class: `kComponentPropertyClassPropertyInfo` (`'pnfo'`) A `QTComponentPropertyInfo` structure that defines a property information class. `kComponentPropertyInfoList` (`'list'`) An array of `QTComponentPropertyInfo` structures, one for each property. `kComponentPropertyCacheSeed` (`'seed'`) A component property cache seed value. `kComponentPropertyExtendedInfo` (`'meta'`) A `CFDictionary` with extended property information. `kComponentPropertyCacheFlags` (`'flgs'`) One of the following two flags: `kComponentPropertyCacheFlagNotPersistent` Property metadata should not be saved in persistent cache. `kComponentPropertyCacheFlagIsDynamic` Property metadata should not be cached at all. See these constants:

```
kComponentPropertyClassPropertyInfo
kComponentPropertyInfoList
kComponentPropertyCacheSeed
kComponentPropertyExtendedInfo
kComponentPropertyCacheFlags
kComponentPropertyCacheFlagNotPersistent
kComponentPropertyCacheFlagIsDynamic
```

inPropID

A value of type `OSType` that specifies a property ID.

inListenerProc

A `QTComponentPropertyListenerProc` callback.

inListenerProcRefCon

A reference constant to be passed to your callback. Use this parameter to point to a data structure containing any information your function needs.

Return Value

See `Error Codes` in the QuickTime API Reference. Returns `noErr` if there is no error.

Version Notes

Introduced in QuickTime 6.4.

Availability

Available in Mac OS X v10.3 and later.

Declared In

`ImageCompression.h`

QTComponentPropertyListenerCollectionCreate

Creates a collection of component property monitors.

```
OSStatus QTComponentPropertyListenerCollectionCreate (
   CFAllocatorRef inAllocator,
   const QTComponentPropertyListenerCollectionContext *inContext,
   QTComponentPropertyListenersRef *outCollection
);
```

Parameters

inAllocator

A pointer to the allocator used to create the collection and its contents. You can pass `NIL`.

inContext

A pointer to a `QTComponentPropertyInfo` data structure. You can pass `NIL` if no structure exists. A copy of the contents of the structure is made; therefore you can pass a pointer to a structure on the stack.

outCollection

On return, a pointer to the new empty listener collection.

Return Value

See `Error Codes` in the QuickTime API Reference. Returns `noErr` if there is no error.

Version Notes

Introduced in QuickTime 6.4.

Availability

Available in Mac OS X v10.3 and later.

Declared In

`ImageCompression.h`

QTComponentPropertyListenerCollectionHasListenersForProperty

Determines if there are any listeners in a component property listener collection registered for a specified property class and ID.

```
Boolean QTComponentPropertyListenerCollectionHasListenersForProperty (
   QTComponentPropertyListenersRef inCollection,
   ComponentPropertyClass inPropClass,
   ComponentPropertyID inPropID
);
```

Parameters

inCollection

A property listener collection created by a previous call to `QTComponentPropertyListenerCollectionCreate`.

inPropClass

A value (see below) of type `OSType` that specifies a property class: `kComponentPropertyClassPropertyInfo` (`'pnfo'`) A `QTComponentPropertyInfo` structure that defines a property information class. `kComponentPropertyInfoList` (`'list'`) An array of `QTComponentPropertyInfo` structures, one for each property. `kComponentPropertyCacheSeed` (`'seed'`) A component property cache seed value. `kComponentPropertyExtendedInfo` (`'meta'`) A `CFDictionary` with extended property information. `kComponentPropertyCacheFlags` (`'flgs'`) One of the following two flags: `kComponentPropertyCacheFlagNotPersistent` Property metadata should not be saved in persistent cache. `kComponentPropertyCacheFlagIsDynamic` Property metadata should not be cached at all. See these constants:

```
kComponentPropertyClassPropertyInfo
kComponentPropertyInfoList
kComponentPropertyCacheSeed
kComponentPropertyExtendedInfo
kComponentPropertyCacheFlags
kComponentPropertyCacheFlagNotPersistent
kComponentPropertyCacheFlagIsDynamic
```

inPropID

A value of type `OSType` that specifies a property ID.

Return Value
Returns TRUE if there are any listeners in the listener collection registered for the specified property class and ID, FALSE otherwise.

Version Notes
Introduced in QuickTime 6.4.

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

Declared In
`ImageCompression.h`

QTComponentPropertyListenerCollectionIsEmpty

Determines if a listener collection is empty.

```
Boolean QTComponentPropertyListenerCollectionIsEmpty (
   QTComponentPropertyListenersRef inCollection
);
```

Parameters

inCollection

A property listener collection created by a previous call to `QTComponentPropertyListenerCollectionCreate`.

Return Value
Returns TRUE if the collection is empty, FALSE otherwise.

Version Notes
Introduced in QuickTime 6.4.

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

Declared In
`ImageCompression.h`

QTComponentPropertyListenerCollectionNotifyListeners

Calls all listener callbacks in a component property listener collection registered for a specified property class and ID.

```
OSStatus QTComponentPropertyListenerCollectionNotifyListeners (
   QTComponentPropertyListenersRef inCollection,
   ComponentInstance inNotifier,
   ComponentPropertyClass inPropClass,
   ComponentPropertyID inPropID,
   const void *inFilterProcRefCon,
   UInt32 inFlags
);
```

Parameters

`inCollection`

A property listener collection created by a previous call to `QTComponentPropertyListenerCollectionCreate`.

`inNotifier`

The caller's component instance.

`inPropClass`

A value (see below) of type `OSType` that specifies a property class: `kComponentPropertyClassPropertyInfo` (`'pnfo'`) A `QTComponentPropertyInfo` structure that defines a property information class. `kComponentPropertyInfoList` (`'list'`) An array of `QTComponentPropertyInfo` structures, one for each property. `kComponentPropertyCacheSeed` (`'seed'`) A component property cache seed value. `kComponentPropertyExtendedInfo` (`'meta'`) A `CFDictionary` with extended property information. `kComponentPropertyCacheFlags` (`'flgs'`) One of the following two flags: `kComponentPropertyCacheFlagNotPersistent` Property metadata should not be saved in persistent cache. `kComponentPropertyCacheFlagIsDynamic` Property metadata should not be cached at all. See these constants:

```
kComponentPropertyClassPropertyInfo
kComponentPropertyInfoList
kComponentPropertyCacheSeed
kComponentPropertyExtendedInfo
kComponentPropertyCacheFlags
kComponentPropertyCacheFlagNotPersistent
kComponentPropertyCacheFlagIsDynamic
```

`inPropID`

A value of type `OSType` that specifies a property ID.

`inFilterProcRefCon`

A reference constant to be passed to your callback. Use this parameter to point to a data structure containing any information your function needs. You may pass `NIL`.

inFlags

Currently not used.

Return Value
See `Error Codes` in the QuickTime API Reference. Returns `noErr` if there is no error.

Discussion
If the `filterProcUPP` field in the `QTComponentPropertyListenerCollectionContext` data structure that was passed to `QTComponentPropertyListenerCollectionCreate` is not `NIL`, the `QTComponentPropertyListenerFilterProc` callback it points to will be called before each call to a registered listener that matches the specified property class and ID passed to this function. If the filter function return FALSE, that listener callback will not be called. This lets a component change the calling semantics (for example, to call another thread) or use a different listener callback signature.

Version Notes
Introduced in QuickTime 6.4.

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

Declared In
`ImageCompression.h`

QTComponentPropertyListenerCollectionRemoveListener

Removes a listener callback with a specified property class and ID from a property listener collection.

```
OSStatus QTComponentPropertyListenerCollectionRemoveListener (
   QTComponentPropertyListenersRef inCollection,
   ComponentPropertyClass inPropClass,
   ComponentPropertyID inPropID,
   QTComponentPropertyListenerUPP inListenerProc,
   const void *inListenerProcRefCon
);
```

Parameters

inCollection

A property listener collection created by a previous call to `QTComponentPropertyListenerCollectionCreate`.

inPropClass

A value (see below) of type `OSType` that specifies a property class: `kComponentPropertyClassPropertyInfo` (`'pnfo'`) A `QTComponentPropertyInfo` structure that defines a property information class. `kComponentPropertyInfoList` (`'list'`) An array of `QTComponentPropertyInfo` structures, one for each property. `kComponentPropertyCacheSeed` (`'seed'`) A component property cache seed value. `kComponentPropertyExtendedInfo` (`'meta'`) A `CFDictionary` with extended property information. `kComponentPropertyCacheFlags` (`'flgs'`) One of the following two flags: `kComponentPropertyCacheFlagNotPersistent` Property metadata should not be saved in persistent cache. `kComponentPropertyCacheFlagIsDynamic` Property metadata should not be cached at all. See these constants:

```
kComponentPropertyClassPropertyInfo
kComponentPropertyInfoList
kComponentPropertyCacheSeed
kComponentPropertyExtendedInfo
kComponentPropertyCacheFlags
kComponentPropertyCacheFlagNotPersistent
kComponentPropertyCacheFlagIsDynamic
```

inPropID

A value of type `OSType` that specifies a property ID.

inListenerProc

The `QTComponentPropertyListenerProc` callback to be removed.

inListenerProcRefCon

A reference constant to be passed to your callback. Use this parameter to point to a data structure containing any information your function needs.

Return Value
See `Error Codes` in the QuickTime API Reference. Returns `noErr` if there is no error.

Version Notes
Introduced in QuickTime 6.4.

