Image Codec Reference for QuickTime

QuickTime > Compression & Decompression

2006-05-23

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

Simultaneously published in the United States and Canada.

Contents

Image Codec Reference for QuickTime 7

Overview 7
Functions by Task 7
- Base Image Decompressor Functions 7
- Low-Level Effects Functions 9
- Vector Codec Component Functions 9
- Supporting Functions 10

Functions 12
- CurveAddAtomToVectorStream 12
- CurveAddPathAtomToVectorStream 13
- CurveAddZeroAtomToVectorStream 14
- CurveCountPointsInPath 14
- CurveCreateVectorStream 15
- CurveGetAtomDataFromVectorStream 16
- CurveGetLength 17
- CurveGetNearestPathPoint 17
- CurveGetPathPoint 18
- CurveInsertPointIntoPath 19
- CurveLengthToPoint 20
- CurveNewPath 21
- CurvePathPointToLength 22
- CurveSetPathPoint 23
- DisposeImageCodecDrawBandCompleteUPP 24
- DisposeImageCodecMPDrawBandUPP 24
- DisposeImageCodecTimeTriggerUPP 25
- ImageCodecBandCompress 25
- ImageCodecBandDecompress 26
- ImageCodecBeginBand 27
- ImageCodecBeginPass 28
- ImageCodecBusy 29
- ImageCodecCancelTrigger 30
- ImageCodecCompleteFrame 30
- ImageCodecCreateStandardParameterDialog 31
- ImageCodecDecodeBand 32
- ImageCodecDismissStandardParameterDialog 33
- ImageCodecDisposeImageGWorld 34
- ImageCodecDisposeMemory 34
- ImageCodecDITLEvent 35
- ImageCodecDITLInstall 36
- ImageCodecDITLItem 36
- ImageCodecDITLRemove 37

ImageCodecDITLValidateInput 38
ImageCodecDrawBand 38
ImageCodecDroppingFrame 39
ImageCodecEffectBegin 40
ImageCodecEffectCancel 40
ImageCodecEffectConvertEffectSourceToFormat 41
ImageCodecEffectDisposeSMPTEFrame 42
ImageCodecEffectGetSpeed 42
ImageCodecEffectPrepareSMPTEFrame 43
ImageCodecEffectRenderFrame 43
ImageCodecEffectRenderSMPTEFrame 44
ImageCodecEffectSetup 45
ImageCodecEncodeFrame 46
ImageCodecEndBand 46
ImageCodecExtractAndCombineFields 47
ImageCodecFlush 49
ImageCodecFlushFrame 50
ImageCodecGetBaseMPWorkFunction 50
ImageCodecGetCodecInfo 51
ImageCodecGetCompressedImageSize 52
ImageCodecGetCompressionTime 53
ImageCodecGetDecompressLatency 54
ImageCodecGetDITLForSize 55
ImageCodecGetMaxCompressionSize 56
ImageCodecGetMaxCompressionSizeWithSources 57
ImageCodecGetParameterList 58
ImageCodecGetParameterListHandle 59
ImageCodecGetSettings 59
ImageCodecGetSettingsAsText 60
ImageCodecGetSimilarity 60
ImageCodecGetSourceDataGammaLevel 61
ImageCodecHitTestData 62
ImageCodecHitTestDataWithFlags 63
ImageCodecInitialize 64
ImageCodecIsImageDescriptionEquivalent 64
ImageCodecIsStandardParameterDialogEvent 65
ImageCodecMergeFloatingImageOntoWindow 66
ImageCodecNewImageBufferMemory 67
ImageCodecNewImageGWorld 68
ImageCodecNewMemory 69
ImageCodecPreCompress 70
ImageCodecPreDecompress 70
ImageCodecPreflight 71
ImageCodecPrepareToCompressFrames 72
ImageCodecProcessBetweenPasses 73
ImageCodecQueueStarting 74

ImageCodecQueueStopping 74
ImageCodecRemoveFloatingImage 75
ImageCodecRequestGammaLevel 76
ImageCodecRequestSettings 77
ImageCodecScheduleFrame 78
ImageCodecSetSettings 78
ImageCodecSetTimeBase 79
ImageCodecSetTimeCode 80
ImageCodecSourceChanged 80
ImageCodecStandardParameterDialogDoAction 81
ImageCodecTrimImage 83
ImageCodecValidateParameters 84
NewImageCodecDrawBandCompleteUPP 85
NewImageCodecMPDrawBandUPP 85
NewImageCodecTimeTriggerUPP 86
QTPhotoDefineHuffmanTable 86
QTPhotoDefineQuantizationTable 87
QTPhotoSetRestartInterval 88
QTPhotoSetSampling 89
Callbacks 90
ComponentMPWorkFunctionProc 90
ImageCodecMPDrawBandProc 90
ImageCodecTimeTriggerProc 91
Data Types 91
CDSequenceDataSource 91
CDSequenceDataSourcePtr 94
CodecCompressParams 94
CodecDecompressParams 99
ComponentMPWorkFunctionUPP 107
EffectsFrameParams 107
EffectsFrameParamsPtr 108
EffectSource 109
EffectSourcePtr 109
gxPaths 110
gxPoint 110
ImageCodecMPDrawBandUPP 111
ImageCodecTimeTriggerUPP 111
ImageSubCodecDecompressCapabilities 112
ImageSubCodecDecompressRecord 112
QTParameterValidationOptions 114
SMPTEFlags 114
SMPTEFrameReference 115
SMPTEWipeType 115
Constants 115
Codec Properties 115
ImageSubCodecDecompressRecord Values 120

EffectSource Values 121
ImageCodecValidateParameters Values 121
CodecDecompressParams Values 121

Document Revision History 123

Index 125

Image Codec Reference for QuickTime

Framework:	Frameworks/QuickTime.framework
Declared in	Components.h ImageCodec.h

Overview

An image codec (or image compressor component) is a code resource that provides QuickTime with compression or decompression services for image data.

Functions by Task

Base Image Decompressor Functions

`ImageCodecBeginBand` (page 27)
Called before drawing a band or frame; it allows your image decompressor component to save information about a band before decompressing it.

`ImageCodecDisposeMemory` (page 34)
Disposes codec-allocated memory.

`ImageCodecDITLEvent` (page 35)
Lets an image codec component receive and process dialog events.

`ImageCodecDITLInstall` (page 36)
Installs added items in an image codec settings dialog box before the dialog box is displayed to the user.

`ImageCodecDITLItem` (page 36)
Receives and processes mouse clicks in the image codec settings dialog box.

`ImageCodecDITLRemove` (page 37)
Removes a panel from the image codec settings dialog box.

`ImageCodecDITLValidateInput` (page 38)
Validates the contents of the user dialog box for an image codec component.

`ImageCodecDrawBand` (page 38)
Decompresses a band or frame.

`ImageCodecEndBand` (page 46)
Notifies your image decompressor component that decompression of a band has finished or that it was terminated by the Image Compression Manager.

`ImageCodecExtractAndCombineFields` (page 47)
Performs field operations on video data.

`ImageCodecFlush` (page 49)
Empties an image decompressor component's input queue of pending scheduled frames.

`ImageCodecGetDITLForSize` (page 55)
Returns the size of various dialog item lists.

`ImageCodecGetMaxCompressionSizeWithSources` (page 57)
Notifies your codec when an application calls GetCSequenceMaxCompressionSize.

`ImageCodecGetSettings` (page 59)
Returns the codec settings chosen by the user.

`ImageCodecHitTestData` (page 62)
Notifies your codec when the application calls PtInDSequenceData.

`ImageCodecInitialize` (page 64)
Called before making any other all calls to your component.

`ImageCodecIsImageDescriptionEquivalent` (page 64)
Compares image descriptions.

`ImageCodecMergeFloatingImageOntoWindow` (page 66)
Draws the current contents of a floating image.

`ImageCodecNewImageBufferMemory` (page 67)
Asks a codec to allocate memory for an offscreen buffer of non-RGB pixels.

`ImageCodecNewMemory` (page 69)
Requests codec-allocated memory.

`ImageCodecPreflight` (page 71)
Called before decompressing an image, in response to an ImageCodecPreDecompress call from the Image Compression Manager.

`ImageCodecQueueStarting` (page 74)
Called by the base image decompressor before decompressing the frames in the queue if your image decompressor component supports asynchronous scheduled decompression.

`ImageCodecQueueStopping` (page 74)
Notifies your component that the frames in the queue have been decompressed, if your image decompressor component supports asynchronous scheduled decompression.

`ImageCodecRemoveFloatingImage` (page 75)
Hides an image codec's floating image without having to close the component.

`ImageCodecRequestSettings` (page 77)
Displays a dialog box containing codec-specific compression settings.

`ImageCodecSetSettings` (page 78)
Sets the settings of an optional image codec dialog box.

`ImageCodecSetTimeCode` (page 80)
Sets the timecode for the next frame that is to be decompressed.

`ImageCodecSourceChanged` (page 80)
Notifies your codec that one of the data sources has changed when an application calls CDSequenceSetSourceData or CDSequenceChangedSourceData.

Low-Level Effects Functions

`ImageCodecCreateStandardParameterDialog` (page 31)
Creates a parameters dialog box for a specified effect.

`ImageCodecDismissStandardParameterDialog` (page 33)
Retrieves values from a standard parameter dialog box created by the low-level ImageCodecCreateStandardParameterDialog function, then closes the dialog box.

`ImageCodecGetParameterList` (page 58)
Returns a parameter description atom container for a specified effect component instance.

`ImageCodecIsStandardParameterDialogEvent` (page 65)
Processes events related to a standard parameters dialog box created by ImageCodecCreateStandardParameterDialog.

`ImageCodecStandardParameterDialogDoAction` (page 81)
Allows you to control the behavior of a standard parameter dialog box created by ImageCodecCreateStandardParameterDialog.

Vector Codec Component Functions

`CurveAddAtomToVectorStream` (page 12)
Adds an atom to a vector data stream.

`CurveAddPathAtomToVectorStream` (page 13)
Adds a path to a vector data stream.

`CurveAddZeroAtomToVectorStream` (page 14)
Adds a kCurveEndAtom to a vector data stream; this atom marks the end of the vector data stream,

`CurveCountPointsInPath` (page 14)
Counts the points along either one of a path's contours or all of its contours.

`CurveCreateVectorStream` (page 15)
Creates a new, empty vector data stream.

`CurveGetAtomDataFromVectorStream` (page 16)
Finds the first atom of a specified type within a vector data stream and get its data.

`CurveGetLength` (page 17)
Calculates the length of one of a path's contours or the sum of the lengths of all of its contours.

`CurveGetNearestPathPoint` (page 17)
Finds the closest point on a path to a specified point.

`CurveGetPathPoint` (page 18)
Obtains a point from a path and to find out if the point is on the curve.

`CurveInsertPointIntoPath` (page 19)
Adds a new point to a path.

`CurveLengthToPoint` (page 20)
Obtains the point at a specified distance along a curve.

`CurveNewPath` (page 21)
Creates a new path.

`CurvePathPointToLength` (page 22)
Obtains the length of a path between specified starting and ending distances that is nearest a point.

`CurveSetPathPoint` (page 23)
Changes the location of a point in a path.

Supporting Functions

`DisposeImageCodecDrawBandCompleteUPP` (page 24)
Disposes of an ImageCodecDrawBandCompleteUPP pointer.

`DisposeImageCodecMPDrawBandUPP` (page 24)
Disposes of an ImageCodecMPDrawBandUPP pointer.

`DisposeImageCodecTimeTriggerUPP` (page 25)
Disposes of an ImageCodecTimeTriggerUPP pointer.

`ImageCodecBandCompress` (page 25)
Asks your component to compress an image or a band of an image.

`ImageCodecBandDecompress` (page 26)
Asks your component to decompress a frame.

`ImageCodecBeginPass` (page 28)
Notifies the compressor that it should operate in multipass mode and use the given multipass storage.

`ImageCodecBusy` (page 29)
Lets your component report whether it is performing an asynchronous operation.

`ImageCodecCancelTrigger` (page 30)
Cancels an image codec's ImageCodecTimeTriggerProc callback.

`ImageCodecCompleteFrame` (page 30)
Directs the compressor to finish with a queued source frame, either emitting or dropping it.

`ImageCodecDecodeBand` (page 32)
Returns an ImageSubCodecDecompressRecord structure for an image codec component.

`ImageCodecDisposeImageGWorld` (page 34)
Disposes of an image graphics world associated with an image codec.

`ImageCodecDroppingFrame` (page 39)
Undocumented

`ImageCodecEffectBegin` (page 40)
Undocumented

`ImageCodecEffectCancel` (page 40)
Undocumented

`ImageCodecEffectConvertEffectSourceToFormat` (page 41)
Undocumented

`ImageCodecEffectDisposeSMPTEFrame` (page 42)
Undocumented

`ImageCodecEffectGetSpeed` (page 42)
Undocumented

`ImageCodecEffectPrepareSMPTEFrame` (page 43)
Undocumented

`ImageCodecEffectRenderFrame` (page 43)
Undocumented

ImageCodecEffectRenderSMPTEFrame (page 44)
Undocumented

ImageCodecEffectSetup (page 45)
Undocumented

ImageCodecEncodeFrame (page 46)
Presents the compressor with a frame to encode.

ImageCodecFlushFrame (page 50)
Undocumented

ImageCodecGetBaseMPWorkFunction (page 50)
Gets an image codec's ComponentMPWorkFunctionProc callback.

ImageCodecGetCodecInfo (page 51)
Notifies your codec whenever an application calls GetCodecInfo.

ImageCodecGetCompressedImageSize (page 52)
Notifies your codec whenever an application calls GetCompressedImageSize.

ImageCodecGetCompressionTime (page 53)
Notifies your codec whenever an application calls GetCompressionTime.

ImageCodecGetDecompressLatency (page 54)
Retrieves the video latency value from a specified video codec.

ImageCodecGetMaxCompressionSize (page 56)
Notifies your codec whenever an application calls GetMaxCompressionSize.

ImageCodecGetParameterListHandle (page 59)
Returns a handle to a Mac OS resource of type 'atms'.

ImageCodecGetSettingsAsText (page 60)
Undocumented

ImageCodecGetSimilarity (page 60)
Notifies your codec when an application calls GetSimilarity.

ImageCodecGetSourceDataGammaLevel (page 61)
Returns the native gamma of compressed data, if any.

ImageCodecHitTestDataWithFlags (page 63)
Undocumented

ImageCodecNewImageGWorld (page 68)
Undocumented

ImageCodecPreCompress (page 70)
Notifies your component before compressing an image or a band of an image.

ImageCodecPreDecompress (page 70)
Notifies your component before decompressing an image or sequence of frames.

ImageCodecPrepareToCompressFrames (page 72)
Prepares the compressor to receive frames.

ImageCodecProcessBetweenPasses (page 73)
Provides the compressor with an opportunity to perform processing between passes.

ImageCodecRequestGammaLevel (page 76)
Asks an image codec to convert from source to destination gamma levels.

ImageCodecScheduleFrame (page 78)
Undocumented

`ImageCodecSetTimeBase` (page 79)
Sets the time base for an image codec component.

`ImageCodecTrimImage` (page 83)
Notifies your component whenever an application calls TrimImage.

`ImageCodecValidateParameters` (page 84)
Validates effect parameters.

`NewImageCodecDrawBandCompleteUPP` (page 85)
Allocates a Universal Procedure Pointer for an ImageCodecDrawBandCompleteProc callback.

`NewImageCodecMPDrawBandUPP` (page 85)
Allocates a Universal Procedure Pointer for the ImageCodecMPDrawBandProc callback.

`NewImageCodecTimeTriggerUPP` (page 86)
Allocates a Universal Procedure Pointer for the ImageCodecTimeTriggerProc callback.

`QTPhotoDefineHuffmanTable` (page 86)
Defines a Huffman table.

`QTPhotoDefineQuantizationTable` (page 87)
Specifies a custom quantization table.

`QTPhotoSetRestartInterval` (page 88)
Specifies the restart interval to use in future JPEG compression operations.

`QTPhotoSetSampling` (page 89)
Specifies the chrominance downsampling ratio to use in future JPEG compression operations.

Functions

CurveAddAtomToVectorStream

Adds an atom to a vector data stream.

```
ComponentResult CurveAddAtomToVectorStream (
   ComponentInstance effect,
   OSType atomType,
   Size atomSize,
   void *pAtomData,
   Handle vectorStream
);
```

Parameters

effect
The instance of the QuickTime vector codec component for the request. Your software obtains this reference when calling the Component Manager's `OpenComponent` or `OpenDefaultComponent` function.

atomType
The type of atom to add to the vector data stream.

atomSize
The size of the data for the atom.

pAtomData
A pointer to the data for the atom.

vectorStream
A handle to the vector data stream to which to add the atom.