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

Declared In
`ImageCompression.h`

QTGetComponentProperty

Returns the value of a specific component property.

```
ComponentResult QTGetComponentProperty (
   ComponentInstance inComponent,
   ComponentPropertyClass inPropClass,
   ComponentPropertyID inPropID,
   ByteCount inPropValueSize,
   ComponentValuePtr outPropValueAddress,
   ByteCount *outPropValueSizeUsed
);
```

Parameters

inComponent

A component instance, which you can get by calling `OpenComponent` or `OpenDefaultComponent`.

inPropClass

A value (see below) of type `OSType` that specifies a property class: `kComponentPropertyClassPropertyInfo` (`'pnfo'`) A `QTComponentPropertyInfo` structure that defines a property information class. `kComponentPropertyInfoList` (`'list'`) An array of `QTComponentPropertyInfo` structures, one for each property. `kComponentPropertyCacheSeed` (`'seed'`) A component property cache seed value. `kComponentPropertyExtendedInfo` (`'meta'`) A `CFDictionary` with extended property information. `kComponentPropertyCacheFlags` (`'flgs'`) One of the following two flags: `kComponentPropertyCacheFlagNotPersistent` Property metadata should not be saved in persistent cache. `kComponentPropertyCacheFlagIsDynamic` Property metadata should not be cached at all. See these constants:

`kComponentPropertyClassPropertyInfo`
`kComponentPropertyInfoList`
`kComponentPropertyCacheSeed`
`kComponentPropertyExtendedInfo`
`kComponentPropertyCacheFlags`
`kComponentPropertyCacheFlagNotPersistent`
`kComponentPropertyCacheFlagIsDynamic`

inPropID

A value of type `OSType` that specifies a property ID.

inPropValueSize

The size of the buffer allocated to hold the property value.

outPropValueAddress

A pointer to the buffer allocated to hold the property value.

outPropValueSizeUsed

On return, the actual size of the value written to the buffer.

Return Value

See `Error Codes` in the QuickTime API Reference. Returns `noErr` if there is no error.

Version Notes

Introduced in QuickTime 6.4.

Availability

Available in Mac OS X v10.3 and later.

Related Sample Code

CaptureAndCompressIPBMovie
QTExtractAndConvertToAIFF
QTExtractAndConvertToMovieFile

SCAudioCompress
WhackedTV

Declared In
`ImageCompression.h`

QTGetComponentPropertyInfo

Returns information about the properties of a component.

```
ComponentResult QTGetComponentPropertyInfo (
   ComponentInstance inComponent,
   ComponentPropertyClass inPropClass,
   ComponentPropertyID inPropID,
   ComponentValueType *outPropType,
   ByteCount *outPropValueSize,
   UInt32 *outPropertyFlags
);
```

Parameters

inComponent

A component instance, which you can get by calling `OpenComponent` or `OpenDefaultComponent`.

inPropClass

A value (see below) of type `OSType` that specifies a property class: `kComponentPropertyClassPropertyInfo` (`'pnfo'`) A `QTComponentPropertyInfo` structure that defines a property information class. `kComponentPropertyInfoList` (`'list'`) An array of `QTComponentPropertyInfo` structures, one for each property. `kComponentPropertyCacheSeed` (`'seed'`) A component property cache seed value. `kComponentPropertyExtendedInfo` (`'meta'`) A `CFDictionary` with extended property information. `kComponentPropertyCacheFlags` (`'flgs'`) One of the following two flags: `kComponentPropertyCacheFlagNotPersistent` Property metadata should not be saved in persistent cache. `kComponentPropertyCacheFlagIsDynamic` Property metadata should not be cached at all. See these constants:

- `kComponentPropertyClassPropertyInfo`
- `kComponentPropertyInfoList`
- `kComponentPropertyCacheSeed`
- `kComponentPropertyExtendedInfo`
- `kComponentPropertyCacheFlags`
- `kComponentPropertyCacheFlagNotPersistent`
- `kComponentPropertyCacheFlagIsDynamic`

inPropID

A value of type `OSType` that specifies a property ID.

outPropType

A pointer to memory allocated to hold the `property` type on return. This pointer may be NULL.

outPropValueSize

A pointer to memory allocated to hold the size of the property value on return. This pointer may be NULL.

outPropertyFlags

A pointer to memory allocated to hold property flags on return.

Return Value
See `Error Codes` in the QuickTime API Reference. Returns `noErr` if there is no error.

Version Notes
Introduced in QuickTime 6.4.

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

Related Sample Code
ElectricImageComponent
QTExtractAndConvertToAIFF
QTExtractAndConvertToMovieFile
SCAudioCompress
WhackedTV

Declared In
`ImageCompression.h`

QTOpenGLTextureContextCreate

Creates a new OpenGL texture context for a specified OpenGL context and pixel format.

```
OSStatus QTOpenGLTextureContextCreate (
   CFAllocatorRef allocator,
   CGLContextObj cglContext,
   CGLPixelFormatObj cglPixelFormat,
   CFDictionaryRef attributes,
   QTVisualContextRef *newTextureContext
);
```

Parameters

`allocator`
The allocator used to create the texture context.

`cglContext`
A pointer to an opaque `CGLPContextObj` structure representing the OpenGL context used to create textures. You can create this structure using `CGLCreateContext`.

`cglPixelFormat`
The pixel format object that specifies buffer types and other attributes of the new context.

`attributes`
A dictionary of attributes.

`newTextureContext`
A pointer to a variable to receive the new OpenGL texture context.

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

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

Related Sample Code
LiveVideoMixer3

QTCoreImage101
QTCoreVideo102
QTCoreVideo201
QTCoreVideo301

Declared In
`ImageCompression.h`

QTPixelBufferContextCreate

Creates a new pixel buffer context with the given attributes.

```
OSStatus QTPixelBufferContextCreate (
   CFAllocatorRef allocator,
   CFDictionaryRef attributes,
   QTVisualContextRef *newPixelBufferContext
);
```

Parameters

`allocator`
Allocator used to create the pixel buffer context.

`attributes`
Dictionary of attributes.

`newPixelBufferContext`
Points to a variable to receive the new pixel buffer context.

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

Discussion
This routine creates a new pixel buffer context with the given attributes.

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

Related Sample Code
QTCoreVideo102
QTCoreVideo201
QTCoreVideo202
QTCoreVideo301
QTPixelBufferVCToCGImage

Declared In
`ImageCompression.h`

QTRemoveComponentPropertyListener

Removes a component property monitoring callback.

```
ComponentResult QTRemoveComponentPropertyListener (
   ComponentInstance inComponent,
   ComponentPropertyClass inPropClass,
   ComponentPropertyID inPropID,
   QTComponentPropertyListenerUPP inDispatchProc,
   void *inUserData
);
```

Parameters

inComponent

A component instance, which you can get by calling `OpenComponent` or `OpenDefaultComponent`.

inPropClass

A value (see below) of type `OSType` that specifies a property class: `kComponentPropertyClassPropertyInfo` (`'pnfo'`) A `QTComponentPropertyInfo` structure that defines a property information class. `kComponentPropertyInfoList` (`'list'`) An array of `QTComponentPropertyInfo` structures, one for each property. `kComponentPropertyCacheSeed` (`'seed'`) A component property cache seed value. `kComponentPropertyExtendedInfo` (`'meta'`) A `CFDictionary` with extended property information. `kComponentPropertyCacheFlags` (`'flgs'`) One of the following two flags: `kComponentPropertyCacheFlagNotPersistent` Property metadata should not be saved in persistent cache. `kComponentPropertyCacheFlagIsDynamic` Property metadata should not be cached at all. See these constants:

```
kComponentPropertyClassPropertyInfo
kComponentPropertyInfoList
kComponentPropertyCacheSeed
kComponentPropertyExtendedInfo
kComponentPropertyCacheFlags
kComponentPropertyCacheFlagNotPersistent
kComponentPropertyCacheFlagIsDynamic
```

inPropID

A value of type `OSType` that specifies a property ID.

inDispatchProc

A Universal Procedure Pointer to a `QTComponentPropertyListenerProc` callback.

inUserData

User data to be passed to the callback.

Return Value

See `Error Codes` in the QuickTime API Reference. Returns `noErr` if there is no error.

Version Notes

Introduced in QuickTime 6.4.

Availability

Available in Mac OS X v10.3 and later.

Declared In

`ImageCompression.h`

QTSetComponentProperty

Sets the value of a specific component property.

```
ComponentResult QTSetComponentProperty (
   ComponentInstance inComponent,
   ComponentPropertyClass inPropClass,
   ComponentPropertyID inPropID,
   ByteCount inPropValueSize,
   ConstComponentValuePtr inPropValueAddress
);
```

Parameters

inComponent

A component instance, which you can get by calling `OpenComponent` or `OpenDefaultComponent`.

inPropClass

A value of type `OSType` that specifies a property class: `kComponentPropertyClassPropertyInfo` (`'pnfo'`) A `QTComponentPropertyInfo` structure that defines a property information class. `kComponentPropertyInfoList` (`'list'`) An array of `QTComponentPropertyInfo` structures, one for each property. `kComponentPropertyCacheSeed` (`'seed'`) A component property cache seed value. `kComponentPropertyExtendedInfo` (`'meta'`) A `CFDictionary` with extended property information. `kComponentPropertyCacheFlags` (`'flgs'`) One of the following two flags: `kComponentPropertyCacheFlagNotPersistent` Property metadata should not be saved in persistent cache. `kComponentPropertyCacheFlagIsDynamic` Property metadata should not be cached at all. See these constants:

`kComponentPropertyClassPropertyInfo`
`kComponentPropertyInfoList`
`kComponentPropertyCacheSeed`
`kComponentPropertyExtendedInfo`
`kComponentPropertyCacheFlags`
`kComponentPropertyCacheFlagNotPersistent`
`kComponentPropertyCacheFlagIsDynamic`

inPropID

A value of type `OSType` that specifies a property ID.

inPropValueSize

The size of the buffer allocated to hold the property value.

outPropValueAddress

A pointer to the buffer allocated to hold the property value.

Return Value

See `Error Codes` in the QuickTime API Reference. Returns `noErr` if there is no error.

Version Notes

Introduced in QuickTime 6.4.

Availability

Available in Mac OS X v10.3 and later.

Related Sample Code

CaptureAndCompressIPBMovie
QTExtractAndConvertToAIFF
QTExtractAndConvertToMovieFile
SCAudioCompress
WhackedTV

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.

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

Related Sample Code
LiveVideoMixer3
QTCoreImage101
QTCoreVideo102
QTCoreVideo201
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.

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

Declared In
`ImageCompression.h`

QTVisualContextGetTypeID

Returns the CFTypeID for QTVisualContextRef.