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

Discussion
This function adds the atom to the end of the specified vector data stream and resizes the vector data stream handle.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Related Sample Code
qtvectors
qtvectors.win

Declared In
`ImageCodec.h`

CurveAddPathAtomToVectorStream

Adds a path to a vector data stream.

```
ComponentResult CurveAddPathAtomToVectorStream (
   ComponentInstance effect,
   Handle pathData,
   Handle vectorStream
);
```

Parameters

effect
The instance of the QuickTime vector codec component for the request. Your software obtains this reference when calling the Component Manager's `OpenComponent` or `OpenDefaultComponent` function.

pathData
A handle to the data for the path.

vectorStream
A handle to the vector data stream to which to add the path.

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

Discussion
This function adds the path to the end of the specified vector data stream and resizes the vector data stream handle.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Related Sample Code
qtvectors
qtvectors.win

Declared In
`ImageCodec.h`

CurveAddZeroAtomToVectorStream

Adds a kCurveEndAtom to a vector data stream; this atom marks the end of the vector data stream,

```
ComponentResult CurveAddZeroAtomToVectorStream (
   ComponentInstance effect,
   Handle vectorStream
);
```

Parameters

effect
: The instance of the QuickTime vector codec component for the request. Your software obtains this reference when calling the Component Manager's `OpenComponent` or `OpenDefaultComponent` function.

vectorStream
: A handle to the vector data stream to which to add the `kCurveEndAtom` atom.

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

Discussion
This function adds a `kCurveEndAtom` atom to the end of the specified vector data stream and resizes the vector data stream handle. The `kCurveEndAtom` atom is required at the end of a vector data stream, and there may be only one `kCurveEndAtom` atom in the stream.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Related Sample Code
qtvectors
qtvectors.win

Declared In
`ImageCodec.h`

CurveCountPointsInPath

Counts the points along either one of a path's contours or all of its contours.

```
ComponentResult CurveCountPointsInPath (
   ComponentInstance effect,
   gxPaths *aPath,
   unsigned long contourIndex,
   unsigned long *pCount
);
```

Parameters

effect

The instance of the QuickTime vector codec component for the request.

aPath

A pointer to the path.

contourIndex

The index of the contour to be counted.

pCount

A pointer to a field that is to receive the number of points in the contour or path.

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

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`ImageCodec.h`

CurveCreateVectorStream

Creates a new, empty vector data stream.

```
ComponentResult CurveCreateVectorStream (
   ComponentInstance effect,
   Handle *pStream
);
```

Parameters

effect

The instance of the QuickTime vector codec component for the request. Your software obtains this reference when calling the Component Manager's `OpenComponent` or `OpenDefaultComponent` function.

pStream

A pointer to the handle that is to receive the newly created vector data stream.

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

Discussion
The caller is responsible for disposing of the stream when finished with it. This can be done by calling `DisposeHandle`.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Related Sample Code
qtvectors
qtvectors.win

Declared In
`ImageCodec.h`

CurveGetAtomDataFromVectorStream

Finds the first atom of a specified type within a vector data stream and get its data.

```
ComponentResult CurveGetAtomDataFromVectorStream (
   ComponentInstance effect,
   Handle vectorStream,
   long atomType,
   long *dataSize,
   Ptr *dataPtr
);
```

Parameters

effect
: The instance of the QuickTime vector codec component for the request. Your software obtains this reference when calling the Component Manager's `OpenComponent` or `OpenDefaultComponent` function.

vectorStream
: A handle to the vector data stream from which to get the atom.

atomType
: The type of atom to find.

dataSize
: A pointer to a field that is to receive the size of the atom's data.

dataPtr
: A pointer to a pointer to a field that is to receive the atom's data.

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

Discussion
Before calling this function, your software must lock the handle for the vector data stream (with Macintosh, by calling `HLock`). This prevents the handle from being moved, which could invalidate the pointer to the atom data before your software gets the data.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`ImageCodec.h`

CurveGetLength

Calculates the length of one of a path's contours or the sum of the lengths of all of its contours.

```
ComponentResult CurveGetLength (
   ComponentInstance effect,
   gxPaths *target,
   long index,
   wide *wideLength
);
```

Parameters

effect

The instance of the QuickTime vector codec component for the request. Your software obtains this reference when calling the Component Manager's `OpenComponent` or `OpenDefaultComponent` function.

target

A pointer to the path.

index

Contains the index of the contour whose length is to be calculated or, if the value is 0, specifies to calculate the lengths of all of the path's contours and return the sum of the lengths.

wideLength

A pointer to a field that is to receive the length.

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

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`ImageCodec.h`

CurveGetNearestPathPoint

Finds the closest point on a path to a specified point.

```
ComponentResult CurveGetNearestPathPoint (
   ComponentInstance effect,
   gxPaths *aPath,
   FixedPoint *thePoint,
   unsigned long *contourIndex,
   unsigned long *pointIndex,
   Fixed *theDelta
);
```

Parameters

effect

The instance of the QuickTime vector codec component for the request. Your software obtains this reference when calling the Component Manager's `OpenComponent` or `OpenDefaultComponent` function.

aPath

A pointer to the path.

thePoint

A pointer to a point for which to find the closest point on the path.

contourIndex

A pointer to a field that is to receive the index of the contour that contains the closest point.

pointIndex

A pointer to a field that is to receive the index of the closest point.

theDelta

A pointer to a field that is to receive the distance between the specified point and the closest point in the contour to it.

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

Discussion
In programs where users directly manipulate curves, you can use this function to determine the closest control point to a given point.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`ImageCodec.h`

CurveGetPathPoint

Obtains a point from a path and to find out if the point is on the curve.

```
ComponentResult CurveGetPathPoint (
   ComponentInstance effect,
   gxPaths *aPath,
   unsigned long contourIndex,
   unsigned long pointIndex,
   gxPoint *thePoint,
   Boolean *ptIsOnPath
);
```

Parameters

effect

The instance of the QuickTime vector codec component for the request. Your software obtains this reference when calling the Component Manager's `OpenComponent` or `OpenDefaultComponent` function.

aPath

A pointer to the path.

contourIndex

The index of the contour from which to get the point.

pointIndex

The index of the point to get.

thePoint

A pointer to a field that is to receive the point.

ptIsOnPath

A pointer to a field that is to receive a Boolean value that, if TRUE, specifies that the point is on the curve.

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

Discussion
This function lets programs get a single point from a path without walking the path data.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`ImageCodec.h`

CurveInsertPointIntoPath

Adds a new point to a path.

```
ComponentResult CurveInsertPointIntoPath (
   ComponentInstance effect,
   gxPoint *aPoint,
   Handle thePath,
   unsigned long contourIndex,
   unsigned long pointIndex,
   Boolean ptIsOnPath
);
```

Parameters

effect

The instance of the QuickTime vector codec component for the request. Your software obtains this reference when calling the Component Manager's `OpenComponent` or `OpenDefaultComponent` function.

aPoint

A pointer to the point to add to the path.

thePath

A handle to the path to which to add the point.

contourIndex

The index of the path contour to which to add the point.

pointIndex

The index of the point to add.

ptIsOnPath

If TRUE, specifies that the new point is to be on the path.

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

Discussion
This function is best for adding a single point to a path rather than large numbers of points.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Related Sample Code
qtvectors
qtvectors.win

Declared In
`ImageCodec.h`

CurveLengthToPoint

Obtains the point at a specified distance along a curve.

```
ComponentResult CurveLengthToPoint (
   ComponentInstance effect,
   gxPaths *target,
   long index,
   Fixed length,
   FixedPoint *location,
   FixedPoint *tangent
);
```

Parameters

effect

The instance of the QuickTime vector codec component for the request. Your software obtains this reference when calling the Component Manager's `OpenComponent` or `OpenDefaultComponent` function.

target

A pointer to the path.

index

The index of the path contour from which to get the point.

length

The distance along the curve at which to find the point.

location

A pointer to a field that is to receive the point.

tangent

A pointer to a field that is to receive a point that is tangent to the point at the specified distance.

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

Discussion
This function is useful for converting a value to a point, such as when creating an animation that follows a curve.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`ImageCodec.h`

CurveNewPath

Creates a new path.

```
ComponentResult CurveNewPath (
   ComponentInstance effect,
   Handle *pPath
);
```

Parameters

effect

The instance of the QuickTime vector codec component for the request. Your software obtains this reference when calling the Component Manager's `OpenComponent` or `OpenDefaultComponent` function.

pPath

A pointer to a handle that is to receive the new path.

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

Discussion
The path created by this function contains one contour and no points. The caller must dispose of the handle when it is finished with it (with Macintosh, by calling `DisposeHandle`).

Version Notes
Introduced in QuickTime 3 or earlier.

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

Related Sample Code
qtvectors
qtvectors.win

Declared In
`ImageCodec.h`

CurvePathPointToLength

Obtains the length of a path between specified starting and ending distances that is nearest a point.

```
ComponentResult CurvePathPointToLength (
   ComponentInstance ci,
   gxPaths *aPath,
   Fixed startDist,
   Fixed endDist,
   FixedPoint *thePoint,
   Fixed *pLength
);
```

Parameters

ci

The instance of the QuickTime vector codec component for the request. Your software obtains this reference when calling the Component Manager's `OpenComponent` or `OpenDefaultComponent` function.

aPath

A pointer to the path.

startDist
The distance along the path at which to start measuring the path's length.

endDist
The distance along the path at which to stop measuring the path's length.

thePoint
A pointer to a point; the function measures the path closest to this point.

pLength
A pointer to a field that is to receive the length of the specified part of the path.

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

Discussion
You can use this function to test if the user has clicked on the specified portion of the curve.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`ImageCodec.h`

CurveSetPathPoint

Changes the location of a point in a path.

```
ComponentResult CurveSetPathPoint (
   ComponentInstance effect,
   gxPaths *aPath,
   unsigned long contourIndex,
   unsigned long pointIndex,
   gxPoint *thePoint,
   Boolean ptIsOnPath
);
```

Parameters

effect
The instance of the QuickTime vector codec component for the request. Your software obtains this reference when calling the Component Manager's `OpenComponent` or `OpenDefaultComponent` function.

aPath
A pointer to the path.

contourIndex
The index of the path contour that contains the point to change.

pointIndex
The index of the point to change.

thePoint
A pointer to the new value for the point.

ptIsOnPath

If TRUE, specifies that the new point is to be on the path.

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

Discussion
This function edits an existing point location within a path. The function that you call to send the interpolated value to the receiving track is defined as a universal procedure in systems that support the Macintosh Code Fragment Manager (CFM) or is defined as a data procedure for non-CFM systems. With Macintosh, the `TweenerDataUPP` function pointer specifies the function the tween component calls with the value generated by the tween operation. A tween component calls this function from its implementation of the `TweenerDoTween` function. You call this function by invoking the function specified in the tween record's `dataProc` field.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`ImageCodec.h`

DisposeImageCodecDrawBandCompleteUPP

Disposes of an ImageCodecDrawBandCompleteUPP pointer.

```
void DisposeImageCodecDrawBandCompleteUPP (
   ImageCodecDrawBandCompleteUPP userUPP
);
```

Parameters

userUPP

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

Version Notes
Introduced in QuickTime 5.

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

Declared In
`ImageCodec.h`

DisposeImageCodecMPDrawBandUPP

Disposes of an ImageCodecMPDrawBandUPP pointer.

```
void DisposeImageCodecMPDrawBandUPP (
   ImageCodecMPDrawBandUPP userUPP
);
```

Parameters

userUPP

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

Version Notes
Introduced in QuickTime 4.1.

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

Related Sample Code
ElectricImageComponent
ElectricImageComponent.win
OpenGLCompositorLab
SoftVideoOutputComponent

Declared In
`ImageCodec.h`

DisposeImageCodecTimeTriggerUPP

Disposes of an ImageCodecTimeTriggerUPP pointer.

```
void DisposeImageCodecTimeTriggerUPP (
   ImageCodecTimeTriggerUPP userUPP
);
```

Parameters

userUPP

An `ImageCodecTimeTriggerUPP` 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
`ImageCodec.h`

ImageCodecBandCompress

Asks your component to compress an image or a band of an image.

```
ComponentResult ImageCodecBandCompress (
   ComponentInstance ci,
   CodecCompressParams *params
);
```

Parameters

ci

An image decompressor component instance. Your software obtains this reference from `OpenComponent` or `OpenDefaultComponent`.

params

A pointer to a `CodecCompressParams` structure. The Image Compression Manager places the appropriate parameter information in that structure.

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

Discussion
The image being compressed may be part of a sequence.

Special Considerations

Only compressors receive this request.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`ImageCodec.h`

ImageCodecBandDecompress

Asks your component to decompress a frame.

```
ComponentResult ImageCodecBandDecompress (
   ComponentInstance ci,
   CodecDecompressParams *params
);
```

Parameters

ci

An image decompressor component instance. Your software obtains this reference from `OpenComponent` or `OpenDefaultComponent`.

params

A pointer to a `CodecDecompressParams` structure. The Image Compression Manager places the appropriate parameter information in that structure.

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

Discussion
For scheduled asynchronous decompression operations, the Image Compression Manager supplies a reference to an `ICMFrameTimeRecord` structure in this function's `CodecDecompressParams` structure parameter. The `ICMFrameTimeRecord` structure contains time information governing the scheduled decompression operation, including the time at which the frame must be displayed. For synchronous or immediate asynchronous decompress operations, the frame time is set to `NIL`.

When your component has finished the decompression operation, it must call the completion function. In the past, for asynchronous operations, your component called that function directly. For scheduled asynchronous decompression operations, your component should call `ICMDecompressComplete`.

If your component sets the `codecCanAsyncWhen` flag in predecompress but cannot support scheduled asynchronous decompression for a given frame, it must return an error code of `codecCantWhenErr`. If your component's queue is full, it should return an error code of `codecCantQueueErr`. Only decompressors receive these requests.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`ImageCodec.h`

ImageCodecBeginBand

Called before drawing a band or frame; it allows your image decompressor component to save information about a band before decompressing it.

```
ComponentResult ImageCodecBeginBand (
   ComponentInstance ci,
   CodecDecompressParams *params,
   ImageSubCodecDecompressRecord *drp,
   long flags
);
```

Parameters

ci
: An image codec component instance. Your software obtains this reference from `OpenComponent` or `OpenDefaultComponent`.

params
: A pointer to a `CodecDecompressParams` structure.

drp
: A pointer to an `ImageSubCodecDecompressRecord` structure.

flags
: Currently unused; set to 0.

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

Discussion
Your image decompressor component receives the address of the destination pixel map in the `baseAddr` field of the `drp` parameter. This address includes an adjustment for the offset. Note that if the bit depth of the pixel map is less than 8, your image decompressor component must adjust for the bit offset.

The `codecData` field of the `drp` parameter contains a pointer to the compressed video data. The `userDecompressRecord` field of the `drp` parameter contains a pointer to storage for the decompression operation. The storage is allocated by the base image decompressor after it calls the `ImageCodecInitialize` (page 64) function. The size of the storage is determined by the `decompressRecordSize` field of the `ImageSubCodecDecompressCapabilities` structure that is returned by `ImageCodecInitialize` (page 64). Your image decompressor component should use this storage to store any additional information needed about the frame in order to decompress it.

Changes your image decompressor component makes to the `ImageSubCodecDecompressRecord` or `CodecDecompressParams` structures are preserved by the base image decompressor. For example, if your component does not need to decompress all of the data, it can change the pointer to the data to be decompressed that is stored in the `codecData` field of the `ImageSubCodecDecompressRecord` structure.

Special Considerations

Your component must implement this function. Also, the base image decompressor never calls `ImageCodecBeginBand` at interrupt time. If your component supports asynchronous scheduled decompression, it may receive more than one `ImageCodecBeginBand` call before receiving an `ImageCodecDrawBand` (page 38) call.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`ImageCodec.h`

ImageCodecBeginPass

Notifies the compressor that it should operate in multipass mode and use the given multipass storage.

```
ComponentResult ImageCodecBeginPass (
   ComponentInstance ci,
   ICMCompressionPassModeFlags passModeFlags,
   UInt32 flags,
   ICMMultiPassStorageRef multiPassStorage
);
```

Parameters

ci

A component instance that identifies a connection to an image codec component.

passModeFlags

Indicates how the compressor should operate in this pass. If the `kICMCompressionPassMode_WriteToMultiPassStorage` flag is set, the compressor may gather information of interest and store it in `multiPassStorage`. If the `kICMCompressionPassMode_ReadFromMultiPassStorage` flag is set, the compressor may retrieve information from `multiPassStorage`. If the `kICMCompressionPassMode_OutputEncodedFrames` flag is set, the compressor must encode or drop every frame by calling `ICMCompressorSessionDropFrame` or `ICMCompressorSessionEmitEncodedFrame`. If that flag is not set, the compressor should not call these routines.

flags

Reserved. Ignore this parameter.

multiPassStorage

The multipass storage object that the compressor should use to store and retrieve information between passes.