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

Return Value
Undocumented.

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

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

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` (page 121).

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

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.

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

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.

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

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.

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

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.

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

Related Sample Code
QTPixelBufferVCToCGImage

Declared In
`ImageCompression.h`

QTVisualContextTask

Causes visual context to release internally held resources for later re-use.

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

Parameters

visualContext

The visual context.

Discussion

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

Availability

Available in Mac OS X v10.3 and later.

Related Sample Code

LiveVideoMixer3

QTCoreImage101

QTCoreVideo102

QTCoreVideo201

QTCoreVideo301

Declared In

`ImageCompression.h`

Callbacks

ICMAlignmentProc

Provides an alignment behavior for windows based on the screen's bit depth.

```
typedef void (*ICMAlignmentProcPtr) (Rect *rp, long refcon);
```

If you name your function `MyICMAlignmentProc`, you would declare it this way:

```
void MyICMAlignmentProc (
    Rect    *rp,
    long    refcon );
```

Parameters

rp

Contains a pointer to a rectangle that has already been aligned with a default alignment function.

refcon

Contains a reference constant value for use by your alignment function. Your application specifies the value of this reference constant in the alignment function structure you pass to the Image Compression Manager.

Declared In

`ImageCompression.h`

ICMCompletionProc

Called by a compressor component upon completion of an asynchronous operation.

```
typedef void (*ICMCompletionProcPtr) (OSErr result, short flags, long refcon);
```

If you name your function `MyICMCompletionProc`, you would declare it this way:

```
void MyICMCompletionProc (
    OSErr    result,
    short    flags,
    long     refcon );
```

Parameters

result

Indicator of success of current operation.

flags

Contains flags (see below) that indicate which part of the operation is complete. Note that more than one of the flags may be set to 1. See these constants:

`codecCompletionSource`

`codecCompletionDest`

refcon

Contains a reference constant value for use by your completion function. Your application specifies the value of this reference constant in the callback function structure you pass to the Image Compression Manager.

Declared In

`ImageCompression.h`

ICMCursorShieldedProc

Undocumented

```
typedef void (*ICMCursorShieldedProcPtr) (const Rect *r, void *refcon, long flags);
```

If you name your function `MyICMCursorShieldedProc`, you would declare it this way:

```
void MyICMCursorShieldedProc (
    const Rect    *r,
    void          *refcon,
    long          flags );
```

Parameters

r

Undocumented

refcon

Pointer to a reference constant that the client code supplies to your callback. You can use this reference to point to a data structure containing any information your callback needs.

flags

Undocumented

Declared In
ImageCompression.h

ICMDataProc

Supplies compressed data during a decompression operation.

```
typedef OSErr (*ICMDataProcPtr) (Ptr *dataP, long bytesNeeded, long refcon);
```

If you name your function MyICMDataProc, you would declare it this way:

```
OSErr MyICMDataProc (
    Ptr     *dataP,
    long    bytesNeeded,
    long    refcon );
```

Parameters

dataP

Contains a pointer to the address of the data buffer. The decompressor uses this parameter to indicate where your data-loading function should return the compressed data. You establish this data buffer when you start the decompression operation. For example, the data parameter to FDecompressImage defines the location of the data buffer for that operation. Upon return from your data-loading function, this pointer should refer to the beginning of the compressed data that you loaded. The decompressor may also use this parameter to indicate that it wants to reset the mark within the compressed data stream. If the dataP parameter is set to NIL, the bytesNeeded parameter contains the new mark position, relative to the current position of the data stream. If your data-loading function does not support this operation, return a nonzero result code.

bytesNeeded

Specifies the number of bytes requested or the new mark offset. If the decompressor has requested additional compressed data (that is, the value of the dataP parameter is not NIL), then this parameter specifies how many bytes to return. This value never exceeds the size of the original data buffer. Your data-loading function should read the data from the current mark in the input data stream. If the decompressor has requested to set a new mark position in the data stream (that is, the value of the dataP parameter is NIL), then this parameter specifies the new mark position relative to the current position of the data stream.

refcon

Contains a reference constant value for use by your data-loading function. Your application specifies the value of this reference constant in the data-loading function structure you pass to the Image Compression Manager.

Return Value
See Error Codes. Your callback should return noErr if there is no error.

Declared In
ImageCompression.h

ICMFlushProc

Writes compressed data to a storage device during a compression operation.

```
typedef OSErr (*ICMFlushProcPtr) (Ptr data, long bytesAdded, long refcon);
```

If you name your function `MyICMFlushProc`, you would declare it this way:

```
OSErr MyICMFlushProc (
    Ptr     data,
    long    bytesAdded,
    long    refcon );
```

Parameters

data

Points to the data buffer. The compressor uses this parameter to indicate where your data-unloading function can find the compressed data. You establish this data buffer when you start the compression operation. For example, the `data` parameter to `FCompressImage` defines the location of the data buffer for that operation. This pointer contains a 32-bit clean address. Your `ICMFlushProc` function should make no other assumptions about the value of this address. The compressor may also use this parameter to indicate that it wants to reset the mark within the compressed data stream. If the `data` parameter is set to `NIL`, the `bytesNeeded` parameter contains the new mark position, relative to the current position of the output data stream. If your `ICMFlushProc` function does not support this operation, return a nonzero result code.

bytesAdded

Specifies the number of bytes to write or the new mark offset. If the compressor wants to write out some compressed data (that is, the value of data is not `NIL`), then this parameter specifies how many bytes to write. This value never exceeds the size of the original data buffer. Your `ICMFlushProc` function should write that data at the current mark in the output data stream. If the compressor has requested to set a new mark position in the output data stream (that is, the value of data is `NIL`), then this parameter specifies the new mark position relative to the current position of the data stream.

refcon

Contains a reference constant value for use by your `ICMFlushProc` function. Your application specifies the value of this reference constant in the data-unloading function structure you pass to the Image Compression Manager.

Return Value

See `Error Codes`. Your callback should return `noErr` if there is no error.

Discussion

You assign an `ICMFlushProc` function to an image or a sequence by passing a pointer to a structure that identifies the function to the appropriate compression function.

Declared In

`ImageCompression.h`

ICMProgressProc

Reports on the progress of a compressor or decompressor.

```
typedef OSErr (*ICMProgressProcPtr) (short message, Fixed completeness, long refcon);
```

If you name your function `MyICMProgressProc`, you would declare it this way:

```
OSErr MyICMProgressProc (
    short    message,
    Fixed    completeness,
```

```
    long     refcon );
```

Parameters

message

Indicates why the Image Compression Manager called your function. There are three valid messages, listed below. See these constants:

```
codecProgressOpen
codecProgressUpdatePercent
codecProgressClose
```

completeness

Contains a fixed-point value indicating how far the operation has progressed. Its value is always between 0.0 and 1.0. This parameter is valid only when the `message` field is set to `codecProgressUpdatePercent`.

refcon

Contains a reference constant value for use by your progress function. Your application specifies the value of this reference constant in the progress function structure you pass to the Image Compression Manager.

Return Value

See `Error Codes`. Your callback should return `noErr` if there is no error. When a component calls your progress function, it supplies you with a number that indicates the completion percentage. Your program can cause the component to terminate the current operation by returning a result code of `codecAbortErr`.

Discussion

The Image Compression Manager calls your progress function only during long operations, and it does not call your function more than 30 times per second.

Declared In

`ImageCompression.h`

QDPixProc

Undocumented

```
typedef void (*QDPixProcPtr) (PixMap *src, Rect *srcRect, MatrixRecord *matrix,
short mode, RgnHandle mask, PixMap *matte, Rect *matteRect,
short flags);
```

If you name your function `MyQDPixProc`, you would declare it this way:

```
void MyQDPixProc (
    PixMap          *src,
    Rect            *srcRect,
    MatrixRecord    *matrix,
    short           mode,
    RgnHandle       mask,
    PixMap          *matte,
    Rect            *matteRect,
    short           flags );
```

Parameters

src

Undocumented

srcRect

Undocumented

matrix

Undocumented

mode

Undocumented

mask

Undocumented

matte

Undocumented

matteRect

Undocumented

flags

Undocumented

Declared In

`ImageCompression.h`

StdPixProc

Undocumented

```
typedef void (*StdPixProcPtr) (PixMap *src, Rect *srcRect, MatrixRecord *matrix,
short mode, RgnHandle mask, PixMap *matte, Rect *matteRect,
short flags);
```

If you name your function `MyStdPixProc`, you would declare it this way:

```
void MyStdPixProc (
    PixMap          *src,
    Rect            *srcRect,
    MatrixRecord    *matrix,
    short           mode,
    RgnHandle       mask,
    PixMap          *matte,
    Rect            *matteRect,
    short           flags );
```

Parameters

src

Undocumented

srcRect

Undocumented

matrix

Undocumented

mode
Undocumented

mask
Undocumented

matte
Undocumented

matteRect
Undocumented

flags
Undocumented

Declared In
ImageCompression.h

Data Types

ICMAlignmentUPP

Represents a type used by the Image Compression API.

```
typedef STACK_UPP_TYPE(ICMAlignmentProcPtr) ICMAlignmentUPP;
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
ImageCompression.h

ICMCompletionUPP

Represents a type used by the Image Compression API.

```
typedef STACK_UPP_TYPE(ICMCompletionProcPtr) ICMCompletionUPP;
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
ImageCompression.h

ICMCursorShieldedUPP

Represents a type used by the Image Compression API.

```
typedef STACK_UPP_TYPE(ICMCursorShieldedProcPtr) ICMCursorShieldedUPP;
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
ImageCompression.h

ICMDataUPP

Represents a type used by the Image Compression API.

```
typedef STACK_UPP_TYPE(ICMDataProcPtr) ICMDataUPP;
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
ImageCompression.h

ICMDecompressionTrackingCallbackRecord

Designates a tracking callback for an ICM decompression session.

```
struct ICMDecompressionTrackingCallbackRecord {
ICMDecompressionTrackingCallback    decompressionTrackingCallback;
void                                *decompressionTrackingRefCon;
};
```

Fields
decompressionTrackingCallback

Discussion
The callback function pointer. See ICMDecompressionTrackingCallbackProc.

decompressionTrackingRefCon

Discussion
The callback's reference value.

Declared In
ImageCompression.h

ICMFlushUPP

Represents a type used by the Image Compression API.

```
typedef STACK_UPP_TYPE(ICMFlushProcPtr) ICMFlushUPP;
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
ImageCompression.h

ICMMultiPassStorageCallbacks

Designates a collection of callbacks for creating a custom multipass storage object.

```
struct ICMMultiPassStorageCallbacks {
UInt32                                      version;
void                                        *storageRefCon;
ICMMultiPassSetDataAtTimeStampCallback      setDataAtTimeStampCallback;
ICMMultiPassGetTimeStampCallback            getTimeStampCallback;
ICMMultiPassCopyDataAtTimeStampCallback     copyDataAtTimeStampCallback;
ICMMultiPassReleaseCallback                 releaseCallback;
};
```

Fields

`version`

Discussion

The version of this structure. Set to `kICMMultiPassStorageCallbacksVersionOne`.

`storageRefCon`

Discussion

A pointer to a reference constant. Use this parameter to point to a data structure containing any information your callback needs.

`setDataAtTimeStampCallback`

Discussion

A callback for storing values.

`getTimeStampCallback`

Discussion

A callback for finding time stamps.

`copyDataAtTimeStampCallback`

Discussion

A callback for retrieving values.

`releaseCallback`

Discussion

A callback for disposing the callback's state when done.

Discussion

This structure is used by `ICMMultiPassStorageCreateWithCallbacks`.

Declared In

`ImageCompression.h`

ICMProgressUPP

Represents a type used by the Image Compression API.

```
typedef STACK_UPP_TYPE(ICMProgressProcPtr) ICMProgressUPP;
```

Availability

Available in Mac OS X v10.0 and later.

Declared In

`ImageCompression.h`

ImageTranscoderComponent

Represents a type used by the Image Compression API.

```
typedef ComponentInstance ImageTranscoderComponent;
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
`ImageCompression.h`

QDPixUPP

Represents a type used by the Image Compression API.

```
typedef STACK_UPP_TYPE(QDPixProcPtr) QDPixUPP;
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
`ImageCompression.h`

QTComponentPropertyListenerCollectionContext

Provides context information for a QTComponentPropertyListenerFilterProc callback.

```
struct QTComponentPropertyListenerCollectionContext
{
    UInt32                                  version;
    QTComponentPropertyListenerFilterUPP    filterProcUPP;
    void                                    *filterProcData;
};
```

Fields
`version`

Discussion
The version of this callback.

`filterProcUPP`

Discussion
A Universal Procedure Pointer to a `QTComponentPropertyListenerFilterProc` callback.

`filterProcData`

Discussion
A pointer to data for the callback.

Version Notes
Introduced in QuickTime 6.4.

Related Functions
Associated function:
`QTComponentPropertyListenerCollectionNotifyListeners` (page 112)

Declared In
`ImageCompression.h`

StdPixUPP

Represents a type used by the Image Compression API.

```
typedef STACK_UPP_TYPE(StdPixProcPtr) StdPixUPP;
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
`ImageCompression.h`

Constants

ICMProgressProc Values

Constants passed to ICMProgressProc.

```
enum {
  codecProgressOpen             = 0,
  codecProgressUpdatePercent    = 1,
  codecProgressClose            = 2
};
```

Constants

`codecProgressOpen`

Indicates the start of a long operation. This is always the first message sent to your function. Your function can use this message to trigger the display of your progress window.

Available in Mac OS X v10.0 and later.

Declared in `ImageCompression.h`.

`codecProgressUpdatePercent`

Passes completion information to your function. The Image Compression Manager repeatedly sends this message to your function. The completeness parameter indicates the relative completion of the operation. You can use this value to update your progress window.

Available in Mac OS X v10.0 and later.

Declared in `ImageCompression.h`.