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

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

Declared In
`ImageCodec.h`

ImageCodecBusy

Lets your component report whether it is performing an asynchronous operation.

```
ComponentResult ImageCodecBusy (
   ComponentInstance ci,
   ImageSequence seq
);
```

Parameters

ci

An image decompressor component instance. Your software obtains this reference from `OpenComponent` or `OpenDefaultComponent`.

seq

The unique sequence identifier assigned by `CompressSequenceBegin` or `DecompressSequenceBegin`.

Return Value
Your component should return a result code value of 1 if an asynchronous operation is in progress; it should return a result code value of 0 if the component is not performing an asynchronous operation. You can indicate an error by returning a negative result code. See `Error Codes`.

Special Considerations

Both compressors and decompressors may receive this request.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`ImageCodec.h`

ImageCodecCancelTrigger

Cancels an image codec's ImageCodecTimeTriggerProc callback.

```
ComponentResult ImageCodecCancelTrigger (
   ComponentInstance ci
);
```

Parameters

ci

An image codec component instance. Your software obtains this reference from `OpenComponent` or `OpenDefaultComponent`.

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

Version Notes
Introduced in QuickTime 4.

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

Declared In
`ImageCodec.h`

ImageCodecCompleteFrame

Directs the compressor to finish with a queued source frame, either emitting or dropping it.

```
ComponentResult ImageCodecCompleteFrame (
   ComponentInstance ci,
   ICMCompressorSourceFrameRef sourceFrame,
   UInt32 flags
);
```

Parameters

ci

A component instance that identifies a connection to an image codec component.

sourceFrame

The source frame that must be completed.

flags

Reserved; ignore.

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

Discussion
This frame does not necessarily need to be the first or only source frame emitted or dropped during this call, but the compressor must call either `ICMCompressorSessionDropFrame` or `ICMCompressorSessionEmitEncodedFrame` with this frame before returning. The ICM will call this function to force frames to be encoded for the following reasons: (a) the maximum frame delay count or maximum frame delay time in the `compressionSessionOptions` does not permit frames to be queued; (b) the client has called `ICMCompressionSessionCompleteFrames`.

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

Declared In
`ImageCodec.h`

ImageCodecCreateStandardParameterDialog

Creates a parameters dialog box for a specified effect.

```
ComponentResult ImageCodecCreateStandardParameterDialog (
   ComponentInstance ci,
   QTAtomContainer parameterDescription,
   QTAtomContainer parameters,
   QTParameterDialogOptions dialogOptions,
   DialogPtr existingDialog,
   short existingUserItem,
   QTParameterDialog *createdDialog
);
```

Parameters

ci

An effects component instance. Your software obtains this reference from `OpenComponent` or `OpenDefaultComponent`. The dialog box that is created will allow the user to specify the parameters of this effect.

parameterDescription

The parameter description atom container for this effect. You can obtain a valid parameter description by calling `ImageCodecGetParameterList` (page 58). A parameter is optionally tweenable if defined as `kAtomInterpolateIsOptional` in its parameter description atom.

parameters

The atom container that will receive the user's chosen parameter values once the dialog has been dismissed.

dialogOptions

Controls how parameters containing tween data are presented in the created dialog box. If `dialogOptions` contains 0, two values are collected for each parameter that can be tweened, and the usual tweening operation will be performed for the duration of the effect being controlled. For other values,. See these constants:

`pdOptionsCollectOneValue`

`pdOptionsAllowOptionalInterpolations`

existingDialog

An existing dialog box that will have the controls from the standard parameters dialog box added to it. Set this parameter to `NIL` if you want this function to create a stand-alone dialog box.

existingUserItem

The number of the user item in the existing dialog box that should be replaced with controls from the standard parameter dialog box. You should only pass a value to this parameter if the `existingDialog` parameter is not `NIL`.

createdDialog

On return, a reference to the dialog created and displayed by the function. This reference is required by several other low-level effects functions. It will contain a valid dialog identifier even if you requested that the controls from the standard parameter dialog box be incorporated into an existing dialog box.

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

Discussion
This is a low-level function that can be used to create a standard parameter dialog box for a specified effect, allowing the user to set the parameter values for the effect. You can optionally request that the controls from the dialog box be included within a dialog box of the calling application. The following sample code shows how to create a standard parameter dialog box and add effects controls:

```
// ImageCodecCreateStandardParameterDialog coding example
// See "Discovering QuickTime," page 303
pMovableModalDialog =GetNewDialog(kExtraDialogID, NIL, (WindowRef)-1);
if (pMovableModalDialog !=NIL) {
    ImageCodecCreateStandardParameterDialog(
        compInstance,
        qtacParameterDescription,
        qtacEffectSample,
        pdOptionsModelDialogBox,
        pMovableModalDialog,
        kExtraUserItemID,
        &lCreatedDialogID);
    ShowWindow(pMovableModalDialog);
    SelectWindow(pMovableModalDialog);
    SetPort(pMovableModalDialog);
}
```

Version Notes
Introduced in QuickTime 3 or earlier.

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

Related Sample Code
qtshoweffect

qtshoweffect.win

Declared In
`ImageCodec.h`

ImageCodecDecodeBand

Returns an ImageSubCodecDecompressRecord structure for an image codec component.

```
ComponentResult ImageCodecDecodeBand (
   ComponentInstance ci,
   ImageSubCodecDecompressRecord *drp,
   unsigned long flags
);
```

Parameters

ci

A component instance that identifies a connection to an image codec component.

drp

A pointer to an `ImageSubCodecDecompressRecord` structure.

flags

Not used; set to 0.

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

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

Declared In
`ImageCodec.h`

ImageCodecDismissStandardParameterDialog

Retrieves values from a standard parameter dialog box created by the low-level ImageCodecCreateStandardParameterDialog function, then closes the dialog box.

```
ComponentResult ImageCodecDismissStandardParameterDialog (
   ComponentInstance ci,
   QTParameterDialog createdDialog
);
```

Parameters

ci

An effect component instance. Your software obtains this reference from `OpenComponent` or `OpenDefaultComponent`. This must be the instance passed to `ImageCodecCreateStandardParameterDialog` (page 31) to create the dialog box.

createdDialog

A reference to the dialog box created by the call to `ImageCodecCreateStandardParameterDialog`.

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

Discussion
This function should be called after the `ImageCodecIsStandardParameterDialogEvent` (page 65) function returns `codecParameterDialogConfirm` or `userCanceledErr`, which indicate that the user has dismissed the dialog box. The function dismisses the dialog box, deallocating any memory allocated during the call to `ImageCodecCreateStandardParameterDialog` (page 31).

Version Notes
Introduced in QuickTime 3 or earlier.

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

Related Sample Code
qtshoweffect
qtshoweffect.win

Declared In
`ImageCodec.h`

ImageCodecDisposeImageGWorld

Disposes of an image graphics world associated with an image codec.

```
ComponentResult ImageCodecDisposeImageGWorld (
   ComponentInstance ci,
   GWorldPtr theGW
);
```

Parameters

ci

An image codec component instance. Your software obtains this reference from `OpenComponent` or `OpenDefaultComponent`.

theGW

A pointer to a `CGrafPort` structure.

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

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`ImageCodec.h`

ImageCodecDisposeMemory

Disposes codec-allocated memory.

```
ComponentResult ImageCodecDisposeMemory (
   ComponentInstance ci,
   Ptr data
);
```

Parameters

ci

An image compressor component instance. Your software obtains this reference from `OpenComponent` or `OpenDefaultComponent`.

data

A pointer to the previously allocated memory block.

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

Discussion
Your component receives the `ImageCodecDisposeMemory` request whenever an application calls `CDSequenceDisposeMemory`.

Special Considerations

When a codec instance is closed, it must ensure that all blocks allocated by that instance are disposed and call `ICMMemoryDisposedProc`.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`ImageCodec.h`

ImageCodecDITLEvent

Lets an image codec component receive and process dialog events.

```
ComponentResult ImageCodecDITLEvent (
   ComponentInstance ci,
   DialogRef d,
   short itemOffset,
   const EventRecord *theEvent,
   short *itemHit,
   Boolean *handled
);
```

Parameters

ci
: The component instance that identifies your connection to an image codec component.

d
: A dialog reference identifying the settings dialog box.

itemOffset
: The offset to your panel's first item in the dialog box.

theEvent
: A pointer to an `EventRecord` structure. This structure contains information identifying the nature of the event.

itemHit
: A pointer to a field that is to receive the item number in cases where your component handles the event. The number returned is an absolute, not a relative number, so it must be offset by the `itemOffset` parameter handled.

handled
: A pointer to a Boolean value. Set this Boolean value to TRUE if you handle the event; set it to FALSE if you do not.

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

Version Notes
Introduced in QuickTime 6.

Availability
Available in Mac OS X v10.2 and later.

Declared In
`ImageCodec.h`

ImageCodecDITLInstall

Installs added items in an image codec settings dialog box before the dialog box is displayed to the user.

```
ComponentResult ImageCodecDITLInstall (
   ComponentInstance ci,
   DialogRef d,
   short itemOffset
);
```

Parameters

ci
The component instance that identifies your connection to an image codec component.

d
A pointer to the dialog box to be installed.

itemOffset
The offset to your image codec's first dialog item.

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

Version Notes
Introduced in QuickTime 6.

Availability
Available in Mac OS X v10.2 and later.

Declared In
`ImageCodec.h`

ImageCodecDITLItem

Receives and processes mouse clicks in the image codec settings dialog box.

```
ComponentResult ImageCodecDITLItem (
   ComponentInstance ci,
   DialogRef d,
   short itemOffset,
   short itemNum
);
```

Parameters

ci

The component instance that identifies your connection to an image codec component.

d

A dialog reference identifying the settings dialog box.

itemOffset

The offset to your panel's first item in the dialog box.

itemNum

The item number of the dialog item selected by the user. The sequence grabber provides an absolute item number. It is your responsibility to adjust this value to account for the offset to your panel's first item in the dialog box.

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

Discussion
An image codec component calls this function whenever the user clicks an item in the settings dialog box. Your component may then perform whatever processing is appropriate, depending upon the item number.

Version Notes
Introduced in QuickTime 6.

Availability
Available in Mac OS X v10.2 and later.

Declared In
`ImageCodec.h`

ImageCodecDITLRemove

Removes a panel from the image codec settings dialog box.

```
ComponentResult ImageCodecDITLRemove (
   ComponentInstance ci,
   DialogRef d,
   short itemOffset
);
```

Parameters

ci

The component instance that identifies your connection to an image codec component.

d

A dialog pointer identifying the settings dialog box.

itemOffset

The offset to your panel's first item in the dialog box.

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

Discussion
An image codec component calls this function just before removing your items from the settings dialog box.

Version Notes
Introduced in QuickTime 6.

Availability
Available in Mac OS X v10.2 and later.

Declared In
`ImageCodec.h`

ImageCodecDITLValidateInput

Validates the contents of the user dialog box for an image codec component.

```
ComponentResult ImageCodecDITLValidateInput (
   ComponentInstance ci,
   Boolean *ok
);
```

Parameters

ci

The component instance that identifies your connection to an image codec component.

ok

A pointer to a Boolean value. Set this value to TRUE if the settings are OK; otherwise, set it to FALSE.

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

Discussion
The image codec calls this function when the user clicks the OK button. If the user clicks the Cancel button, the image codec does not call this function. You indicate whether the settings are acceptable by setting the Boolean value pointed to by the `ok` parameter. If you set this value to FALSE, the sequence grabber component ignores the OK button in the dialog box.

Version Notes
Introduced in QuickTime 6.

Availability
Available in Mac OS X v10.2 and later.

Declared In
`ImageCodec.h`

ImageCodecDrawBand

Decompresses a band or frame.

```
ComponentResult ImageCodecDrawBand (
   ComponentInstance ci,
   ImageSubCodecDecompressRecord *drp
);
```

Parameters

ci

An image decompressor component instance. Your software obtains this reference from `OpenComponent` or `OpenDefaultComponent`.

drp

A pointer to an `ImageSubCodecDecompressRecord` structure.

Return Value

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

Discussion

When the base image decompressor calls your image decompressor component's `ImageCodecDrawBand` function, your component must perform the decompression specified by the fields of the `ImageSubCodecDecompressRecord` structure. The structure includes any changes your component made to it when performing the `ImageCodecBeginBand` (page 27) function. If your component supports asynchronous scheduled decompression, it may receive more than one `ImageCodecBeginBand` call before receiving an `ImageCodecDrawBand` call.

Special Considerations

Your component must implement this function. If the `ImageSubCodecDecompressRecord` structure specifies a progress function or data-loading function, the base image decompressor never calls this function at interrupt time. If not, the base image decompressor may call this function at interrupt time.

Version Notes

Introduced in QuickTime 3 or earlier.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`ImageCodec.h`

ImageCodecDroppingFrame

Undocumented

```
ComponentResult ImageCodecDroppingFrame (
   ComponentInstance ci,
   const ImageSubCodecDecompressRecord *drp
);
```

Parameters

ci

An image codec component instance. Your software obtains this reference from `OpenComponent` or `OpenDefaultComponent`.

drp

A pointer to an `ImageSubCodecDecompressRecord` structure.

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

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`ImageCodec.h`

ImageCodecEffectBegin

Undocumented

```
ComponentResult ImageCodecEffectBegin (
   ComponentInstance effect,
   CodecDecompressParams *p,
   EffectsFrameParamsPtr ePtr
);
```

Parameters

effect

An effect component instance. Your software obtains this reference from `OpenComponent` or `OpenDefaultComponent`.

p

A pointer to a `CodecDecompressParams` structure.

ePtr

A pointer to a `EffectsFrameParams` structure.

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

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`ImageCodec.h`

ImageCodecEffectCancel

Undocumented

```
ComponentResult ImageCodecEffectCancel (
   ComponentInstance effect,
   EffectsFrameParamsPtr p
);
```

Parameters

effect

An effect component instance. Your software obtains this reference from `OpenComponent` or `OpenDefaultComponent`.

p

A pointer to a `EffectsFrameParams` structure.

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

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`ImageCodec.h`

ImageCodecEffectConvertEffectSourceToFormat

Undocumented

```
ComponentResult ImageCodecEffectConvertEffectSourceToFormat (
   ComponentInstance effect,
   EffectSourcePtr sourceToConvert,
   ImageDescriptionHandle requestedDesc
);
```

Parameters

effect

An effect component instance. Your software obtains this reference from `OpenComponent` or `OpenDefaultComponent`.

sourceToConvert

Undocumented

requestedDesc

Undocumented

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

Version Notes
Introduced in QuickTime 3 or earlier.

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

Related Sample Code
Dimmer2Effect

Dimmer2Effect.win
GreyscaleEffectSample

Declared In
`ImageCodec.h`

ImageCodecEffectDisposeSMPTEFrame

Undocumented

```
ComponentResult ImageCodecEffectDisposeSMPTEFrame (
   ComponentInstance effect,
   SMPTEFrameReference frameRef
);
```

Parameters

effect

An effect component instance. Your software obtains this reference from `OpenComponent` or `OpenDefaultComponent`.

frameRef

Undocumented

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

Version Notes
Introduced in QuickTime 4.

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

Declared In
`ImageCodec.h`

ImageCodecEffectGetSpeed

Undocumented

```
ComponentResult ImageCodecEffectGetSpeed (
   ComponentInstance effect,
   QTAtomContainer parameters,
   Fixed *pFPS
);
```

Parameters

effect

An effect component instance. Your software obtains this reference from `OpenComponent` or `OpenDefaultComponent`.

parameters

Undocumented

pFPS

Undocumented

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

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`ImageCodec.h`

ImageCodecEffectPrepareSMPTEFrame

Undocumented

```
ComponentResult ImageCodecEffectPrepareSMPTEFrame (
   ComponentInstance effect,
   PixMapPtr destPixMap,
   SMPTEFrameReference *returnValue
);
```

Parameters

effect
: An effect component instance. Your software obtains this reference from `OpenComponent` or `OpenDefaultComponent`.

destPixMap
: *Undocumented*

returnValue
: *Undocumented*

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

Version Notes
Introduced in QuickTime 4.

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

Declared In
`ImageCodec.h`

ImageCodecEffectRenderFrame

Undocumented

```
ComponentResult ImageCodecEffectRenderFrame (
   ComponentInstance effect,
   EffectsFrameParamsPtr p
);
```

Parameters

effect

An effect component instance. Your software obtains this reference from `OpenComponent` or `OpenDefaultComponent`.

p

A pointer to an `EffectsFrameParams` structure.

Return Value

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

Version Notes

Introduced in QuickTime 3 or earlier.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`ImageCodec.h`

ImageCodecEffectRenderSMPTEFrame

Undocumented

```
ComponentResult ImageCodecEffectRenderSMPTEFrame (
   ComponentInstance effect,
   PixMapPtr destPixMap,
   SMPTEFrameReference frameRef,
   Fixed effectPercentageEven,
   Fixed effectPercentageOdd,
   Rect *pSourceRect,
   MatrixRecord *matrixP,
   SMPTEWipeType effectNumber,
   long xRepeat,
   long yRepeat,
   SMPTEFlags flags,
   Fixed penWidth,
   long strokeValue
);
```

Parameters

effect

An effect component instance. Your software obtains this reference from `OpenComponent` or `OpenDefaultComponent`.

destPixMap

Undocumented

frameRef

Undocumented

effectPercentageEven
Undocumented

effectPercentageOdd
Undocumented

pSourceRect
Undocumented

pMatrix
Undocumented

effectNumber
Undocumented

xRepeat
Undocumented

yRepeat
Undocumented

flags
Undocumented

penWidth
Undocumented

strokeValue
Undocumented

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

Version Notes
Introduced in QuickTime 4.

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

Declared In
`ImageCodec.h`

ImageCodecEffectSetup

Undocumented

```
ComponentResult ImageCodecEffectSetup (
   ComponentInstance effect,
   CodecDecompressParams *p
);
```

Parameters

effect
An effect component instance. Your software obtains this reference from `OpenComponent` or `OpenDefaultComponent`.

p
A pointer to a `CodecDecompressParams` structure.

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

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`ImageCodec.h`

ImageCodecEncodeFrame

Presents the compressor with a frame to encode.

```
ComponentResult ImageCodecEncodeFrame (
   ComponentInstance ci,
   ICMCompressorSourceFrameRef sourceFrame,
   UInt32 flags
);
```

Parameters

ci
A component instance that identifies a connection to an image codec component.

sourceFrame
The source frame to encode.

flags
Reserved; ignore.

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

Discussion
The compressor may encode the frame immediately or queue it for later encoding. If the compressor queues the frame for later decode, it must retain it (by calling `ICMCompressorSourceFrameRetain`) and release it when it is done with it (by calling `ICMCompressorSourceFrameRelease`). Pixel buffers are guaranteed to conform to the pixel buffer attributes returned by `ImageCodecPrepareToCompressFrames`. During multipass encoding, if the compressor requested the `kICMCompressionPassMode_NoSourceFrames` flag, the source frame pixel buffers may be NULL. (Note: this replaces `ImageCodecBandCompress` (page 25).)

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

Declared In
`ImageCodec.h`

ImageCodecEndBand

Notifies your image decompressor component that decompression of a band has finished or that it was terminated by the Image Compression Manager.

```
ComponentResult ImageCodecEndBand (
   ComponentInstance ci,
   ImageSubCodecDecompressRecord *drp,
   OSErr result,
   long flags
);
```

Parameters

ci

An image decompressor component instance. Your software obtains this reference from `OpenComponent` or `OpenDefaultComponent`.

drp

A pointer to an `ImageSubCodecDecompressRecord` structure.

result

A result code.

flags

Currently unused; set to 0.

Return Value

Returns `noErr` if the band or frame was drawn successfully. If it is any other value, the band or frame was not drawn. See `Error Codes`.

Discussion

Your image decompressor component is not required to implement this function. After your image decompressor component handles a call to this function, it can perform any tasks that are required when decompression is finished, such as disposing of data structures that are no longer needed. Because this function can be called at interrupt time, your component cannot use it to dispose of data structures; this must occur after handling the function.

Special Considerations

The base image decompressor may call `ImageCodecEndBand` at interrupt time.

Version Notes

Introduced in QuickTime 3 or earlier.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`ImageCodec.h`

ImageCodecExtractAndCombineFields

Performs field operations on video data.

```
ComponentResult ImageCodecExtractAndCombineFields (
   ComponentInstance ci,
   long fieldFlags,
   void *data1,
   long dataSize1,
   ImageDescriptionHandle desc1,
   void *data2,
   long dataSize2,
   ImageDescriptionHandle desc2,
   void *outputData,
   long *outDataSize,
   ImageDescriptionHandle descOut
);
```

Parameters

ci

An image decompressor component instance. Your software obtains this reference from `OpenComponent` or `OpenDefaultComponent`.

fieldFlags

Flags (see below) that specify the operation to be performed. A correctly formed request will specify two input fields, mapping one to the odd output field and the other to the even output field. See these constants:

```
evenField1ToEvenFieldOut
evenField1ToOddFieldOut
oddField1ToEvenFieldOut
oddField1ToOddFieldOut
evenField2ToEvenFieldOut
evenField2ToOddFieldOut
oddField2ToEvenFieldOut
oddField2ToOddFieldOut
```

data1

A pointer to a buffer containing the compressed image data for the first input field.

dataSize1

The size of the data1 buffer.

desc1

An `ImageDescription` structure describing the format and characteristics of the data in the data1 buffer.

data2

A pointer to a buffer containing the compressed image data for the second input field. Set to `NIL` if the requested operation uses only one input frame.

dataSize2

The size of the data2 buffer. Set to 0 if the requested operation uses only one input frame.

desc2

An `ImageDescription` structure describing the format and characteristics of the data in the data2 buffer. Set to `NIL` if the requested operation uses only one input frame.

outputData

A pointer to a buffer to receive the resulting frame.

outDataSize

On output this parameter returns the actual size of the new compressed image data.

descOut

The desired format of the resulting frames. Typically this is the same format specified by the desc1 and desc2 parameters.

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

Discussion
This function allows fields from two separate images, compressed in the same format, to be combined into a new compressed frame. Typically the operation results in an image of identical quality to the source images. Fields of a single image may also be duplicated or reversed by this function. The component selector for this function is:

```
kImageCodecExtractAndCombineFieldsSelect    =0x0015
```

Special Considerations

This codec routine implements the functionality of the `ImageFieldSequenceExtractCombine` function. If your codec is capable of separately compressing both fields of a video frame, you should incorporate support for this function. Your codec must ensure that it understands the image formats specified by desc1 and desc2 parameters, as these may not be the same as the codec's native image format. The image format specified by the `descOut` parameter will be the same as the codec's native image format.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`ImageCodec.h`

ImageCodecFlush

Empties an image decompressor component's input queue of pending scheduled frames.

```
ComponentResult ImageCodecFlush (
   ComponentInstance ci
);
```

Parameters

ci

An image decompressor component instance. Your software obtains this reference from `OpenComponent` or `OpenDefaultComponent`.

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

Discussion
Your component receives the `ImageCodecFlush` function whenever the Image Compression Manager needs to cancel the display of all scheduled frames. Your decompressor should empty its queue of scheduled asynchronous decompression requests. For each request, your component must call `ICMDecompressComplete`. Be sure to set the `err` parameter to -1, indicating that the request was canceled. Also, you must set both the `codecCompletionSource` and `codecCompletionDest` flags to 1.

Special Considerations

Your component's `ImageCodecFlush` function may be called at interrupt time. Only decompressor components that support scheduled asynchronous decompression receive this call.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`ImageCodec.h`

ImageCodecFlushFrame

Undocumented

```
ComponentResult ImageCodecFlushFrame (
   ComponentInstance ci,
   UInt32 flags
);
```

Parameters

ci

An image codec component instance. Your software obtains this reference from `OpenComponent` or `OpenDefaultComponent`.

flags

Undocumented

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

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`ImageCodec.h`

ImageCodecGetBaseMPWorkFunction

Gets an image codec's ComponentMPWorkFunctionProc callback.

```
ComponentResult ImageCodecGetBaseMPWorkFunction (
   ComponentInstance ci,
   ComponentMPWorkFunctionUPP *workFunction,
   void **refCon,
   ImageCodecMPDrawBandUPP drawProc,
   void *drawProcRefCon
);
```

Parameters

ci

An image codec component instance. Your software obtains this reference from `OpenComponent` or `OpenDefaultComponent`.

workFunction

On return, a pointer to a `ComponentMPWorkFunctionProc` callback.

refCon

On return, a handle to the reference constant that is passed to the `ComponentMPWorkFunctionProc` callback.

drawProc

An `ImageCodecMPDrawBandProc` callback.

drawProcRefCon

A pointer to a reference constant that is passed to your `ImageCodecMPDrawBandProc` callback. Use this parameter to point to a data structure containing any information your function needs.

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

Version Notes
Introduced in QuickTime 3 or earlier.

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

Related Sample Code
ElectricImageComponent
ElectricImageComponent.win
SoftVideoOutputComponent

Declared In
`ImageCodec.h`

ImageCodecGetCodecInfo

Notifies your codec whenever an application calls GetCodecInfo.

```
ComponentResult ImageCodecGetCodecInfo (
   ComponentInstance ci,
   CodecInfo *info
);
```

Parameters

ci

An image decompressor component instance. Your software obtains this reference from `OpenComponent` or `OpenDefaultComponent`.

info

A pointer to a `CodecInfo` structure to update. Your component should report its capabilities by formatting the structure in the location specified by this parameter.

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

Discussion
Both compressors and decompressors may receive this request.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`ImageCodec.h`

ImageCodecGetCompressedImageSize

Notifies your codec whenever an application calls GetCompressedImageSize.

```
ComponentResult ImageCodecGetCompressedImageSize (
   ComponentInstance ci,
   ImageDescriptionHandle desc,
   Ptr data,
   long bufferSize,
   ICMDataProcRecordPtr dataProc,
   long *dataSize
);
```

Parameters

ci

An image decompressor component instance. Your software obtains this reference from `OpenComponent` or `OpenDefaultComponent`.

desc

A handle to the `ImageDescription` structure that defines the compressed image for the operation.

data

A pointer to the compressed image data.

bufferSize

The size of the buffer to be used by the data-loading function specified by the `dataProc` parameter. If the application did not specify a data-loading function this parameter is `NIL`.

dataProc

A pointer to an `ICMDataProcRecord` structure. If the data stream is not all in memory when the application calls `GetCompressedImageSize`, your component may call an application function that loads more compressed data.

dataSize

A pointer to a field that is to receive the size, in bytes, of the compressed image.

Return Value

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

Special Considerations

Only decompressors receive this request.

Version Notes

Introduced in QuickTime 3 or earlier.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`ImageCodec.h`

ImageCodecGetCompressionTime

Notifies your codec whenever an application calls GetCompressionTime.

```
ComponentResult ImageCodecGetCompressionTime (
   ComponentInstance ci,
   PixMapHandle src,
   const Rect *srcRect,
   short depth,
   CodecQ *spatialQuality,
   CodecQ *temporalQuality,
   unsigned long *time
);
```

Parameters

ci

An image decompressor component instance. Your software obtains this reference from `OpenComponent` or `OpenDefaultComponent`.

src

A handle to the source image. The source image is stored in a `PixMap` structure. Applications may use the time information you return for more than one image. Consequently, your compressor should not consider the contents of the image when determining the maximum compression time. Rather, you should consider only the quality level, pixel depth, and image size. This parameter may also be set to `NIL`. In this case the application has not supplied a source image; your component should use the `other` parameters to determine the characteristics of the image to be compressed.

srcRect

A pointer to a `Rect` structure that defines the portion of the source image to compress.

depth

The depth at which the image is to be compressed. Values of 1, 2, 4, 8, 16, 24, and 32 indicate the number of bits per pixel for color images. Values of 33, 34, 36, and 40 indicate 1-bit, 2-bit, 4-bit, and 8-bit grayscale, respectively, for grayscale images.

spatialQuality

A pointer to a field containing the desired compressed image quality (see below). The compressor sets this field to the closest actual quality that it can achieve. See these constants:

```
codecMinQuality
codecLowQuality
codecNormalQuality
codecHighQuality
codecMaxQuality
codecLosslessQuality
```

temporalQuality

A pointer to a field containing the desired sequence temporal quality (see below). The compressor sets this field to the closest actual quality that it can achieve.

time

A pointer to a field to receive the compression time, in milliseconds. If your component cannot determine the amount of time required to compress the image, set this field to 0. Check to see if the `value` of this field is `NIL` and, if so, do not write to location 0.

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

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`ImageCodec.h`

ImageCodecGetDecompressLatency

Retrieves the video latency value from a specified video codec.

```
ComponentResult ImageCodecGetDecompressLatency (
   ComponentInstance ci,
   TimeRecord *latency
);
```

Parameters

ci

A video codec component instance. Your software obtains this reference from `OpenComponent` or `OpenDefaultComponent`.

latency

Pointer to a `TimeRecord` structure containing the latency time required for that codec.

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

Version Notes
Introduced in QuickTime 5.

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

Declared In
`ImageCodec.h`

ImageCodecGetDITLForSize

Returns the size of various dialog item lists.

```
ComponentResult ImageCodecGetDITLForSize (
   ComponentInstance ci,
   Handle *ditl,
   Point *requestedSize
);
```

Parameters

ci

The component instance that identifies your connection to an image codec component.

ditl

A pointer to a handle provided by the sequence grabber component. Your panel component returns the dialog item list in this handle. Your component should resize this handle as appropriate. The sequence grabber component will dispose of this handle after retrieving the item list, so make sure that the item list is not stored in a resource.

requestedSize

The size of the panel in pixels, or constants (see below). Two special values, `kSGSmallestDITLSize` and `kSGLargestDITLSize`, request the smallest or largest size of the list. The sequence grabber will interpolate the panel elements between the two sizes if just the constants are returned. A codec must at a minimum support `kSGSmallestDITLSize` if it implements this call. See these constants:

`kSGSmallestDITLSize`

`kSGLargestDITLSize`

Return Value
The codec returns `badComponentSelector` for sizes it does not implement and `noErr` if there is no error. See `Error Codes`.

Discussion
This function allows an image codec to return dialog item lists of various size in pixels. Once you have created the area, you can use other image codec calls to handle the dialog items managed by your panel component.

Version Notes
Introduced in QuickTime 6.

Availability
Available in Mac OS X v10.2 and later.

Declared In
`ImageCodec.h`

ImageCodecGetMaxCompressionSize

Notifies your codec whenever an application calls GetMaxCompressionSize.

```
ComponentResult ImageCodecGetMaxCompressionSize (
   ComponentInstance ci,
   PixMapHandle src,
   const Rect *srcRect,
   short depth,
   CodecQ quality,
   long *size
);
```

Parameters

ci

An image decompressor component instance. Your software obtains this reference from `OpenComponent` or `OpenDefaultComponent`.

src

A handle to the source image. The source image is stored in a `PixMap` structure. Applications use the size information you return to allocate buffers that may be used for more than one image. Consequently, your compressor should not consider the contents of the image when determining the maximum compressed size. Rather, you should consider only the quality level, pixel depth, and image size. This parameter may also be set to `NIL`. In this case the application has not supplied a source image; your component should use the `other` parameters to determine the characteristics of the image to be compressed.

srcRect

A pointer to a `Rect` structure that defines the portion of the source image to compress.

depth

The depth at which the image is to be compressed. Values of 1, 2, 4, 8, 16, 24, and 32 indicate the number of bits per pixel for color images. Values of 33, 34, 36, and 40 indicate 1-bit, 2-bit, 4-bit, and 8-bit grayscale, respectively, for grayscale images.

quality

The desired compressed image quality (see below). See these constants:

`codecMinQuality`
`codecLowQuality`
`codecNormalQuality`
`codecHighQuality`
`codecMaxQuality`
`codecLosslessQuality`

size

A pointer to a field to receive the maximum size, in bytes, of the compressed image.

Return Value

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

Discussion

The caller uses this function to determine the maximum size the data will become for a given parameter. Your component returns a long integer indicating the maximum number of bytes of compressed data that results from compressing the specified image.