Declared In
`ImageCompression.h`

ICM Property IDs

Constants that contain the IDs of ICM properties.

```
enum {
  /*
   * Both fields should be decompressed.
   */
  kICMFieldMode_BothFields      = 0,
  /*
   * Only the top field should be decompressed, producing a half-height
   * image.
   */
  kICMFieldMode_TopFieldOnly    = 1,
  /*
   * Only the bottom field should be decompressed, producing a
   * half-height image.
   */
  kICMFieldMode_BottomFieldOnly = 2,
  /*
   * Both fields should be decompressed, and then filtered to reduce
   * interlacing artifacts.
   */
  kICMFieldMode_DeinterlaceFields = 3
};
enum {
  /*
   * Class identifier for compression frame options object properties.
   */
  kQTPropertyClass_ICMCompressionFrameOptions = 'icfo',
  /*
   * Forces frames to be compressed as key frames.
   * The compressor must obey the "force key frame" flag if set. By
   * default this property is false.
   */
  kICMCompressionFrameOptionsPropertyID_ForceKeyFrame = 'keyf', /* Boolean,
Read/Write */
  /*
   * Requests a frame be compressed as a particular frame type.
   *  The frame type setting may be ignored by the compressor if not
   * appropriate.
   * By default this is set to kICMFrameType_Unknown.
   * Do not assume that kICMFrameType_I means a key frame; if you need
   * a key frame, set the "force key frame" property.
   */
  kICMCompressionFrameOptionsPropertyID_FrameType = 'frty' /* ICMFrameType,
Read/Write */
};
enum {
  /*
   * Class identifier for compression session options object properties.
   */
  kQTPropertyClass_ICMCompressionSessionOptions = 'icso',
  /*
   * Enables temporal compression. By default, temporal compression is
   * disabled.
   * IMPORTANT: If you want temporal compression (P frames and/or B
   * frames) you must set this to true.
   */
  kICMCompressionSessionOptionsPropertyID_AllowTemporalCompression = 'p ok', /*
Boolean, Read/Write */
  /*
```

```
    * Enables frame reordering.
    * In order to encode B frames, a compressor must reorder frames,
    * which means that the order in which they will be emitted and
    * stored (the decode order) is different from the order in which
    * they were presented to the compressor (the display order).
    * By default, frame reordering is disabled.
    * IMPORTANT: In order to encode using B frames, you must enable
    * frame reordering.
    */
   kICMCompressionSessionOptionsPropertyID_AllowFrameReordering = 'b ok', /* Boolean,
  Read/Write */
   /*
    * Indicates that durations of emitted frames are needed.
    * If this flag is set and source frames are provided with times but
    * not durations, then frames will be delayed so that durations can
    * be calculated as the difference between one frame's time stamp and
    * the next frame's time stamp.
    * By default, this flag is clear, so frames will not be delayed in
    * order to calculate durations.
    * IMPORTANT: If you will be passing encoded frames to
    * AddMediaSampleFromEncodedFrame, you must set this flag to true.
    */
   kICMCompressionSessionOptionsPropertyID_DurationsNeeded = 'need', /* Boolean,
Read/Write */
   /*
    * The maximum interval between key frames, also known as the key
    * frame rate.
    * Key frames, also known as sync frames, reset inter-frame
    * dependencies; decoding a key frame is sufficient to prepare a
    * decompressor for correctly decoding the difference frames that
    * follow.
    * Compressors are allowed to generate key frames more frequently if
    * this would result in more efficient compression.
    * The default key frame interval is 0, which indicates that the
    * compressor should choose where to place all key frames. A key
    * frame interval of 1 indicates that every frame must be a key
    * frame, 2 indicates that at least every other frame must be a key
    * frame, etc.
    */
   kICMCompressionSessionOptionsPropertyID_MaxKeyFrameInterval = 'kyfr', /* SInt32,
  Read/Write */
   /*
    * The requested maximum interval between partial sync frames. If the
    * interval is n, any sequence of n successive frames must include at
    * least one key or partial sync frame.
    * Where supported, partial sync frames perform a partial reset of
    * inter-frame dependencies; decoding two partial sync frames and the
    * non-droppable difference frames between them is sufficient to
    * prepare a decompressor for correctly decoding the difference
    * frames that follow.
    * Compressors are allowed to generate partial sync frames more
    * frequently if this would result in more efficient compression.
    *
    * The default partial sync frame interval is 0, which indicates that
    * the compressor should choose where to place partial sync frames. A
    * partial sync frame interval of 1 means there can be no difference
    * frames, so it is equivalent to a key frame interval of 1. A
    * partial sync frame interval of 2 means that every other frame must
```

```
   * be a key frame or a partial sync frame.
   * Compressors that do not support partial sync frames will ignore
   * this setting.
   */
  kICMCompressionSessionOptionsPropertyID_MaxPartialSyncFrameInterval = 'psfr', /*
SInt32, Read/Write */
  /*
   * Enables the compressor to modify frame times.
   * Some compressors are able to identify and coalesce runs of
   * identical frames and output single frames with longer duration, or
   * output frames at a different frame rate from the original. This
   * feature is controlled by the "allow frame time changes" flag. By
   * default, this flag is set to false, which forces compressors to
   * emit one encoded frame for every source frame, and to preserve
   * frame display times.
   * (Note: this feature replaces the practice of having compressors
   * return special high similarity values to indicate that frames
   * could be dropped.)
   * If you want to allow the compressor to modify frame times in order
   * to improve compression performance, enable frame time changes.
   */
  kICMCompressionSessionOptionsPropertyID_AllowFrameTimeChanges = '+ ok', /* Boolean,
Read/Write */
  /*
   * Enables the compressor to call the encoded-frame callback from a
   * different thread.
   * By default, the flag is false, which means that the compressor
   * must call the encoded-frame callback from the same thread that
   * ICMCompressionSessionEncodeFrame and
   * ICMCompressionSessionCompleteFrames were called on.
   */
  kICMCompressionSessionOptionsPropertyID_AllowAsyncCompletion = 'asok', /* Boolean,
Read/Write */
  /*
   * The maximum frame delay count is the maximum number of frames that
   * a compressor is allowed to hold before it must output a compressed
   * frame. It limits the number of frames that may be held in the
   * "compression window". If the maximum frame delay count is M, then
   * before the call to encode frame N returns, frame N-M must have
   * been emitted.
   * The default is kICMUnlimitedFrameDelayCount, which sets no limit
   * on the compression window.
   */
  kICMCompressionSessionOptionsPropertyID_MaxFrameDelayCount = 'cwin', /* SInt32,
Read/Write */
  /*
   * The maximum frame delay time is the maximum difference between a
   * source frame's display time and the corresponding encoded frame's
   * decode time. It limits the span of display time that may be held
   * in the "compression window". If the maximum frame delay time is
   * TM, then before the call to encode a frame with display time TN
   * returns, all frames with display times up to and including TN-TM
   * must have been emitted.
   * The default is kICMUnlimitedFrameDelayTime, which sets no time
   * limit on the compression window.
   */
  kICMCompressionSessionOptionsPropertyID_MaxFrameDelayTime = 'cwit', /* TimeValue64,
Read/Write */
```

```
  /*
   * Sets a specific compressor component or component instance to be
   * used, or one of the wildcards anyCodec, bestSpeedCodec,
   * bestFidelityCodec, or bestCompressionCodec.
   * Use this API to force the Image Compression Manager to use a
   * specific compressor component or compressor component instance.
   * (If you pass in a component instance that you opened, the ICM will
   * not close that instance; you must do so after the compression
   * session is released.) To allow the Image Compression Manager to
   * choose the compressor component, set the compressorComponent to
   * anyCodec (the default), bestSpeedCodec, bestFidelityCodec or
   * bestCompressionCodec.
   */
  kICMCompressionSessionOptionsPropertyID_CompressorComponent = 'imco', /*
CompressorComponent, Read/Write */
  /*
   * A handle containing compressor settings. The compressor will be
   * configured with these settings (by a call to
   * ImageCodecSetSettings) during ICMCompressionSessionCreate.
   */
  kICMCompressionSessionOptionsPropertyID_CompressorSettings = 'cost', /* Handle,
 Read/Write */
  /*
   * The depth for compression.
   * If a compressor does not support a specific depth, the closest
   * supported depth will be used (preferring deeper depths to
   * shallower depths). The default depth is k24RGBPixelFormat.
   */
  kICMCompressionSessionOptionsPropertyID_Depth = 'deep', /* UInt32, Read/Write */
  /*
   * The color table for compression.  Used with indexed-color depths.
   *
   * Clients who get this property are responsible for disposing the
   * returned CTabHandle.
   */
  kICMCompressionSessionOptionsPropertyID_ColorTable = 'clut', /* CTabHandle,
Read/Write*/
  /*
   * The compression quality.
   * This value is always used to set the spatialQuality; if temporal
   * compression is enabled, it is also used to set temporalQuality.
   * <BR> The default quality is codecNormalQuality.
   */
  kICMCompressionSessionOptionsPropertyID_Quality = 'qual', /* CodecQ, Read/Write
 */
  /*
   * The long-term desired average data rate in bytes per second.
   *  This is not a hard limit.
   * The default data rate is zero, which indicates that the quality
   * setting should determine the size of compressed data.
   * Note that data rate settings only have an effect when timing
   * information is provided for source frames, and that some codecs do
   * not support limiting to specified data rates.
   */
  kICMCompressionSessionOptionsPropertyID_AverageDataRate = 'aver', /* SInt32,
Read/Write */
  /*
   * Zero, one or two hard limits on data rate.
```

```
   * Each hard limit is described by a data size in bytes and a
   * duration in seconds, and requires that the total size of
   * compressed data for any contiguous segment of that duration (in
   * decode time) must not exceed the data size.
   * By default, no data rate limits are set.
   * When setting this property, the inPropValueSize parameter should
   * be the number of data rate limits multiplied by
   * sizeof(ICMDataRateLimit).
   * Note that data rate settings only have an effect when timing
   * information is provided for source frames, and that some codecs do
   * not support limiting to specified data rates.
   */
  kICMCompressionSessionOptionsPropertyID_DataRateLimits = 'hard', /* C array of
ICMDataRateLimit struct, Read/Write */
  /*
   * The current number of data rate limits.
   */
  kICMCompressionSessionOptionsPropertyID_DataRateLimitCount = 'har#', /* UInt32,
 Read */
  /*
   * The maximum allowed number of data rate limits.  (Currently 2.)
   */
  kICMCompressionSessionOptionsPropertyID_MaxDataRateLimits = 'mhar', /* UInt32,
Read */
  /*
   * Indicates that the source was previously compressed.
   * This property is purely an optional, informational hint to the
   * compressor; by default it is false.
   */
  kICMCompressionSessionOptionsPropertyID_WasCompressed = 'wasc', /* Boolean,
Read/Write */
  /*
   * Recommends a CPU time budget for the compressor in microseconds
   * per frame.
   * Zero means to go as fast as possible.
   * By default, this is set to kICMUnlimitedCPUTimeBudget, which sets
   * no limit.
   * This is an advisory hint to the compressor, and some compressors
   * may ignore it. Multithreaded compressors may use this amount of
   * CPU time on each processor.
   * Compressors should not feel compelled to use the full time budget
   * if they complete ahead of time!
   */
  kICMCompressionSessionOptionsPropertyID_CPUTimeBudget = 'cput', /* UInt32,
Read/Write */
  /*
   * Storage for multi-pass compression.
   * To enable multipass compression, the client must provide a storage
   * location for multipass data.  Use
   * ICMMultiPassStorageCreateWithTemporaryFile to have the ICM store
   * it in a temporary file.  Use
   * ICMMultiPassStorageCreateWithCallbacks to manage the storage
   * yourself.
   * Note that the amount of multipass data to be stored can be
   * substantial; it could be greater than the size of the output movie
   * file.
   * If this property is not NULL, the client must call
   * ICMCompressionSessionBeginPass and ICMCompressionSessionEndPass
```

```
   * around groups of calls to ICMCompressionSessionEncodeFrame.
   *  By default, this property is NULL and multipass compression is
   * not enabled. The compression session options object retains the
   * multipass storage object, when one is set.
   */
  kICMCompressionSessionOptionsPropertyID_MultiPassStorage = 'imps', /*
ICMMultiPassStorageRef, Read/Write */
  /*
   * Indicates the number of source frames, if known. If nonzero, this
   * should be the exact number of times that the client calls
   * ICMCompressionSessionEncodeFrame in each pass.
   * The default is 0, which indicates that the number of source frames
   * is not known.
   */