Special Considerations

Only compressors receive this request.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`ImageCodec.h`

ImageCodecGetMaxCompressionSizeWithSources

Notifies your codec when an application calls GetCSequenceMaxCompressionSize.

```
ComponentResult ImageCodecGetMaxCompressionSizeWithSources (
   ComponentInstance ci,
   PixMapHandle src,
   const Rect *srcRect,
   short depth,
   CodecQ quality,
   CDSequenceDataSourcePtr sourceData,
   long *size
);
```

Parameters

ci

An image decompressor component instance. Your software obtains this reference from `OpenComponent` or `OpenDefaultComponent`.

src

A handle to the source image. The source image is stored in a `PixMap` structure. Applications use the size information you return to allocate buffers for more than one image. Consequently, your compressor should not consider the contents of the image when determining the maximum compressed size. Rather, you should consider only the quality level, pixel depth, and image size. This parameter may also be set to `NIL`. In this case the application has not supplied a source image; your component should use the `other` parameters to determine the characteristics of the image to be compressed.

srcRect

A pointer to a `Rect` structure that defines the portion of the source image to compress.

depth

The depth at which the image is to be compressed. Values of 1, 2, 4, 8, 16, 24, and 32 indicate the number of bits per pixel for color images. Values of 33, 34, 36, and 40 indicate 1-bit, 2-bit, 4-bit, and 8-bit grayscale, respectively, for grayscale images.

quality

The desired compression image quality.

sourceData

A pointer to a `CDSequenceDataSource` structure, which contains a linked list of all data sources. Because each data source contains a link to the next data source, a codec can access all data sources from this structure.

size

A pointer to a field to receive the maximum size, in bytes, of the compressed image.

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

Discussion
The caller uses this function to determine the maximum size the data will be compressed to for a given image and set of data sources.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`ImageCodec.h`

ImageCodecGetParameterList

Returns a parameter description atom container for a specified effect component instance.

```
ComponentResult ImageCodecGetParameterList (
   ComponentInstance ci,
   QTAtomContainer *parameterDescription
);
```

Parameters

ci

An effect component instance. Your software obtains this reference from `OpenComponent` or `OpenDefaultComponent`.

parameterDescription

The returned atom container for this component instance.

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

Discussion
This function returns the parameter description for the effect specified by the component instance `ci`, as a handle containing an `'atms'` resource of ID 1. The handle should be detached if it has been read in from a resource. Each parameter of the effect is described in the parameter description, with details of its name, type, legal values and hints about how a user interface to the parameter should be constructed.

Special Considerations

The calling application is responsible for disposing of the QT atom container returned in `parameterDescription`. The application should do this by calling `QTDisposeAtomContainer` once it has finished using the parameter description.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Related Sample Code
Fiendishthngs
qtshoweffect
qtshoweffect.win

Declared In
`ImageCodec.h`

ImageCodecGetParameterListHandle

Returns a handle to a Mac OS resource of type 'atms'.

```
ComponentResult ImageCodecGetParameterListHandle (
   ComponentInstance ci,
   Handle *parameterDescriptionHandle
);
```

Parameters

ci

An effect component instance. Your software obtains this reference from `OpenComponent` or `OpenDefaultComponent`.

parameterDescriptionHandle

A pointer to a handle to a Mac OS resource of type `'atms'`.

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

Discussion
The purpose of this function is to build a QT atom container in response to a call to `ImageCodecGetParameterList` (page 58). The handle should be detached if it has been read in from a resource. The caller is responsible for disposing of the handle.

Special Considerations

The implementation of this function in the Base Effect component reads in and detaches a Component Public Resource of type `'atms'` and ID 1.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`ImageCodec.h`

ImageCodecGetSettings

Returns the codec settings chosen by the user.

```
ComponentResult ImageCodecGetSettings (
   ComponentInstance ci,
   Handle settings
);
```

Parameters

ci

An image compressor component instance. Your software obtains this reference from `OpenComponent` or `OpenDefaultComponent`.

settings

A handle that the codec should resize and fill in with the current internal settings. If there are no current internal settings, resize it to 0. Don't dispose of this handle.

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

Discussion
`ImageCodecGetSettings` returns the codec's current internal settings. If there are no current settings or the settings are the same as the defaults, the codec can set the handle to `NIL`.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Related Sample Code
Quartz Composer QCTV

Declared In
`ImageCodec.h`

ImageCodecGetSettingsAsText

Undocumented

```
ComponentResult ImageCodecGetSettingsAsText (
   ComponentInstance ci,
   Handle *text
);
```

Parameters

ci

An image compressor component instance. Your software obtains this reference from `OpenComponent` or `OpenDefaultComponent`.

text

Undocumented

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

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`ImageCodec.h`

ImageCodecGetSimilarity

Notifies your codec when an application calls GetSimilarity.

```
ComponentResult ImageCodecGetSimilarity (
   ComponentInstance ci,
   PixMapHandle src,
   const Rect *srcRect,
   ImageDescriptionHandle desc,
   Ptr data,
   Fixed *similarity
);
```

Parameters

ci

An image decompressor component instance. Your software obtains this reference from `OpenComponent` or `OpenDefaultComponent`.

src

A handle to the noncompressed image. The image is stored in a `PixMap` structure.

srcRect

A pointer to a `Rect` structure that defines the portion of the image to compare to the compressed image.

desc

A handle to the `ImageDescription` structure that defines the compressed image for the operation.

data

A pointer to the compressed image data.

similarity

A pointer to a field that is to receive the similarity value. Your component sets this field to reflect the relative similarity of the two images. Valid values range from 0 (key frame) to 1 (identical).

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

Discussion
Only decompressors receive this request.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`ImageCodec.h`

ImageCodecGetSourceDataGammaLevel

Returns the native gamma of compressed data, if any.

```
ComponentResult ImageCodecGetSourceDataGammaLevel (
   ComponentInstance ci,
   Fixed *sourceDataGammaLevel
);
```

Parameters

ci

An image decompressor component instance. Your software obtains this reference from `OpenComponent` or `OpenDefaultComponent`.

sourceDataGammaLevel

On return, the native gamma level of the compressed data. If the value is 0, it is the default gamma level of the platform.

Return Value

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

Discussion

The ICM uses the information returned by this function to determine what gamma correction is necessary. For example, the Apple DV Codec returns 2.2.

Version Notes

Introduced in QuickTime 5.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`ImageCodec.h`

ImageCodecHitTestData

Notifies your codec when the application calls PtInDSequenceData.

```
ComponentResult ImageCodecHitTestData (
   ComponentInstance ci,
   ImageDescriptionHandle desc,
   void *data,
   Size dataSize,
   Point where,
   Boolean *hit
);
```

Parameters

ci

An image decompressor component instance. Your software obtains this reference from `OpenComponent` or `OpenDefaultComponent`.

desc

An `ImageDescriptionHandle` for the image data pointed to by the `data` parameter.

data

Pointer to compressed data in the format specified by the `desc` parameter.

dataSize

`Size` of the compressed data referred to by the `data` parameter.

where

A QuickDraw `Point` structure (0,0) based at the top-left corner of the image.

hit

A pointer to a Boolean value. The value should be set to TRUE if the point specified by the `where` parameter is contained within the compressed image data specified by the `data` parameter, or FALSE if the specified point falls within a blank portion of the image.

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

Discussion
`ImageCodecHitTestData` allows the calling application to perform hit testing on compressed data.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`ImageCodec.h`

ImageCodecHitTestDataWithFlags

Undocumented

```
ComponentResult ImageCodecHitTestDataWithFlags (
   ComponentInstance ci,
   ImageDescriptionHandle desc,
   void *data,
   Size dataSize,
   Point where,
   long *hit,
   long hitFlags
);
```

Parameters

ci

An image codec component instance. Your software obtains this reference from `OpenComponent` or `OpenDefaultComponent`.

desc

A handle to an `ImageDescription` structure for the image data pointed to by the `data` parameter.

data

Pointer to compressed data in the format specified by the `desc` parameter.

dataSize

`Size` of the compressed data referred to by the `data` parameter.

where

A QuickDraw `Point` structure (0,0) based at the top-left corner of the image.

hit

A pointer to a Boolean. The Boolean should be set to TRUE if the point specified by the `where` parameter is contained within the compressed image data specified by the `data` parameter.

hitFlags

Undocumented

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

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`ImageCodec.h`

ImageCodecInitialize

Called before making any other all calls to your component.

```
ComponentResult ImageCodecInitialize (
   ComponentInstance ci,
   ImageSubCodecDecompressCapabilities *cap
);
```

Parameters

ci

An image decompressor component instance. Your software obtains this reference from `OpenComponent` or `OpenDefaultComponent`.

cap

On return, an `ImageSubCodecDecompressCapabilities` structure that contains the capabilities of the image decompressor component. This structure contains two fields. The `canAsync` field specifies whether your component can support asynchronous decompression operations. The `decompressRecordSize` field specifies the size of the decompression record structure for your component.

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

Discussion
Your component must implement this function.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`ImageCodec.h`

ImageCodecIsImageDescriptionEquivalent

Compares image descriptions.

```
ComponentResult ImageCodecIsImageDescriptionEquivalent (
   ComponentInstance ci,
   ImageDescriptionHandle newDesc,
   Boolean *equivalent
);
```

Parameters

ci

An image compressor component instance. Your software obtains this reference from `OpenComponent` or `OpenDefaultComponent`.

newDesc

A handle to the `ImageDescription` structure that describes the compressed image.

equivalent

A pointer to a Boolean value. If the structure provided in the `newDesc` parameter is equivalent to the `ImageDescription` structure currently in use by the image sequence, this value is set to TRUE. If they are not equivalent, and therefore a new image sequence must be created to display an image using the new `ImageDescription` structure, this value is set to FALSE.

Return Value

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

Discussion

Your component receives this call whenever an application calls `CDSequenceEquivalentImageDescription`. Implementing this function can significantly improve playback of edited video sequences using your codec. For example, if two sequences are compressed at different quality levels and are edited together they will have different image descriptions because their quality values will be different. This will force QuickTime to use two separate decompressor instances to display the images. By implementing this function your decompressor can tell QuickTime that differences in quality levels don't require separate decompressors. This saves memory and time, thus improving performance.

Special Considerations

The current `ImageDescription` structure is not passed in this function because the Image Compression Manager assumes the codec has already made copies of all relevant data fields from the current `ImageDescription` structure during the `ImageCodecPreDecompress` (page 70) call.

Version Notes

Introduced in QuickTime 3 or earlier.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`ImageCodec.h`

ImageCodecIsStandardParameterDialogEvent

Processes events related to a standard parameters dialog box created by ImageCodecCreateStandardParameterDialog.

```
ComponentResult ImageCodecIsStandardParameterDialogEvent (
   ComponentInstance ci,
   EventRecord *pEvent,
   QTParameterDialog createdDialog
);
```

Parameters

ci

An effect component instance. Your software obtains this reference from `OpenComponent` or `OpenDefaultComponent`. This must be the instance that was passed to `ImageCodecCreateStandardParameterDialog` (page 31) to create the dialog box.

pEvent

A pointer to an `EventRecord` structure.

createdDialog

A reference to the dialog box created by the call to `ImageCodecCreateStandardParameterDialog` (page 31).

Return Value

If the error code returned is `featureUnsupported`, your application should process the event in the normal way. If it is `noErr`, the event was processed. If this function returns any other value, an error occurred. See `Error Codes`.

Discussion

This function returns an error code that indicates whether the event pointed to by `pEvent` was processed or not. After you call `ImageCodecCreateStandardParameterDialog` (page 31) to create a standard parameter dialog box, you must pass every non-null event to this function. It processes events related to the standard parameter dialog box, passing other events to your application for processing.

Version Notes

Introduced in QuickTime 3 or earlier.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

qtshoweffect

qtshoweffect.win

Declared In

`ImageCodec.h`

ImageCodecMergeFloatingImageOntoWindow

Draws the current contents of a floating image.

```
ComponentResult ImageCodecMergeFloatingImageOntoWindow (
   ComponentInstance ci,
   UInt32 flags
);
```

Parameters

ci

The component instance that identifies your connection to an image codec component.

flags

Currently not implemented.

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

Discussion
Some hardware acceleration transfer codecs create a floating image in front of the window; when this is deactivated or hidden, whatever was previously drawn in that section of the window reappears. Such transfer codecs should implement this function, which draws the current contents of the floating image onto the window below, so that the floating image may be deactivated or hidden without the image changing.

Version Notes
Introduced in QuickTime 6.

Availability
Available in Mac OS X v10.2 and later.

Declared In
`ImageCodec.h`

ImageCodecNewImageBufferMemory

Asks a codec to allocate memory for an offscreen buffer of non-RGB pixels.

```
ComponentResult ImageCodecNewImageBufferMemory (
   ComponentInstance ci,
   CodecDecompressParams *params,
   long flags,
   ICMMemoryDisposedUPP memoryGoneProc,
   void *refCon
);
```

Parameters

ci

An image decompressor component instance. Your software obtains this reference from `OpenComponent` or `OpenDefaultComponent`.

params

A pointer to a decompression parameters structure.

flags

Currently, this parameter is always set to 0.

memoryGoneProc

A pointer to an `ICMMemoryDisposedProc` callback that will be called before disposing of the memory allocated by the codec.

refCon

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

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

Discussion
This call is used to support a codec decompressing into a non-RGB buffer. The transfer codec is responsible for defining the offscreen buffer and transferring the image from the offscreen buffer to the destination. Your component receives this call whenever another codec has requested a non-RGB offscreen buffer of the type of your component's subtype.

Special Considerations

The Image Compression Manager does not currently track memory allocations. When a compressor or decompressor component instance is closed, it must ensure that all blocks allocated by that instance are disposed of and call the `ICMMemoryDisposedProc` callback.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`ImageCodec.h`

ImageCodecNewImageGWorld

Undocumented

```
ComponentResult ImageCodecNewImageGWorld (
   ComponentInstance ci,
   CodecDecompressParams *params,
   GWorldPtr *newGW,
   long flags
);
```

Parameters

ci

An image codec component instance. Your software obtains this reference from `OpenComponent` or `OpenDefaultComponent`.

params

A pointer to a `CodecDecompressParams` structure.

newGW

A pointer to a pointer to a `CGrafPort` structure.

flags

Undocumented

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

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`ImageCodec.h`

ImageCodecNewMemory

Requests codec-allocated memory.

```
ComponentResult ImageCodecNewMemory (
   ComponentInstance ci,
   Ptr *data,
   Size dataSize,
   long dataUse,
   ICMMemoryDisposedUPP memoryGoneProc,
   void *refCon
);
```

Parameters

ci

An image decompressor component instance. Your software obtains this reference from `OpenComponent` or `OpenDefaultComponent`.

data

Returns a pointer to the allocated memory.

dataSize

The desired size of the data buffer.

dataUse

A constant (see below) that indicates how the memory is to be used. For example, the memory may be used to store compressed data before it's displayed, mask plane data, or decompressed data. If there is no benefit to storing a particular kind of data in codec memory, the codec should refuse the request for the memory allocation. See these constants:

memoryGoneProc

A pointer to an `ICMMemoryDisposedProc` callback that will be called before disposing of the memory allocated by a codec.

refCon

A reference constant value that your codec must pass to the `memoryGoneProc` callback. Use this parameter to point to a data structure containing any information your callback needs.

Return Value

See `Error Codes`. Returns `noErr` if there is no error. If your codec does not currently have free memory for compression frame data, but will soon, you can return `codecNoMemoryPleaseWaitErr` to indicate this fact.

Discussion

Some hardware codecs may have on-board memory that can be used to store compressed and/or decompressed data. This function makes this memory available for use by clients of the codec. Some software codecs may be able to optimize their performance by having more control over memory allocation. This function makes such control available. Your component receives this call whenever an application calls `CDSequenceNewMemory`.

Special Considerations

The Image Compression Manager does not currently track memory allocations. When a compressor or decompressor component instance is closed, it must ensure that all blocks allocated by that instance are disposed and call the `ICMMemoryDisposedProc` callback.

Version Notes

Introduced in QuickTime 3 or earlier.

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

Declared In
`ImageCodec.h`

ImageCodecPreCompress

Notifies your component before compressing an image or a band of an image.

```
ComponentResult ImageCodecPreCompress (
   ComponentInstance ci,
   CodecCompressParams *params
);
```

Parameters

ci

An image decompressor component instance. Your software obtains this reference from `OpenComponent` or `OpenDefaultComponent`.

params

A pointer to a `CodecCompressParams` structure. The Image Compression Manager places the appropriate parameter information in that structure.

Return Value
See `Error Codes`. Your component should return a result code of `codecConditionErr` if it cannot field the compression request. Return `noErr` if there is no error.

Discussion
Your component receives this call before compressing an image or a band of an image. The Image Compression Manager also calls this function when processing a sequence. In that case, the Image Compression Manager calls this function whenever the parameters governing the sequence operation have changed substantially. Your component indicates whether it can perform the requested compression operation.