  kICMCompressionSessionOptionsPropertyID_SourceFrameCount = 'frco', /* UInt64,
Read/Write */
  /*
   * Indicates the expected frame rate, if known. The frame rate is
   * measured in frames per second. This is not used to control the
   * frame rate; it is provided as a hint to the compressor so that it
   * can set up internal configuration before compression begins. The
   * actual frame rate will depend on frame durations and may vary. By
   * default, this is zero, indicating "unknown".
   */
  kICMCompressionSessionOptionsPropertyID_ExpectedFrameRate = 'fran', /* Fixed,
Read/Write */
  /*
   * Indicates how source frames to a compression session should be
   * scaled if the dimensions and/or display aspect ratio do not match.
   */
 kICMCompressionSessionOptionsPropertyID_ScalingMode = 'scam', /* OSType, Read/Write
*/
  /*
   * Describes the clean aperture for compressed frames. Note that if
   * the compressor enforces a clean aperture, it will override this
   * setting. The clean aperture will be set on the output image
   * description and may affect scaling in some scaling modes. By
   * default, this is all zeros, meaning unset.
   */
 kICMCompressionSessionOptionsPropertyID_CleanAperture = 'clap', /* Native-endian
CleanApertureImageDescriptionExtension, Read/Write */
  /*
   * Describes the pixel aspect ratio for compressed frames. Note that
   * if the compressor enforces a pixel aspect ratio, it will override
   * this setting. The pixel aspect ratio will be set on the output
   * image description and may affect scaling in some scaling modes. By
   * default, this is all zeros, meaning unset.
   */
 kICMCompressionSessionOptionsPropertyID_PixelAspectRatio = 'pasp', /* Native-endian
PixelAspectRatioImageDescriptionExtension, Read/Write */
  /*
   * Describes the number and order of fields for compressed frames.
   * Note that if the compressor enforces field info, it will override
   * this setting. The field info will be set on the output image
   * description and may affect scaling in some scaling modes. By
   * default, this is all zeros, meaning unset.
   */
  kICMCompressionSessionOptionsPropertyID_FieldInfo = 'fiel' /*
```

```
FieldInfoImageDescriptionExtension2, Read/Write */
};
enum {
  /*
   * Class identifier for compression session properties.
   */
  kQTPropertyClass_ICMCompressionSession = 'icse',
  /*
   * The time scale for the compression session.
   */
  kICMCompressionSessionPropertyID_TimeScale = 'tscl', /* TimeScale, Read */
  /*
   * The compressor's pixel buffer attributes for the compression
   * session. You can use these to create a pixel buffer pool for
   * source pixel buffers. Note that this is not the same as the
   * sourcePixelBufferAttributes passed in to
   * ICMCompressionSessionCreate. Getting this property does not change
   * its retain count.
   */
  kICMCompressionSessionPropertyID_CompressorPixelBufferAttributes = 'batt', /*
CFDictionaryRef, Read */
  /*
   * A pool that can provide ideal source pixel buffers for a
   * compression session. The compression session creates this pixel
   * buffer pool based on the compressor's pixel buffer attributes and
   * any pixel buffer attributes passed in to
   * ICMCompressionSessionCreate. If the source pixel buffer attributes
   * and the compressor pixel buffer attributes can not be reconciled,
   * the pool is based on the source pixel buffer attributes and the
   * ICM converts each CVPixelBuffer internally.
   */
  kICMCompressionSessionPropertyID_PixelBufferPool = 'pool', /* CVPixelBufferPoolRef,
 Read */
  /*
   * The image description for the compression session. For some
   * codecs, the image description may not be available before the
   * first frame is compressed. Multiple calls to retrieve this
   * property will return the same handle. The ICM will dispose this
   * handle when the compression session is disposed.
   * IMPORTANT: The caller must NOT dispose this handle.
   */
  kICMCompressionSessionPropertyID_ImageDescription = 'idsc' /*
ImageDescriptionHandle, Read */
};
enum {
  /*
   * Class identifier for decompression frame options object properties.
   */
  kQTPropertyClass_ICMDecompressionFrameOptions = 'idfo',
  /*
   * A specific pixel buffer that the frame should be decompressed
   * into. Setting this circumvents the pixel buffer pool mechanism. If
   * this buffer is not compatible with the codec's pixel buffer
   * requirements, decompression will fail.
   */
  kICMDecompressionFrameOptionsPropertyID_DestinationPixelBuffer = 'cvpb' /*
CVPixelBufferRef, Read/Write */
};
```

```
enum {
  /*
   * Class identifier for decompression session options object
   * properties.
   */
  kQTPropertyClass_ICMDecompressionSessionOptions = 'idso',
  /*
   * By default, this is true, meaning that frames must be output in
   * display order. Set this to false to allow frames to be output in
   * decode order rather than in display order.
   */
  kICMDecompressionSessionOptionsPropertyID_DisplayOrderRequired = 'dorq', /*
Boolean, Read/Write */
  /*
   * A specific decompressor component or component instance to be
   * used, or one of the wildcards anyCodec, bestSpeedCodec,
   * bestFidelityCodec, or bestCompressionCodec.
   * By default, this is anyCodec.
   */
  kICMDecompressionSessionOptionsPropertyID_DecompressorComponent = 'imdc', /*
DecompressorComponent, Read/Write */
  /*
   * The decompression accuracy.
   * The default accuracy is codecNormalQuality.
   */
  kICMDecompressionSessionOptionsPropertyID_Accuracy = 'acur', /* CodecQ, Read/Write
 */
  /*
   * Requests special handling of fields. Not all codecs will obey this
   * request; some codecs will only handle it at certain accuracy
   * levels. Ignored for non-interlaced content.
   */
  kICMDecompressionSessionOptionsPropertyID_FieldMode = 'fiel', /* ICMFieldMode,
Read/Write */
  /*
   * The maximum number of buffers ahead of the current time that
   * should be decompressed. Used in sessions that target visual
   * contexts. By default, the number of buffers will be determined
   * from the visual context.
   */
  kICMDecompressionSessionOptionsPropertyID_MaxBufferCount = 'm#bf', /* UInt32,
Read/Write */
  /*
   * The minimum time ahead of the current time that frames should be
   * decompressed. Used in sessions that target visual contexts. By
   * default, the output-ahead time will be determined from the visual
   * context.
   */
  kICMDecompressionSessionOptionsPropertyID_OutputAheadTime = 'futu' /* TimeRecord,
 Read/Write */
};
enum {
  /*
   * Class identifier for decompression session properties.
   */
  kQTPropertyClass_ICMDecompressionSession = 'icds',
  /*
   * The non-scheduled display time for a decompression session.
```

```
    * Setting this requests display of the non-scheduled queued frame at
    * that display time, if there is one.
    * See ICMDecompressionSessionSetNonScheduledDisplayTime.
    */
  kICMDecompressionSessionPropertyID_NonScheduledDisplayTime = 'nsti', /*
ICMNonScheduledDisplayTime, Read/Write */
  /*
    * The direction for non-scheduled display time.
    * See ICMDecompressionSessionSetNonScheduledDisplayDirection.
    */
  kICMDecompressionSessionPropertyID_NonScheduledDisplayDirection = 'nsdu', /*
Fixed, Read/Write */
  /*
    * The pixel buffer pool from which emitted pixel buffers are
    * allocated. Getting this does not change the retain count of the
    * pool.
    */
  kICMDecompressionSessionPropertyID_PixelBufferPool = 'pool', /*
CVPixelBufferPoolRef, Read */
  /*
    * Indicates whether the a common pixel buffer pool is shared between
    * the decompressor and the session client. This is false if separate
    * pools are used because the decompressor's and the client's pixel
    * buffer attributes were incompatible.
    */
  kICMDecompressionSessionPropertyID_PixelBufferPoolIsShared = 'plsh' /* Boolean,
 Read */
};
enum {
  /*
    * Class identifier for image description properties.
    */
  kQTPropertyClass_ImageDescription = 'idsc',
  /*
    * The width of the encoded image. Usually, but not always, this is
    * the ImageDescription's width field.
    */
  kICMImageDescriptionPropertyID_EncodedWidth = 'encw', /* SInt32, Read/Write */
  /*
    * The height of the encoded image. Usually, but not always, this is
    * the ImageDescription's height field.
    */
  kICMImageDescriptionPropertyID_EncodedHeight = 'ench', /* SInt32, Read/Write */
  /*
    * 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.
    */
  kICMImageDescriptionPropertyID_CleanAperture = 'clap', /* Native-endian
CleanApertureImageDescriptionExtension, Read/Write */
  /*
    * Describes the pixel aspect ratio. If not specified explicitly in
    * the image description, a square (1:1) pixel aspect ratio will be
    * returned.
    */
  kICMImageDescriptionPropertyID_PixelAspectRatio = 'pasp', /* Native-endian
PixelAspectRatioImageDescriptionExtension, Read/Write */
  /*
```

```
    * 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.
    */
   kICMImageDescriptionPropertyID_DisplayWidth = 'disw', /* SInt32, Read */
   /*
    * 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.
    */
   kICMImageDescriptionPropertyID_DisplayHeight = 'dish', /* SInt32, Read */
   /*
    * 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.
    */
   kICMImageDescriptionPropertyID_ProductionDisplayWidth = 'pdsw', /* SInt32, Read
 */
   /*
    * 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.
    */
   kICMImageDescriptionPropertyID_ProductionDisplayHeight = 'pdsh', /* SInt32, Read
 */
   /*
    * Color information, if available in the
    * NCLCColorInfoImageDescriptionExtension format.
    */
   kICMImageDescriptionPropertyID_NCLCColorInfo = 'nclc', /* Native-endian
NCLCColorInfoImageDescriptionExtension, Read/Write */
   /*
    * The gamma level described by the image description.
    */
   kICMImageDescriptionPropertyID_GammaLevel = 'gama', /* Fixed, Read/Write */
   /*
    * Information about the number and order of fields, if available.
    */
   kICMImageDescriptionPropertyID_FieldInfo = 'fiel', /*
FieldInfoImageDescriptionExtension2, Read/Write */
   /*
    * The offset in bytes from the start of one row to the next. Only
    * valid if the codec type is a chunky pixel format.
    */
   kICMImageDescriptionPropertyID_RowBytes = 'rowb', /* SInt32, Read/Write */
   /*
    * A track width suitable for passing to NewMovieTrack when creating
    * a new track to hold this image data.
    */
   kICMImageDescriptionPropertyID_ClassicTrackWidth = 'claw', /* Fixed, Read */
   /*
    * A track height suitable for passing to NewMovieTrack when creating
    * a new track to hold this image data.
    */
```

```
  kICMImageDescriptionPropertyID_ClassicTrackHeight = 'clah' /* Fixed, Read */
};
enum {
  /*
   * In this pass the compressor shall output encoded frames.
   */
  kICMCompressionPassMode_OutputEncodedFrames = 1L << 0,
  /*
   * In this pass the client need not provide source frame buffers.
   */
  kICMCompressionPassMode_NoSourceFrames = 1L << 1,
  /*
   * In this pass the compressor may write private data to multipass
   * storage.
   */
  kICMCompressionPassMode_WriteToMultiPassStorage = 1L << 2,
  /*
   * In this pass the compressor may read private data from multipass
   * storage.
   */
  kICMCompressionPassMode_ReadFromMultiPassStorage = 1L << 3,
  /*
   * The compressor will set this flag to indicate that it will not be
   * able to output encoded frames in the coming pass. If this flag is
   * not set, then the client is allowed to set the
   * kICMCompressionPassMode_OutputEncodedFrames flag before calling
   * ICMCompressionSessionBeginPass.
   */
  kICMCompressionPassMode_NotReadyToOutputEncodedFrames = 1L << 4
};
enum {
  /*
   * Indicates that this is the last call for this sourceFrameRefCon.
   */
  kICMSourceTracking_LastCall   = 1L << 0,
  /*
   * Indicates that the session is done with the source pixel buffer
   * and has released any reference to it that it had.
   */
  kICMSourceTracking_ReleasedPixelBuffer = 1L << 1,
  /*
   * Indicates that this frame was encoded.
   */
  kICMSourceTracking_FrameWasEncoded = 1L << 2,
  /*
   * Indicates that this frame was dropped.
   */
  kICMSourceTracking_FrameWasDropped = 1L << 3,
  /*
   * Indicates that this frame was merged into other frames.
   */
  kICMSourceTracking_FrameWasMerged = 1L << 4,
  /*
   * Indicates that the time stamp of this frame was modified.
   */
  kICMSourceTracking_FrameTimeWasChanged = 1L << 5,
  /*
   * Indicates that the ICM has copied the image from the source pixel
```

```
   * buffer into another pixel buffer because the source pixel buffer
   * was not compatible with the compressor's required pixel buffer
   * attributes.
   */
  kICMSourceTracking_CopiedPixelBuffer = 1L << 6
};
enum {
  /*
   * The full width and height of source frames shall be scaled to the
   * full width and height of the destination. This is the default if
   * no other scaling mode is specified.
   */
  kICMScalingMode_StretchProductionAperture = 'sp2p',
  /*
   * The clean aperture of the source frames shall be scaled to the
   * clean aperture of the destination.
   */
  kICMScalingMode_StretchCleanAperture = 'sc2c',
  /*
   * The clean aperture of the source frames shall be 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 will be centered with black bars
   * above and below, or to the left and right.
   */
  kICMScalingMode_Letterbox     = 'lett',
  /*
   * The clean aperture of the source frames shall be 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 will be centered and cropped.
   */
  kICMScalingMode_Trim          = 'trim'
};
```