Special Considerations

Only compressors receive this call.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`ImageCodec.h`

ImageCodecPreDecompress

Notifies your component before decompressing an image or sequence of frames.

```
ComponentResult ImageCodecPreDecompress (
   ComponentInstance ci,
   CodecDecompressParams *params
);
```

Parameters

ci

An image decompressor component instance. Your software obtains this reference from `OpenComponent` or `OpenDefaultComponent`.

params

A pointer to a `CodecDecompressParams` structure. The Image Compression Manager places the appropriate parameter information in that structure.

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

Discussion
If your decompressor component supports scheduled asynchronous decompression operations, be sure to set the `codecCanAsyncWhen` flag to 1 in the `flags` field of your component's `CodecCapabilities` structure. If you set `codecCanAsyncWhen`, you must also set `codecCanAsync`. Codecs that support scheduled asynchronous decompression are strongly advised to also set the `codecCanShieldCursor` flag.

If your decompressor component uses a secondary hardware buffer for its images, be sure to set the `codecHasVolatileBuffer` flag to 1 in the `flags` field of your component's `CodecCapabilities` structure. If your decompressor component is used solely as a transfer codec and uses the `ImageCodecNewImageBufferMemory` (page 67) function to create an offscreen buffer that is really onscreen, your codec will need to set the `codecImageBufferIsOnScreen` flag to 1.

Special Considerations

Only decompressors receive this request.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`ImageCodec.h`

ImageCodecPreflight

Called before decompressing an image, in response to an ImageCodecPreDecompress call from the Image Compression Manager.

```
ComponentResult ImageCodecPreflight (
   ComponentInstance ci,
   CodecDecompressParams *params
);
```

Parameters

ci

An image decompressor component instance. Your software obtains this reference from `OpenComponent` or `OpenDefaultComponent`.

params

A pointer to a `CodecDecompressParams` structure.

Return Value

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

Discussion

Your codec responds to this call by returning information about its capabilities in a `CodecCapabilities` structure. The Image Compression Manager creates the decompression parameters structure, and your image decompressor component is required only to provide values for the `wantedDestinationPixelSize` and `wantedDestinationPixelTypes` fields of the structure. Your image decompressor component can also modify other fields if necessary. For example, if it can scale images, it must set the `codecCapabilityCanScale` flag in the `capabilities` field of the structure.

Special Considerations

Your component must implement this function.

Version Notes

Introduced in QuickTime 3 or earlier.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`ImageCodec.h`

ImageCodecPrepareToCompressFrames

Prepares the compressor to receive frames.

```
ComponentResult ImageCodecPrepareToCompressFrames (
   ComponentInstance ci,
   ICMCompressorSessionRef session,
   ICMCompressionSessionOptionsRef compressionSessionOptions,
   ImageDescriptionHandle imageDescription,
   void *reserved,
   CFDictionaryRef *compressorPixelBufferAttributesOut
);
```

Parameters

ci

A component instance that identifies a connection to an image codec component.

session

The compressor session reference. The compressor should store this in its globals; it will need it when calling the ICM back (for example, to call `ICMEncodedFrameCreateMutable` and `ICMCompressorSessionEmitEncodedFrame`). This is not a CF type. Do not call `CFRetain` or `CFRelease` on it.

compressionSessionOptions

The session options from the client. The compressor should retain this and use the settings to guide compression.

imageDescription

The image description. The compressor may add image description extensions.

reserved

Reserved for future use. Ignore this parameter.

compressorPixelBufferAttributesOut

The compressor should create a pixel buffer attributes dictionary and set `compressorPixelBufferAttributesOut` to it. The ICM will release it.

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

Discussion
The compressor should record session and retain `compressionSessionOptions` for use in later calls. The compressor may modify `imageDescription` at this point. The compressor should create and return pixel buffer attributes, which the ICM will release. (Note: this replaces `ImageCodecPreCompress` (page 70).)

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

Declared In
`ImageCodec.h`

ImageCodecProcessBetweenPasses

Provides the compressor with an opportunity to perform processing between passes.

```
ComponentResult ImageCodecProcessBetweenPasses (
   ComponentInstance ci,
   ICMMultiPassStorageRef multiPassStorage,
   Boolean *interpassProcessingDoneOut,
   ICMCompressionPassModeFlags *requestedNextPassModeFlagsOut
);
```

Parameters

ci

A component instance that identifies a connection to an image codec component.

multiPassStorage

The multipass storage object that the compressor should use to store and retrieve information between passes.

interpassProcessingDoneOut

Points to a Boolean. Set this to FALSE if you want your `ImageCodecProcessBetweenPasses` function to be called again to perform more processing, TRUE if not.

requestedNextPassModeFlagsOut

Set *`requestedNextPassModeFlagsOut` to indicate the type of pass that should be performed next: To recommend a repeated analysis pass, set it to `kICMCompressionPassMode_ReadFromMultiPassStorage`|`kICMCompressionPassMode_WriteToMultiPassStorage`. To recommend a final encoding pass, set it to `kICMCompressionPassMode_ReadFromMultiPassStorage`|`kICMCompressionPassMode_OutputEncodedFrames`. If source frame buffers are not necessary for the recommended pass (for example, because all the required data has been copied into multipass storage), set `kICMCompressionPassMode_NoSourceFrames`.

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

Discussion
This function will be called repeatedly until it returns TRUE in *`interpassProcessingDoneOut`. The compressor may read and write to `multiPassStorage`. The compressor should indicate which type of pass it would prefer to perform next by setting *`requestedNextPassTypeOut`.

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

Declared In
`ImageCodec.h`

ImageCodecQueueStarting

Called by the base image decompressor before decompressing the frames in the queue if your image decompressor component supports asynchronous scheduled decompression.

```
ComponentResult ImageCodecQueueStarting (
   ComponentInstance ci
);
```

Parameters

ci

An image decompressor component instance. Your software obtains this reference from `OpenComponent` or `OpenDefaultComponent`.

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

Discussion
Your component is not required to implement this function. It can perform any tasks at this time, such as locking data structures.

Special Considerations

The base image decompressor never calls this function at interrupt time.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`ImageCodec.h`

ImageCodecQueueStopping

Notifies your component that the frames in the queue have been decompressed, if your image decompressor component supports asynchronous scheduled decompression.

```
ComponentResult ImageCodecQueueStopping (
   ComponentInstance ci
);
```

Parameters

ci

An image decompressor component instance. Your software obtains this reference from `OpenComponent` or `OpenDefaultComponent`.

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

Discussion
After your image decompressor component handles a call to this function, it can perform any tasks that are required when decompression of the frames is finished, such as disposing of data structures that are no longer needed.

Special Considerations

Your component is not required to implement this function. This function is never called at interrupt time.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`ImageCodec.h`

ImageCodecRemoveFloatingImage

Hides an image codec's floating image without having to close the component.

```
ComponentResult ImageCodecRemoveFloatingImage (
   ComponentInstance ci,
   UInt32 flags
);
```

Parameters

ci

The component instance that identifies your connection to an image codec component.

flags

Undocumented

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

Discussion
Some hardware acceleration transfer codecs create a floating image in front of the window; when this is deactivated or hidden, whatever was previously drawn in that section of the window reappears. Such transfer codecs should implement this function, so the Image Compression Manager can ask it to hide the floating image without having to close the component. The floating image should be shown again on the next call to `ImageCodecDrawBand` (page 38).

Version Notes
Introduced in QuickTime 6.

Availability
Available in Mac OS X v10.2 and later.

Declared In
`ImageCodec.h`

ImageCodecRequestGammaLevel

Asks an image codec to convert from source to destination gamma levels.

```
ComponentResult ImageCodecRequestGammaLevel (
   ComponentInstance ci,
   Fixed srcGammaLevel,
   Fixed dstGammaLevel,
   long *codecCanMatch
);
```

Parameters

ci

An image decompressor component instance. Your software obtains this reference from `OpenComponent` or `OpenDefaultComponent`.

srcGammaLevel

The gamma level to convert from.

dstGammaLevel

The gamma level to convert to.

codecCanMatch

Pointer to a value that indicates if the conversion from `srcGammaLevel` to `dstGammaLevel` is supported.

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

Discussion
This function tells the codec what the gamma of the source buffer and destination pixel map are so that the codec can try to convert between the two gammas when decompressing. Proper gamma conversion is accomplished by normalizing source data to black and white points to 0 to 1 and raising the result by the ratio of the `srcGammaLevel` divided by `dstGammaLevel`. The most accurate correction is done in RGB space, but a visual approximation can be done by raising the luma component alone.

Special Considerations

This function can be called several times as the ICM sets up a gamma conversion chain. The last value takes precedent for future scheduled frames. It may also be called while frames are already scheduled, indicating that conditions have changed. The new request is effective on frames that are scheduled after the call is made. Frames previously scheduled should continue to use the previously requested gamma conversion values.

Version Notes
Introduced in QuickTime 5.

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

Declared In
`ImageCodec.h`

ImageCodecRequestSettings

Displays a dialog box containing codec-specific compression settings.

```
ComponentResult ImageCodecRequestSettings (
   ComponentInstance ci,
   Handle settings,
   Rect *rp,
   ModalFilterUPP filterProc
);
```

Parameters

ci

An image compressor component instance. Your software obtains this reference from `OpenComponent` or `OpenDefaultComponent`.

settings

A handle of data specific to the codec. If the handle is empty, the codec should use its default settings.

rp

A pointer to a `Rect` structure giving the coordinates of the standard compression dialog box in global screen coordinates. The codec can use this to position its dialog box in the same area of the screen.

filterProc

A pointer to a `ModalFilterProc` callback that the codec must either pass to the Mac OS `ModalDialog` function or call at the beginning of the codec dialog process. This callback gives the calling application and standard compression dialog box a chance to process update events.

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

Discussion
The `ImageCodecRequestSettings` function allows the display of a dialog box of additional compression settings specific to the codec. These settings are stored in a settings handle. The codec can store any data in any format it wants in the settings handle and resize it accordingly. It should store some type of tag or version information that it can use to verify that the data belongs to the codec. The codec should not dispose of the handle.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`ImageCodec.h`

ImageCodecScheduleFrame

Undocumented

```
ComponentResult ImageCodecScheduleFrame (
   ComponentInstance ci,
   const ImageSubCodecDecompressRecord *drp,
   ImageCodecTimeTriggerUPP triggerProc,
   void *triggerProcRefCon
);
```

Parameters

ci

An image codec component instance. Your software obtains this reference from `OpenComponent` or `OpenDefaultComponent`.

drp

A pointer to an `ImageSubCodecDecompressRecord` structure.

triggerProc

An `ImageCodecTimeTriggerProc` callback.

triggerProcRefCon

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

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

Version Notes
Introduced in QuickTime 4.

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

Declared In
`ImageCodec.h`

ImageCodecSetSettings

Sets the settings of an optional image codec dialog box.

```
ComponentResult ImageCodecSetSettings (
   ComponentInstance ci,
   Handle settings
);
```

Parameters

ci

An image compressor component instance. Your software obtains this reference from `OpenComponent` or `OpenDefaultComponent`.

settings

A handle to internal settings originally returned by either `ImageCodecRequestSettings` (page 77) or `ImageCodecGetSettings` (page 59). The codec should set its internal settings to match those of the settings handle. Because the codec does not own the handle, it should not dispose of it and should copy only its contents, not the handle itself. If the settings handle passed is empty, the codec should sets its internal settings to a default state.

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

Discussion
This function allows a codec to return its private settings. Set the codec's internal settings to the state specified in the settings handle. The codec should always check the validity of the contents of the handle so that invalid settings are not used.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`ImageCodec.h`

ImageCodecSetTimeBase

Sets the time base for an image codec component.

```
ComponentResult ImageCodecSetTimeBase (
   ComponentInstance ci,
   void *base
);
```

Parameters

ci

An image codec component instance. Your software obtains this reference from `OpenComponent` or `OpenDefaultComponent`.

base

A pointer to the time base for this operation. Your application obtains this time base identifier from `NewTimeBase`.

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

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`ImageCodec.h`

ImageCodecSetTimeCode

Sets the timecode for the next frame that is to be decompressed.

```
ComponentResult ImageCodecSetTimeCode (
   ComponentInstance ci,
   void *timeCodeFormat,
   void *timeCodeTime
);
```

Parameters

ci

An image decompressor component instance. Your software obtains this reference from `OpenComponent` or `OpenDefaultComponent`.

timeCodeFormat

A pointer to a `TimeCodeDef` structure. This structure contains the timecode definition information for the next frame to be decompressed.

timeCodeTime

A pointer to a `TimeCodeRecord` structure. This structure contains the time value for the next frame in the current sequence.

Return Value

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

Discussion

Your component receives this call whenever an application calls `SetDSequenceTimeCode`. That function allows an application to set the timecode for a frame that is to be decompressed. The timecode information you receive applies to the next frame to be decompressed and is provided to the decompressor by `ImageCodecBandDecompress` (page 26).

Version Notes

Introduced in QuickTime 3 or earlier.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`ImageCodec.h`

ImageCodecSourceChanged

Notifies your codec that one of the data sources has changed when an application calls CDSequenceSetSourceData or CDSequenceChangedSourceData.

```
ComponentResult ImageCodecSourceChanged (
   ComponentInstance ci,
   UInt32 majorSourceChangeSeed,
   UInt32 minorSourceChangeSeed,
   CDSequenceDataSourcePtr sourceData,
   long *flagsOut
);
```

Parameters

ci

An image decompressor component instance. Your software obtains this reference from `OpenComponent` or `OpenDefaultComponent`.

majorSourceChangeSeed

An integer value that is incremented each time a data source is added or removed. This provides an easy way for a codec to know when it needs to redetermine which data source inputs are available.

minorSourceChangeSeed

An integer value that is incremented each time a data source is added or removed, or the data contained in any of the data sources changes. This provides a way for a codec to know if the data available to it has changed.

sourceData

A pointer to a `CDSequenceDataSource` structure. This structure contains a linked list of all data sources. Because each data source contains a link to the next data source, a codec can access all data sources from this structure.

flagsOut

Undocumented

Return Value

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

Version Notes

Introduced in QuickTime 3 or earlier.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`ImageCodec.h`

ImageCodecStandardParameterDialogDoAction

Allows you to control the behavior of a standard parameter dialog box created by ImageCodecCreateStandardParameterDialog.

```
ComponentResult ImageCodecStandardParameterDialogDoAction (
   ComponentInstance ci,
   QTParameterDialog createdDialog,
   long action,
   void *params
);
```

Parameters

ci

An effect component instance. Your software obtains this reference from `OpenComponent` or `OpenDefaultComponent`. This must be the same instance as was passed to `ImageCodecCreateStandardParameterDialog` (page 31) to create the dialog box.

createdDialog

A reference to the dialog box created by the call to `ImageCodecCreateStandardParameterDialog`.

action

The action selector (see below), which determines which of the available actions you want the function to perform. See these constants:

`pdActionConfirmDialog`
`pdActionSetAppleMenu`
`pdActionSetEditMenu`
`pdActionGetDialogValues`
`pdActionSetPreviewUserItem`
`pdActionSetPreviewPicture`
`pdActionSetColorPickerEventProc`
`pdActionSetDialogTitle`
`pdActionGetSubPanelMenu`
`pdActionActivateSubPanel`
`pdActionConductStopAlert`

params

The (optional) parameter to the `action`. The type passed in this parameter depends on the value of the `action` parameter.

Return Value

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

Discussion

This function allows you to change the default behavior of the standard parameter dialog box, and provides a mechanism for your application to communicate with controls that were incorporated into an application dialog box. It also allows you to retrieve parameter values from the dialog box at any time. You specify which function will be performed by passing an action selector in the `action` parameter and, optionally, a single parameter in the `params` parameter. Some of the actions you can specify through this function are only appropriate if you have incorporated standard parameter dialog box controls within a dialog box created by your application.

Version Notes

Introduced in QuickTime 3 or earlier.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code
qtshoweffect
qtshoweffect.win

Declared In
`ImageCodec.h`

ImageCodecTrimImage

Notifies your component whenever an application calls TrimImage.

```
ComponentResult ImageCodecTrimImage (
   ComponentInstance ci,
   ImageDescriptionHandle Desc,
   Ptr inData,
   long inBufferSize,
   ICMDataProcRecordPtr dataProc,
   Ptr outData,
   long outBufferSize,
   ICMFlushProcRecordPtr flushProc,
   Rect *trimRect,
   ICMProgressProcRecordPtr progressProc
);
```

Parameters

ci

An image decompressor component instance. Your software obtains this reference from `OpenComponent` or `OpenDefaultComponent`.