Constants

`kICMCompressionFrameOptionsPropertyID_ForceKeyFrame`

Boolean, ReadWrite.

Available in Mac OS X v10.3 and later.

Declared in `ImageCompression.h`.

`kICMCompressionFrameOptionsPropertyID_FrameType`

`ICMFrameType`, ReadWrite.

Available in Mac OS X v10.3 and later.

Declared in `ImageCompression.h`.

`kQTPropertyClass_ICMCompressionSessionOptions`

Class identifier for compression session option object properties. Also `'icso'`.

Available in Mac OS X v10.3 and later.

Declared in `ImageCompression.h`.

`kICMCompressionSessionOptionsPropertyID_AllowTemporalCompression`

Enables temporal compression of P-frames and B-frames. By default, temporal compression is disabled. Also `'p ok'`.

Available in Mac OS X v10.3 and later.

Declared in `ImageCompression.h`.

`kICMCompressionSessionOptionsPropertyID_AllowFrameReordering`
: Enables frame reordering. To encode B-frames a compressor must reorder frames, which may mean that the order in which they are emitted and stored (the decode order) may be different from the order in which they are presented to the compressor (the display order). By default, frame reordering is disabled. To encode using B-frames, you must enable frame reordering by passing TRUE in this property. Also `'b ok'`.

  Available in Mac OS X v10.3 and later.

  Declared in `ImageCompression.h`.

`kICMCompressionSessionOptionsPropertyID_DurationsNeeded`
: Indicates that durations of emitted frames are needed. If this option is set and source frames are provided with times but not durations, then frames will be delayed so that durations can be calculated as the difference between one frame's time stamp and the next frame's time stamp. By default, this flag is FALSE, so frames will not be delayed in order to calculate durations. If you pass encoded frames to `AddMediaSampleFromEncodedFrame`, you must set this flag to TRUE. Also `'need'`.

  Available in Mac OS X v10.3 and later.

  Declared in `ImageCompression.h`.

`kICMCompressionSessionOptionsPropertyID_MaxKeyFrameInterval`
: The maximum interval between key frames, also known as the key frame rate. Compressors are allowed to generate key frames more frequently if this would result in more efficient compression. The default key frame interval is 0, which indicates that the compressor should choose where to place all key frames. This differs from previous practice, in which a key frame rate of zero disabled temporal compression. Also `'kyfr'`.

  Available in Mac OS X v10.3 and later.

  Declared in `ImageCompression.h`.

`kICMCompressionSessionOptionsPropertyID_MaxPartialSyncFrameInterval`
: SInt32, ReadWrite.

  Available in Mac OS X v10.3 and later.

  Declared in `ImageCompression.h`.

`kICMCompressionSessionOptionsPropertyID_AllowFrameTimeChanges`
: Enables the compressor to modify frame times, improving its performance. Some compressors are able to identify and coalesce runs of identical frames and emit single frames with longer duration, or emit frames at a different frame rate from the original. By default, this flag is set to FALSE, which forces the compressor to emit one encoded frame for every source frame and to preserve frame display times. This option replaces the practice of having compressors return special high similarity values to indicate that frames can be dropped. Also `'+ ok'`.

  Available in Mac OS X v10.3 and later.

  Declared in `ImageCompression.h`.

`kICMCompressionSessionOptionsPropertyID_AllowAsyncCompletion`
: Enables the compressor to call the encoded-frame callback from a different thread. By default this option is FALSE, which means that the compressor must call the encoded-frame callback from the same thread as `ICMCompressionSessionEncodeFrame` and `ICMCompressionSessionCompleteFrames`. Also `'asok'`.

  Available in Mac OS X v10.3 and later.

  Declared in `ImageCompression.h`.

`kICMCompressionSessionOptionsPropertyID_MaxFrameDelayCount`
: The maximum frame delay count is the maximum number of frames that a compressor is allowed to hold before it must output a compressed frame. This value limits the number of frames that may be held in the compression window. If the maximum frame delay count is M, then before the call to encode frame N returns, frame N-M must have been emitted. The default value is `kICMUnlimitedFrameDelayCount`, which sets no limit on the compression window. Also `'cwin'`.

  Available in Mac OS X v10.3 and later.

  Declared in `ImageCompression.h`.

`kICMCompressionSessionOptionsPropertyID_MaxFrameDelayTime`
: TimeValue64, ReadWrite.

  Available in Mac OS X v10.3 and later.

  Declared in `ImageCompression.h`.

`kICMCompressionSessionOptionsPropertyID_CompressorComponent`
: Sets a specific compressor component or component instance to be used, or passes one of the wildcards `anyCodec`, `bestSpeedCodec`, `bestFidelityCodec`, or `bestCompressionCodec`. Pass this option to force the Image Compression Manager to use a specific compressor component or compressor component instance. To allow the Image Compression Manager to choose the compressor component, set the `compressorComponent` to `anyCodec` (the default), `bestSpeedCodec`, `bestFidelityCodec`, or `bestCompressionCodec`. If you pass in a component instance that you opened, the ICM will not close that instance; you must do so after the compression session is released. Also `'imco'`.

  Available in Mac OS X v10.3 and later.

  Declared in `ImageCompression.h`.

`kICMCompressionSessionOptionsPropertyID_CompressorSettings`
: A handle containing compressor settings. The compressor will be configured with these settings (by a call to `ImageCodecSetSettings`) during the `ICMCompressionSessionCreate` process. Also `'cost'`.

  Available in Mac OS X v10.3 and later.

  Declared in `ImageCompression.h`.

`kICMCompressionSessionOptionsPropertyID_Depth`
: UInt32, ReadWrite.

  Available in Mac OS X v10.3 and later.

  Declared in `ImageCompression.h`.

`kICMCompressionSessionOptionsPropertyID_ColorTable`
: The color table for compression, used with indexed-color depths. Clients who are passed this property are responsible for disposing the returned `CTabHandle`. Also `'clut'`.

  Available in Mac OS X v10.3 and later.

  Declared in `ImageCompression.h`.

`kICMCompressionSessionOptionsPropertyID_Quality`
: The compression quality. This value is always used to set the spatial quality; if temporal compression is enabled, it is also used to set temporal quality. The default quality is `codecNormalQuality`. Also `'qual'`.

  Available in Mac OS X v10.3 and later.

  Declared in `ImageCompression.h`.

`kICMCompressionSessionOptionsPropertyID_AverageDataRate`
: The long-term desired average data rate in bytes per second. This is not an absolute limit. The default data rate is zero, indicating that the setting of `kICMCompressionSessionOptionsPropertyID_Quality` should determine the size of compressed data. Data rate settings have effect only when timing information is provided for source frames. Some codecs do not accept limiting to specified data rates. Also `'aver'`.

  Available in Mac OS X v10.3 and later.

  Declared in `ImageCompression.h`.

`kICMCompressionSessionOptionsPropertyID_DataRateLimits`
: Zero, one, or two hard limits on data rate. Each hard limit is described by a data size in bytes and a duration in seconds. It requires that the total size of compressed data for any contiguous segment of that duration (in decode time) must not exceed the data size. By default, no data rate limits are set. When setting this property, the `inPropValueSize` parameter should be the number of data rate limits multiplied by `sizeof(ICMDataRateLimit)`. Data rate settings have an effect only when timing information is provided for source frames. Some codecs do not accept limiting to specified data rates. Also `'hard'`.

  Available in Mac OS X v10.3 and later.

  Declared in `ImageCompression.h`.

`kICMCompressionSessionOptionsPropertyID_DataRateLimitCount`
: UInt32, Read.

  Available in Mac OS X v10.3 and later.

  Declared in `ImageCompression.h`.

`kICMCompressionSessionOptionsPropertyID_MaxDataRateLimits`
: UInt32, Read.

  Available in Mac OS X v10.3 and later.

  Declared in `ImageCompression.h`.

`kICMCompressionSessionOptionsPropertyID_WasCompressed`
: Indicates that the source was previously compressed. This property is an optional information hint to the compressor; by default it is FALSE. Also `'wasc'`.

  Available in Mac OS X v10.3 and later.

  Declared in `ImageCompression.h`.

`kICMCompressionSessionOptionsPropertyID_CPUTimeBudget`
: UInt32, ReadWrite.

  Available in Mac OS X v10.3 and later.

  Declared in `ImageCompression.h`.

`kICMCompressionSessionOptionsPropertyID_MultiPassStorage`
A multipass compression client must provide a storage location for multipass data. Pass `ICMMultiPassStorageCreateWithTemporaryFile` to make the ICM store multipass data in a temporary file. Pass `ICMMultiPassStorageCreateWithCallbacks` to manage the storage yourself. Note that the amount of multipass data to be stored can be substantial; it could be greater than the size of the output movie file. If this property is not NULL, the client must call `ICMCompressionSessionBeginPass` and `ICMCompressionSessionEndPass` around groups of calls to `ICMCompressionSessionEncodeFrame`. By default, this property is NULL and multipass compression is not enabled. The compression session options object retains the multipass storage object when one is set. Also `'imps'`.

Available in Mac OS X v10.3 and later.

Declared in `ImageCompression.h`.

`kICMCompressionSessionOptionsPropertyID_SourceFrameCount`
UInt64, ReadWrite.

Available in Mac OS X v10.3 and later.

Declared in `ImageCompression.h`.

`kICMCompressionSessionOptionsPropertyID_ExpectedFrameRate`
`Fixed`, ReadWrite.

Available in Mac OS X v10.3 and later.

Declared in `ImageCompression.h`.

`kICMCompressionSessionOptionsPropertyID_ScalingMode`
`OSType`, ReadWrite.

Available in Mac OS X v10.3 and later.

Declared in `ImageCompression.h`.

`kICMCompressionSessionOptionsPropertyID_CleanAperture`
Native-endian `CleanApertureImageDescriptionExtension`, ReadWrite.

Available in Mac OS X v10.3 and later.

Declared in `ImageCompression.h`.

`kICMCompressionSessionOptionsPropertyID_PixelAspectRatio`
Native-endian `PixelAspectRatioImageDescriptionExtension`, ReadWrite.

Available in Mac OS X v10.3 and later.

Declared in `ImageCompression.h`.