Desc

A handle to the `ImageDescription` structure that describes the compressed image. Your component updates this structure to refer to the resized image.

inData

A pointer to the compressed image data. If the entire compressed image cannot be stored at this location, the application may provide a data-loading function; see the description of the `dataProc` parameter to this function for details. This is a 32-bit clean address.

inBufferSize

The size of the buffer to be used by the data-loading function specified by the `dataProc` parameter. If the application did not specify a data-loading function, this parameter is `NIL`.

dataProc

A pointer to an `ICMDataProcRecord` structure. If the application did not provide a data-loading function, this parameter is `NIL`. In this case, the entire image must be in memory at the location specified by the `inData` parameter. If the data stream is not all in memory when the application calls `GetCompressedImageSize`, your component may call an application function that loads more compressed data.

outData

A pointer to a buffer to receive the trimmed image. If there is not sufficient memory to store the compressed image, the application may choose to write the compressed data to mass storage during the compression operation. The `flushProc` parameter identifies the data-unloading function. This is a 32-bit clean address.

outBufferSize

The size of the buffer to be used by the data-unloading function specified by the `flushProc` parameter. If the application did not specify a data-unloading function, this parameter is `NIL`.

flushProc

A pointer to an `ICMFlushProcRecord` structure. If the application did not provide a data-unloading function, this parameter is `NIL`. In this case, your component writes the entire compressed image into the memory location specified by the `outData` parameter. If there is not enough memory to store the compressed image, your component may call an application function that unloads some of the compressed data.

trimRect

A pointer to a `Rect` structure that defines the desired image dimensions. Your component adjusts the structure's values so that they refer to the same rectangle in the resulting image. This is necessary whenever data is removed from the beginning of the image.

progressProc

A pointer to an `ICMProgressProcRecord` structure. During the operation, your component should occasionally call an application function to report its progress. If the application did not provide a progress function, this parameter is `NIL`.

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

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`ImageCodec.h`

ImageCodecValidateParameters

Validates effect parameters.

```
ComponentResult ImageCodecValidateParameters (
   ComponentInstance ci,
   QTAtomContainer parameters,
   QTParameterValidationOptions validationFlags,
   StringPtr errorString
);
```

Parameters

ci

An image decompressor component instance. Your software obtains this reference from `OpenComponent` or `OpenDefaultComponent`.

parameters

The atom container containing the `effect` parameters to be validated.

validationFlags

Constants (see below) that control validation. See these constants:

`kParameterValidationNoFlags`

`kParameterValidationFinalValidation`

errorString
Undocumented

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

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`ImageCodec.h`

NewImageCodecDrawBandCompleteUPP

Allocates a Universal Procedure Pointer for an ImageCodecDrawBandCompleteProc callback.

```
ImageCodecDrawBandCompleteUPP NewImageCodecDrawBandCompleteUPP (
   ImageCodecDrawBandCompleteProcPtr userRoutine
);
```

Parameters
userRoutine
A pointer to your application-defined function.

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

Version Notes
Introduced in QuickTime 5.

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

Declared In
`ImageCodec.h`

NewImageCodecMPDrawBandUPP

Allocates a Universal Procedure Pointer for the ImageCodecMPDrawBandProc callback.

```
ImageCodecMPDrawBandUPP NewImageCodecMPDrawBandUPP (
   ImageCodecMPDrawBandProcPtr 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 `NewImageCodecMPDrawBandProc`.

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

Related Sample Code
ElectricImageComponent
ElectricImageComponent.win
SoftVideoOutputComponent

Declared In
`ImageCodec.h`

NewImageCodecTimeTriggerUPP

Allocates a Universal Procedure Pointer for the ImageCodecTimeTriggerProc callback.

```
ImageCodecTimeTriggerUPP NewImageCodecTimeTriggerUPP (
   ImageCodecTimeTriggerProcPtr 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 `NewImageCodecTimeTriggerProc`.

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

Declared In
`ImageCodec.h`

QTPhotoDefineHuffmanTable

Defines a Huffman table.

```
ComponentResult QTPhotoDefineHuffmanTable (
   ComponentInstance codec,
   short componentNumber,
   Boolean isDC,
   unsigned char *lengthCounts,
   unsigned char *values
);
```

Parameters

codec

Identifies your connection to the image compressor component.

componentNumber

Specifies a color component. If 0, the luminance Huffman table is set. If 1, the chrominance Huffman table is set.

isDC

If TRUE, the DC Huffman table is set. If FALSE, the AC Huffman table is set.

lengthCounts

A pointer to an array of 16 length counts.

values

A pointer to an array of Huffman values.

Return Value

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

Discussion

This function lets you define a Huffman table to be used in future JPEG compression operations. Normally the JPEG image compressor components use the default Huffman tables as specified in sections K.3 through K.6 of the JPEG specification. You can use this function to override the default tables.

Special Considerations

This call is supported only by the Photo JPEG and Motion JPEG compressors. Only advanced programmers will need to use this function.

Version Notes

Introduced in QuickTime 3 or earlier.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`ImageCodec.h`

QTPhotoDefineQuantizationTable

Specifies a custom quantization table.

```
ComponentResult QTPhotoDefineQuantizationTable (
   ComponentInstance codec,
   short componentNumber,
   unsigned char *table
);
```

Parameters

codec

Identifies your connection to the image compressor component.

componentNumber

If 0, the luminance quantization table is set. If 1, the chrominance quantization table is set.

table

A pointer to an array of 64 quantization values.

Return Value

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

Discussion

By default, the JPEG compressors select quantization tables based on quality settings. This function lets you override these tables with tables of your own choice.

Special Considerations

This call is only supported by the Photo JPEG and Motion JPEG compressors. Only advanced programmers will need to use this function.

Version Notes

Introduced in QuickTime 3 or earlier.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`ImageCodec.h`

QTPhotoSetRestartInterval

Specifies the restart interval to use in future JPEG compression operations.

```
ComponentResult QTPhotoSetRestartInterval (
   ComponentInstance codec,
   unsigned short restartInterval
);
```

Parameters

codec

Identifies your connection to the image compressor component.

restartInterval

The new restart interval. Pass 0 to tell the compressor not to insert restart markers in the data stream.

Return Value

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

Discussion
By default, the JPEG compressor components do not insert restart markers in the compressed data stream unless the "optimize for streaming" setting is selected.

Special Considerations

This call is supported only by the Photo JPEG and Motion JPEG compressors. Only advanced programmers will need to use this function.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`ImageCodec.h`

QTPhotoSetSampling

Specifies the chrominance downsampling ratio to use in future JPEG compression operations.

```
ComponentResult QTPhotoSetSampling (
   ComponentInstance codec,
   short yH,
   short yV,
   short cbH,
   short cbV,
   short crH,
   short crV
);
```

Parameters

codec
: Identifies your connection to the image compressor component.

yH
: The number of horizontal luminance blocks to put in each macroblock.

yV
: The number of vertical luminance blocks to put in each macroblock.

cbH
: The number of horizontal chroma blue blocks to put in each macroblock.

cbV
: The number of vertical chroma blue blocks to put in each macroblock.

crH
: The number of horizontal chroma red blocks to put in each macroblock.

crV
: The number of vertical chroma red blocks to put in each macroblock.

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

Discussion
By default, the Photo JPEG compressor uses 4:1:1 chroma downsampling and the Motion JPEG compressors use 4:2:2 chroma downsampling for most quality settings. For `codecLosslessQuality`, both compressors disable chroma downsampling. Currently the only supported downsampling ratios are none (pass 1,1,1,1,1,1), 4:2:2 (pass 2,1,1,1,1,1) and 4:1:1 (pass 2,2,1,1,1,1).

Special Considerations

This call is supported only by the Photo JPEG and Motion JPEG compressors. Only advanced programmers will need to use this function.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`ImageCodec.h`

Callbacks

ComponentMPWorkFunctionProc

Undocumented

```
typedef ComponentResult (*ComponentMPWorkFunctionProcPtr) (void *globalRefCon,
ComponentMPWorkFunctionHeaderRecordPtr header);
```

If you name your function `MyComponentMPWorkFunctionProc`, you would declare it this way:

```
ComponentResult MyComponentMPWorkFunctionProc (
    void                                    *globalRefCon,
    ComponentMPWorkFunctionHeaderRecordPtr    header );
```

Parameters

`globalRefCon`

Undocumented

`header`

Pointer to a `ComponentMPWorkFunctionHeaderRecord` structure.

Return Value
See `Error Codes`. Your callback should return `noErr` if there is no error.

Declared In
`ImageCodec.h`

ImageCodecMPDrawBandProc

Undocumented

```
typedef ComponentResult (*ImageCodecMPDrawBandProcPtr) (void *refcon,
ImageSubCodecDecompressRecord *drp);
```

If you name your function `MyImageCodecMPDrawBandProc`, you would declare it this way:

```
ComponentResult MyImageCodecMPDrawBandProc (
    void                            *refcon,
    ImageSubCodecDecompressRecord    *drp );
```

Parameters

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.

drp

Pointer to an `ImageSubCodecDecompressRecord` structure.

Return Value

See `Error Codes`. Your callback should return `noErr` if there is no error.

Declared In

`ImageCodec.h`

ImageCodecTimeTriggerProc

Undocumented

```
typedef void (*ImageCodecTimeTriggerProcPtr) (void *refcon);
```

If you name your function `MyImageCodecTimeTriggerProc`, you would declare it this way:

```
void MyImageCodecTimeTriggerProc (
    void    *refcon );
```

Parameters

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.

Declared In

`ImageCodec.h`

Data Types

CDSequenceDataSource

Contains a linked list of all data sources for a decompression sequence.

```
struct CDSequenceDataSource {
     long                       recordSize;
     void *                     next;
     ImageSequence              seqID;
     ImageSequenceDataSource    sourceID;
     OSType                     sourceType;
     long                       sourceInputNumber;
     void *                     dataPtr;
     Handle                     dataDescription;
     long                       changeSeed;
     ICMConvertDataFormatUPP    transferProc;
     void *                     transferRefcon;
     long                       dataSize;
     QHdrPtr                    dataQueue;
     void *                     originalDataPtr;
     long                       originalDataSize;
     Handle                     originalDataDescription;
     long                       originalDataDescriptionSeed;
 };
```

Fields

`recordSize`

Discussion
The size of this structure.

`next`

Discussion
A pointer to the next source entry. If it is `NIL`, there are no more entries.

`seqID`

Discussion
The image sequence that this source is associated with.

`sourceID`

Discussion
The source reference identifying this source.

`sourceType`

Discussion
A four-character code describing how the input will be used. This value is passed to this parameter by `CDSequenceNewDataSource` when the source is created.

`sourceInputNumber`

Discussion
A value is passed to this parameter by `CDSequenceNewDataSource` when the source is created.

`dataPtr`

Discussion
A pointer to the actual source data.

`dataDescription`

Discussion
A handle to a data structure describing the data format. This is often a handle to an `ImageDescription` structure.

`changeSeed`

Discussion
An integer that is incremented each time the `dataPtr` field changes or that data that the `dataPtr` field points to changes. By remembering the `value` of this field and comparing to the value the next time the decompressor or compressor component is called, the component can determine if new data is present.

`transferProc`

Discussion
Reserved.

`transferRefcon`

Discussion
Reserved.

`dataSize`

Discussion
The size of the data pointed to by the `dataPtr` field.

`dataQueue`

Discussion
A pointer to a `QHdr` structure that contains a queue of `CDSequenceDataSourceQueueEntry` structures.

`originalDataPtr`

Discussion
The original value of `dataPtr`.

`originalDataSize`

Discussion
The original value of `dataSize`.

`originalDataDescription`

Discussion
The original value of `dataDescription`.

`originalDataDescriptionSeed`

Discussion
The original value of `changeSeed`.

Discussion
Because each data source is associated with a link to the next data source, a codec can access all data sources using this structure.

Version Notes
Fields from `dataQueue` onward were introduced in QuickTime 3.

Related Functions
`ImageCodecGetMaxCompressionSizeWithSources` (page 57)
`ImageCodecSourceChanged` (page 80)

Declared In
`ImageCodec.h`

CDSequenceDataSourcePtr

Represents a type used by the Image Codec API.

```
typedef CDSequenceDataSource * CDSequenceDataSourcePtr;
```

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

Declared In
`ImageCodec.h`

CodecCompressParams

Contains parameters that govern a compression operation.

```
struct CodecCompressParams {
     ImageSequence              sequenceID;
     ImageDescriptionHandle     imageDescription;
     Ptr                        data;
     long                       bufferSize;
     long                       frameNumber;
     long                       startLine;
     long                       stopLine;
     long                       conditionFlags;
     CodecFlags                 callerFlags;
     CodecCapabilities *        capabilities;
     ICMProgressProcRecord      progressProcRecord;
     ICMCompletionProcRecord    completionProcRecord;
     ICMFlushProcRecord         flushProcRecord;
     PixMap                     srcPixMap;
     PixMap                     prevPixMap;
     CodecQ                     spatialQuality;
     CodecQ                     temporalQuality;
     Fixed                      similarity;
     DataRateParamsPtr          dataRateParams;
     long                       reserved;
     UInt16                     majorSourceChangeSeed;
     UInt16                     minorSourceChangeSeed;
     CDSequenceDataSourcePtr    sourceData;
     long                       preferredPacketSizeInBytes;
     long                       requestedBufferWidth;
     long                       requestedBufferHeight;
     OSType                     wantedSourcePixelType;
     long                       compressedDataSize;
     UInt32                     taskWeight;
     OSType                     taskName;
 };
```

Fields
`sequenceID`

Discussion
Contains a unique sequence identifier. If the image to be compressed is part of a sequence, this field contains the sequence identifier that was assigned by `CompressSequenceBegin`. If the image is not part of a sequence, this field is set to 0.

`imageDescription`

Discussion
Contains a handle to the image description structure that describes the image to be compressed.

`data`

Discussion
Points to a location to receive the compressed image data. This is a 32-bit clean address. If there is not sufficient memory to store the compressed image, the application may choose to write the compressed data to mass storage during the compression operation. The `flushProcRecord` field identifies the data-unloading function that the application provides for this purpose. This field is used only by `ImageCodecBandCompress` (page 25).

`bufferSize`

Discussion
Contains the size of the buffer specified by the `data` field. Your component sets the value of the `bufferSize` field to the number of bytes of compressed data written into the buffer. Your component should not return more data than the buffer can hold; it should return a nonzero result code instead. This field is used only by `ImageCodecBandCompress` (page 25).

`frameNumber`

Discussion
Contains a frame identifier. Indicates the relative frame number within the sequence. The Image Compression Manager increments this value for each frame in the sequence. This field is used only by `ImageCodecBandCompress` (page 25).

`startLine`

Discussion
Contains the starting line for the band. This field indicates the starting line number for the band to be compressed. The line number refers to the pixel row in the image, starting from the top of the image. The first row is row number 0. This field is used only by `ImageCodecBandCompress` (page 25).

`stopLine`

Discussion
Contains the ending line for the band. This field indicates the ending line number for the band to be compressed. The line number refers to the pixel row in the image, starting from the top of the image. The first row in the image is row number 0. The image band includes the row specified by this field. So, to define a band that contains one row of pixels at the top of an image, you set the `startLine` field to 0 and the `stopLine` field to 1.

`conditionFlags`

Discussion
Contains flags (see below) that identify the condition under which your component has been called. This field is used only by `ImageCodecBandCompress` (page 25). In addition, these fields contain information about actions taken by your component. See these constants:

`codecConditionFirstBand`
`codecConditionLastBand`
`codecConditionCodecChangedMask`

`callerFlags`

Discussion
Flags that provide further control information. This field is used only by `ImageCodecBandCompress` (page 25). See these constants:

- `codecFlagUpdatePrevious`
- `codecFlagWasCompressed`
- `codecFlagUpdatePreviousComp`
- `codecFlagLiveGrab`

`capabilities`

Discussion
Points to a compressor capability structure. The Image Compression Manager uses this field to determine the capabilities of your compressor component. This field is used only by `ImageCodecPreCompress` (page 70).

`progressProcRecord`

Discussion
Contains an `ICMProgressProcRecord` structure. During the compression operation, your compressor may occasionally call a function that the application provides in order to report your progress. This field contains a structure that identifies the progress function. If the `progressProc` field in this structure is set to `NIL`, the application has not supplied a progress function. This field is used only by `ImageCodecBandCompress` (page 25).

`completionProcRecord`

Discussion
Contains an `ICMCompletionProcRecord` structure. This structure governs whether you perform the compression asynchronously. If the `completionProc` field in this structure is set to `NIL`, perform the compression synchronously. If this field is not `NIL`, it specifies an application completion function. Perform the compression asynchronously and call that completion function when your component is finished. If the `completionProc` field in this structure has a value of -1, perform the operation asynchronously but do not call the application's completion function. This field is used only by `ImageCodecBandCompress` (page 25).