`kICMCompressionSessionOptionsPropertyID_FieldInfo`
FieldInfoImageDescriptionExtension2, ReadWrite.

Available in Mac OS X v10.3 and later.

Declared in `ImageCompression.h`.

`kQTPropertyClass_ICMCompressionSession`
Class identifier for compression session properties. Also `'icse'`.

Available in Mac OS X v10.3 and later.

Declared in `ImageCompression.h`.

`kICMCompressionSessionPropertyID_TimeScale`
The time scale for the compression session. Also `'tscl'`.

Available in Mac OS X v10.3 and later.

Declared in `ImageCompression.h`.

`kICMCompressionSessionPropertyID_CompressorPixelBufferAttributes`
: The compressor's pixel buffer attributes for the compression session. You can use these to create a pixel buffer pool for source pixel buffers. This is not the same as the `sourcePixelBufferAttributes` property passed to `ICMCompressionSessionCreate`. Getting this property does not change its retain count. Also `'batt'`.

  Available in Mac OS X v10.3 and later.

  Declared in `ImageCompression.h`.

`kICMCompressionSessionPropertyID_PixelBufferPool`
: A pool that can provide ideal source pixel buffers for a compression session. The compression session creates this pixel buffer pool based on the compressor's pixel buffer attributes and any pixel buffer attributes passed in to `ICMCompressionSessionCreate`. If the source pixel buffer attributes and the compressor pixel buffer attributes can not be reconciled, the pool is based on the source pixel buffer attributes and the ICM converts each `CVPixelBuffer` internally. Also `'pool'`.

  Available in Mac OS X v10.3 and later.

  Declared in `ImageCompression.h`.

`kICMCompressionSessionPropertyID_ImageDescription`
: The image description for a compression session. For some codecs, the image description may not be available before the first frame is compressed. Multiple calls to retrieve this property will return the same handle. The ICM will dispose of this handle when the compression session is disposed; the caller must not dispose of it. Also `'idsc'`.

  Available in Mac OS X v10.3 and later.

  Declared in `ImageCompression.h`.

`kICMDecompressionFrameOptionsPropertyID_DestinationPixelBuffer`
: `CVPixelBufferRef`, ReadWrite.

  Available in Mac OS X v10.3 and later.

  Declared in `ImageCompression.h`.

`kICMDecompressionSessionOptionsPropertyID_DisplayOrderRequired`
: Boolean, ReadWrite.

  Available in Mac OS X v10.3 and later.

  Declared in `ImageCompression.h`.

`kICMDecompressionSessionOptionsPropertyID_DecompressorComponent`
: `DecompressorComponent`, ReadWrite.

  Available in Mac OS X v10.3 and later.

  Declared in `ImageCompression.h`.

`kICMDecompressionSessionOptionsPropertyID_Accuracy`
: `CodecQ`, ReadWrite.

  Available in Mac OS X v10.3 and later.

  Declared in `ImageCompression.h`.

`kICMDecompressionSessionOptionsPropertyID_FieldMode`
: `ICMFieldMode`, ReadWrite.

  Available in Mac OS X v10.3 and later.

  Declared in `ImageCompression.h`.

`kICMDecompressionSessionOptionsPropertyID_MaxBufferCount`
UInt32, ReadWrite.

Available in Mac OS X v10.3 and later.

Declared in `ImageCompression.h`.

`kICMDecompressionSessionOptionsPropertyID_OutputAheadTime`
`TimeRecord`, ReadWrite.

Available in Mac OS X v10.3 and later.

Declared in `ImageCompression.h`.

`kICMDecompressionSessionPropertyID_NonScheduledDisplayTime`
`ICMNonScheduledDisplayTime`, ReadWrite.

Available in Mac OS X v10.3 and later.

Declared in `ImageCompression.h`.

`kICMDecompressionSessionPropertyID_NonScheduledDisplayDirection`
`Fixed`, ReadWrite.

Available in Mac OS X v10.3 and later.

Declared in `ImageCompression.h`.

`kICMDecompressionSessionPropertyID_PixelBufferPool`
`CVPixelBufferPoolRef`, Read.

Available in Mac OS X v10.3 and later.

Declared in `ImageCompression.h`.

`kICMDecompressionSessionPropertyID_PixelBufferPoolIsShared`
Boolean, Read.

Available in Mac OS X v10.3 and later.

Declared in `ImageCompression.h`.

`kICMImageDescriptionPropertyID_EncodedWidth`
SInt32, ReadWrite.

Available in Mac OS X v10.3 and later.

Declared in `ImageCompression.h`.

`kICMImageDescriptionPropertyID_EncodedHeight`
SInt32, ReadWrite.

Available in Mac OS X v10.3 and later.

Declared in `ImageCompression.h`.

`kICMImageDescriptionPropertyID_CleanAperture`
Native-endian `CleanApertureImageDescriptionExtension`, ReadWrite.

Available in Mac OS X v10.3 and later.

Declared in `ImageCompression.h`.

`kICMImageDescriptionPropertyID_PixelAspectRatio`
Native-endian `PixelAspectRatioImageDescriptionExtension`, ReadWrite.

Available in Mac OS X v10.3 and later.

Declared in `ImageCompression.h`.

`kICMImageDescriptionPropertyID_DisplayWidth`
SInt32, Read.

Available in Mac OS X v10.3 and later.

Declared in `ImageCompression.h`.

`kICMImageDescriptionPropertyID_DisplayHeight`
SInt32, Read.

Available in Mac OS X v10.3 and later.

Declared in `ImageCompression.h`.

`kICMImageDescriptionPropertyID_ProductionDisplayWidth`
SInt32, Read.

Available in Mac OS X v10.3 and later.

Declared in `ImageCompression.h`.

`kICMImageDescriptionPropertyID_ProductionDisplayHeight`
SInt32, Read.

Available in Mac OS X v10.3 and later.

Declared in `ImageCompression.h`.

`kICMImageDescriptionPropertyID_NCLCColorInfo`
Native-endian `NCLCColorInfoImageDescriptionExtension`, ReadWrite.

Available in Mac OS X v10.3 and later.

Declared in `ImageCompression.h`.

`kICMImageDescriptionPropertyID_GammaLevel`
`Fixed`, ReadWrite.

Available in Mac OS X v10.3 and later.

Declared in `ImageCompression.h`.

`kICMImageDescriptionPropertyID_FieldInfo`
FieldInfoImageDescriptionExtension2, ReadWrite.

Available in Mac OS X v10.3 and later.

Declared in `ImageCompression.h`.

`kICMImageDescriptionPropertyID_RowBytes`
SInt32, ReadWrite.

Available in Mac OS X v10.3 and later.

Declared in `ImageCompression.h`.

`kICMImageDescriptionPropertyID_ClassicTrackWidth`
`Fixed`, Read.

Available in Mac OS X v10.3 and later.

Declared in `ImageCompression.h`.

`kICMImageDescriptionPropertyID_ClassicTrackHeight`
`Fixed`, Read.

Available in Mac OS X v10.3 and later.

Declared in `ImageCompression.h`.

Declared In
`ImageCompression.h`

ICMEncodedFrameSetFrameType Values

Constants passed to ICMEncodedFrameSetFrameType.

```
enum {
  kICMFrameType_I               = 'I',
  kICMFrameType_P               = 'P',
  kICMFrameType_B               = 'B',
  kICMFrameType_Unknown         = 0
};
```

Declared In
`ImageCompression.h`

ICMMultiPassStorageCreateWithTemporaryFile Values

Constants passed to ICMMultiPassStorageCreateWithTemporaryFile.

```
enum {
  /*
   * Indicates that the temporary file should not be deleted when the
   * multipass storage is released.
   */
  kICMMultiPassStorage_DoNotDeleteWhenDone = 1L << 0
};
```

Declared In
`ImageCompression.h`

ICMMultiPassStorageGetTimeStamp Values

Constants passed to ICMMultiPassStorageGetTimeStamp.

```
enum {
  /*
   * Requests the first time stamp at which a value is stored.
   */
  kICMMultiPassStorage_GetFirstTimeStamp = 1,
  /*
   * Requests the previous time stamp before the given time stamp at
   * which a value is stored.
   */
  kICMMultiPassStorage_GetPreviousTimeStamp = 2,
  /*
   * Requests the next time stamp after the given time stamp at which a
   * value is stored.
   */
  kICMMultiPassStorage_GetNextTimeStamp = 3,
  /*
   * Requests the last time stamp at which a value is stored.
   */
  kICMMultiPassStorage_GetLastTimeStamp = 4
};
```

Declared In
ImageCompression.h

kICMValidTime_DecodeDurationIsValid

Constants grouped with kICMValidTime_DecodeDurationIsValid.

```
enum {
  /*
   * Indicates that a display time stamp is valid.
   */
  kICMValidTime_DisplayTimeStampIsValid = 1L << 0,
  /*
   * Indicates that a display duration is valid.
   */
  kICMValidTime_DisplayDurationIsValid = 1L << 1,
  /*
   * Indicates that a decode time stamp is valid.
   */
  kICMValidTime_DecodeTimeStampIsValid = 1L << 2,
  /*
   * Indicates that a decode duration is valid.
   */
  kICMValidTime_DecodeDurationIsValid = 1L << 3,
  /*
   * Indicates that a display offset (the offset from a decode time
   * stamp to a display time stamp) is valid.
   */
  kICMValidTime_DisplayOffsetIsValid = 1L << 4
};
```

Constants

`kICMValidTime_DisplayTimeStampIsValid`

The time value passed in `displayTimeStamp` is valid.

Available in Mac OS X v10.3 and later.

Declared in `ImageCompression.h`.

`kICMValidTime_DisplayDurationIsValid`

The time value passed in `displayDuration` is valid.

Available in Mac OS X v10.3 and later.

Declared in `ImageCompression.h`.

Declared In

`ImageCompression.h`

Document Revision History

This table describes the changes to *Image Compression Manager Reference*.

Date	Notes
2006-05-23	New reference document that describes the API for QuickTime image compression.

Document Revision History

Index

C

D

I

K

N

Q

S