`flushProcRecord`

Discussion
Contains an `ICMFlushProcRecord` structure. If there is not enough memory to store the compressed image, the application may provide a function that unloads some of the compressed data. This field contains a structure that identifies that data-unloading function. If the application did not provide a data-unloading function, the `flushProc` field in this structure is set to `NIL`. In this case, your component writes the entire compressed image into the memory location specified by the `data` field. The data-unloading function structure is used only by `ImageCodecBandCompress` (page 25).

`srcPixMap`

Discussion
Points to the image to be compressed. The image must be stored in a pixel map structure. The contents of this pixel map differ from a standard pixel map in two ways. First, the `rowBytes` field is a full 16-bit value; the high-order bit is not necessarily set to 1. Second, the `baseAddr` field must contain a 32-bit clean address. This field is used only by `ImageCodecBandCompress` (page 25).

`prevPixMap`

Discussion
Points to a pixel map containing the previous image. If the image to be compressed is part of a sequence that is being temporally compressed, this field defines the previous image for temporal compression. Your component should then use this previous image as the basis of comparison for the image to be compressed. If the `temporalQuality` field is set to 0, do not perform temporal compression. If the `codecFlagUpdatePrevious` flag or the `codecFlagUpdatePreviousComp` flag in the `flags` field is set to 1, update the previous image at the end of the compression operation. The contents of this pixel map differ from a standard pixel map in two ways. First, the `rowBytes` field is a full 16-bit value; the high-order bit is not necessarily set to 1. Second, the `baseAddr` field must contain a 32-bit clean address. This field is used only by `ImageCodecBandCompress` (page 25).

`spatialQuality`

Discussion
Specifies the desired compressed image quality. This field is used only by `ImageCodecBandCompress` (page 25).

`temporalQuality`

Discussion
Specifies the desired sequence temporal quality. This field governs the level of compression the application desires with respect to information in successive frames in the sequence. If this field is set to 0, do not perform temporal compression on this frame. This field is used only by `ImageCodecBandCompress` (page 25).

`similarity`

Discussion
Indicates the relative similarity between the frame just compressed and the previous frame when performing temporal compression. Fixed-point value, ranges from 0 (0x00000000), indicating a key frame, to 255 (0x00FF0000), indicating an identical frame that can be discarded without damage. If bad video would result from discarding a frame, the compressor should limit similarity to 254 (0x00FE0000). The Image Compression Manager may request a compressor to recompress a frame as a key frame if its similarity to its predecessor is very low (a value of 1 or 2, for example). The Image Compression Manager will not do this if the codecFlagLiveGrab flag is set, or if an asynchronous completion proc is supplied. This field is used only by `ImageCodecBandCompress` (page 25).

`dataRateParams`

Discussion
Points to the parameters used when performing data rate constraint.

`reserved`

Discussion
Reserved.

`majorSourceChangeSeed`

Discussion
Contains an integer value that is incremented each time a data source is added or removed. This provides a fast way for a codec to know when it needs to redetermine which data source inputs are available.

`minorSourceChangeSeed`

Discussion
Contains an integer value that is incremented each time a data source is added or removed, or the data contained in any of the data sources changes. This provides a way for a codec to know if the data available to it has changed.

`sourceData`

Discussion
Contains a pointer to a `CDSequenceDataSource` structure. This structure contains a linked list of all data sources. Because each data source contains a link to the next data source, a codec can access all data sources from this field.

`preferredPacketSizeInBytes`

Discussion
Specifies the preferred packet size for data.

`requestedBufferWidth`

Discussion
Specifies the the width of the image buffer to use, in pixels. For this value to be used, the `codecWantsSpecialScaling` flag in the `CodecCapabilities` structure must be set.

`requestedBufferHeight`

Discussion
Specifies the the height of the image buffer to use, in pixels. For this value to be used, the `codecWantsSpecialScaling` flag in the `CodecCapabilities` structure must be set.

`wantedSourcePixelType`

Discussion
Undocumented

`compressedDataSize`

Discussion
The size of the compressed image, in bytes. If this field is nonzero, it overrides the `dataSize` field of the `ImageDescription` structure. This provides a safer way for asynchronous compressors to return the size of the compressed frame data, because the `dataSize` field of `ImageDescription` may be referenced by an unlocked handle.

`taskWeight`

Discussion
The preferred weight for multiprocessing tasks implementing this operation. You should assign a value by means of the Mac OS function `MPSetTaskWeight`.

`taskName`

Discussion
The preferred type for multiprocessing tasks implementing this operation. You should assign a value by means of the Mac OS function `MPSetTaskType`.

Discussion
Compressor components accept the parameters that govern a compression operation in the form of the `CodecCompressParams` structure. This structure is used by `ImageCodecBandCompress` (page 25) and `ImageCodecPreCompress` (page 70).

Version Notes
Some of the fields in `CodecCompressParams` were added for various versions of QuickTime starting with version 2.1. See comments in the C interface file for details.

Related Functions
`ImageCodecBandCompress` (page 25)
`ImageCodecPreCompress` (page 70)

Declared In
`ImageCodec.h`

CodecDecompressParams

The basic parameter block that is passed to a decompressor.

```
struct CodecDecompressParams {
     ImageSequence              sequenceID;
     ImageDescriptionHandle     imageDescription;
     Ptr                        data;
     long                       bufferSize;
     long                       frameNumber;
     long                       startLine;
     long                       stopLine;
     long                       conditionFlags;
     CodecFlags                 callerFlags;
     CodecCapabilities *        capabilities;
     ICMProgressProcRecord      progressProcRecord;
     ICMCompletionProcRecord    completionProcRecord;
     ICMDataProcRecord          dataProcRecord;
     CGrafPtr                   port;
     PixMap                     dstPixMap;
     BitMapPtr                  maskBits;
     PixMapPtr                  mattePixMap;
     Rect                       srcRect;
     MatrixRecord *             matrix;
     CodecQ                     accuracy;
     short                      transferMode;
     ICMFrameTimePtr            frameTime;
     long                       reserved[1];
     SInt8                      matrixFlags;
     SInt8                      matrixType;
     Rect                       dstRect;
     UInt16                     majorSourceChangeSeed;
     UInt16                     minorSourceChangeSeed;
     CDSequenceDataSourcePtr    sourceData;
     RgnHandle                  maskRegion;
     OSType **                  wantedDestinationPixelTypes;
     long                       screenFloodMethod;
     long                       screenFloodValue;
     short                      preferredOffscreenPixelSize;
     ICMFrameTimeInfoPtr        syncFrameTime;
     Boolean                    needUpdateOnTimeChange;
     Boolean                    enableBlackLining;
     Boolean                    needUpdateOnSourceChange;
     Boolean                    pad;
     long                       unused;
     CGrafPtr                   finalDestinationPort;
     long                       requestedBufferWidth;
     long                       requestedBufferHeight;
     Rect                       displayableAreaOfRequestedBuffer;
     Boolean                    requestedSingleField;
     Boolean                    needUpdateOnNextIdle;
     Boolean                    pad2[2];
     fixed                      bufferGammaLevel;
     UInt32                     taskWeight;
     OSType                     taskName;
 };
```

Fields

`sequenceID`

Discussion
Contains the unique sequence identifier. If the image to be decompressed is part of a sequence, this field contains the sequence identifier that was assigned by `DecompressSequenceBegin`. If the image is not part of a sequence, this field is set to 0.

`imageDescription`

Discussion
Contains a handle to the `ImageDescription` that describes the image to be decompressed.

`data`

Discussion
Points to the compressed image data. This must be a 32-bit clean address. The `bufferSize` field indicates the size of this data buffer. If the entire compressed image does not fit in memory, the application should provide a data-loading function, identified by the `dataProc` field of the data-loading function structure stored in the `dataProcRecord` field. This field is used only by `ImageCodecBandDecompress` (page 26).

`bufferSize`

Discussion
Specifies the size of the image data buffer. This field is used only by `ImageCodecBandDecompress` (page 26).

`frameNumber`

Discussion
Contains a frame identifier. Indicates the relative frame number within the sequence. The Image Compression Manager increments this value for each frame in the sequence. This field is used only by `ImageCodecBandDecompress` (page 26).

`startLine`

Discussion
Specifies the starting line for the band. The line number refers to the pixel row in the image, starting from the top of the image. The first row in the image is row number 0. This field is used only by `ImageCodecBandDecompress` (page 26).

`stopLine`

Discussion
Specifies the ending line for the band. The line number refers to the pixel row in the image, starting from the top of the image. The first row is row number 0. The image band includes the row specified by this field. So, to define a band that contains one row of pixels at the top of an image, you set the `startLine` field to 0 and the `stopLine` field to 1. This field is used only by `ImageCodecBandDecompress` (page 26).

`conditionFlags`

Discussion
Contains flags (see below) that identify the condition under which your component has been called (in order to save the component some work). The flags in this field are passed to the component by `ImageCodecBandCompress` (page 25) and `ImageCodecPreDecompress` (page 70) when conditions change, to save it some work. In addition, these fields contain information about actions taken by your component. See these constants:

- `codecConditionFirstBand`
- `codecConditionLastBand`
- `codecConditionFirstFrame`
- `codecConditionNewDepth`
- `codecConditionNewTransform`
- `codecConditionNewSrcRect`
- `codecConditionNewMatte`
- `codecConditionNewTransferMode`
- `codecConditionNewClut`
- `codecConditionNewAccuracy`
- `codecConditionNewDestination`
- `codecConditionCodecChangedMask`
- `codecConditionFirstScreen`
- `codecConditionDoCursor`
- `codecConditionCatchUpDiff`
- `codecConditionMaskMayBeChanged`
- `codecConditionToBuffer`

`callerFlags`

Discussion
Contains flags (see below) that provide further control information. This field is used only by `ImageCodecBandCompress` (page 25). See these constants:

- `codecFlagUpdatePrevious`
- `codecFlagWasCompressed`
- `codecFlagUpdatePreviousComp`
- `codecFlagLiveGrab`

`capabilities`

Discussion
Points to a `CodecCapabilities` structure. The Image Compression Manager uses this parameter to determine the capabilities of your decompressor component. This field is used only by `ImageCodecPreDecompress` (page 70).

`progressProcRecord`

Discussion
Contains a `ICMProgressProcRecord` structure. During the decompression operation, your decompressor may occasionally call a function that the application provides in order to report your progress. This field contains a structure that identifies the progress function. If the `progressProc` field of this structure is set to `NIL`, the application did not provide a progress function. This field is used only by `ImageCodecBandDecompress` (page 26).

`completionProcRecord`

Discussion
Contains an `ICMCompletionProcRecord` structure. This field governs whether you perform the decompression asynchronously. If the `completionProc` field in this structure is set to `NIL`, perform the decompression synchronously. If this field is not `NIL`, it specifies an application completion function. Perform the decompression asynchronously and call that completion function when your component is finished. If this field has a value of -1, perform the operation asynchronously but do not call the application's completion function. This field is used only by `ImageCodecBandDecompress` (page 26).

`dataProcRecord`

Discussion
Contains an `ICMDataProcRecord` structure. If the data stream is not all in memory, your component may call an application function that loads more compressed data. This field contains a structure that identifies that data-loading function. If the application did not provide a data-loading function, the `dataProc` field in this structure is set to `NIL`. In this case, the entire image must be in memory at the location specified by the `data` field. This field is used only by `ImageCodecBandDecompress` (page 26).

`port`

Discussion
Points to the color graphics port that receives the decompressed image.

`dstPixMap`

Discussion
Points to the pixel map where the decompressed image is to be displayed. The `GDevice` global variable is set to the destination graphics device. The contents of this pixel map differ from a standard pixel map in two ways. First, the `rowBytes` field is a full 16-bit value; the high-order bit is not necessarily set to 1. Second, the `baseAddr` field must contain a 32-bit clean address.

`maskBits`

Discussion
Contains an update mask. If your component can mask result data, use this mask to indicate which pixels in the destination pixel map to update. Your component indicates whether it can mask with the `codecCanMask` flag in the `flags` field of the `CodecCapabilities` structure referred to by the `capabilities` field. This field is updated in response to the `ImageCodecPreDecompress` (page 70) request. If the mask has not changed since the last `ImageCodecBandDecompress` request, the `codecConditionCodecChangedMask` flag in the `conditionFlags` field is set to 0. This field is used only by `ImageCodecBandDecompress` (page 26).

`mattePixMap`

Discussion
Points to a pixel map that contains a blend matte. The matte can be defined at any supported pixel depth; the matte depth need not correspond to the source or destination depths. The matte must be in the coordinate system of the source image. If the application does not want to apply a blend matte, this field is set to `NIL`. The contents of this pixel map differ from a standard pixel map in two ways. First, the `rowBytes` field is a full 16-bit value; the high-order bit is not necessarily set to 1. Second, the `baseAddr` field must contain a 32-bit clean address. This field is used only by `ImageCodecBandDecompress` (page 26).

`srcRect`

Discussion
Points to a rectangle defining the portion of the image to decompress. This rectangle must lie within the boundary rectangle of the compressed image, which is defined by the `width` and height fields of the image description structure referred to by the `imageDescription` field.

`matrix`

Discussion
Points to a matrix structure that specifies how to transform the image during decompression.

`accuracy`

Discussion
Constant (see below) that specifies the accuracy desired in the decompressed image. Values for this parameter are on the same scale as compression quality; see `CompressImage`. See these constants:

- `codecMinQuality`
- `codecLowQuality`
- `codecNormalQuality`
- `codecHighQuality`
- `codecMaxQuality`
- `codecLosslessQuality`

`transferMode`

Discussion
Specifies the QuickDraw transfer mode for the operation; see `Graphics Transfer Modes`.

`frameTime`

Discussion
Contains a pointer to an `ICMFrameTimeRecord` structure. This structure contains a frame's time information for scheduled asynchronous decompression operations.

`matrixFlags`

Discussion
Flag (see below) specifying the transformation matrix. Set to 0 for no transformation. See these constants:

- `matrixFlagScale2x`
- `matrixFlagScale1x`
- `matrixFlagScaleHalf`

`matrixType`

Discussion
Contains the type of the transformation matrix, as returned by `GetMatrixType`.

`dstRect`

Discussion
The destination rectangle. It is the result of transforming the source rectangle (the `srcRect` parameter) by the transformation matrix (the `matrix` parameter).

`majorSourceChangeSeed`

Discussion
Contains an integer value that is incremented each time a data source is added or removed. This provides a fast way for a codec to know when it needs to redetermine which data source inputs are available.

`minorSourceChangeSeed`

Discussion
Contains an integer value that is incremented each time a data source is added or removed, or the data contained in any of the data sources changes. This provides a way for a codec to know if the data available to it has changed.

`sourceData`

Discussion
Contains a pointer to a `CDSequenceDataSource` structure. This structure contains a linked list of all data sources. Because each data source contains a link to the next data source, a codec can access all data sources from this field.

`maskRegion`

Discussion
If the `maskRegion` field is not `NIL`, it contains a QuickDraw region that is equivalent to the bit map contained in the `maskBits` field. For some codecs, using the QuickDraw region may be more convenient than the mask bit map.

`wantedDestinationPixelTypes`

Discussion
Filled in by the codec during the execution of `ImageCodecPreDecompress` (page 70). Contains a handle to a zero-terminated list of non-RGB pixels that the codec can decompress to. Leave set to `NIL` if the codec does not support non-RGB pixel spaces. The ICM copies this data structure, so it is up to the codec to dispose of it later. Since the predecompress call can be called often, it is suggested that codecs allocate this handle during the Open function and dispose of it during the Close function.

`screenFloodMethod`

Discussion
A constant (see below) for codecs that require key-color flooding. See these constants:

`kScreenFloodMethodNone`

`kScreenFloodMethodKeyColor`

`kScreenFloodMethodAlpha`

`screenFloodValue`

Discussion
If `screenFloodMethod` is `kScreenFloodMethodKeyColor`, contains the index of the color that should be used to flood the image area on screen when a refresh occurs. This is valid for both indexed and direct screen devices (e.g., for devices with 16 bit depth, it should contain the 5-5-5 RGB value). If `screenFloodMethod` is `kScreenFloodMethodAlpha`, contains the value that the alpha channel should be flooded with.

`preferredOffscreenPixelSize`

Discussion
Should be filled in `ImageCodecPreDecompress` (page 70) with the preferred depth of an offscreen buffer should the ICM have to create one. It is not guaranteed that an offscreen buffer will actually be of this depth. A codec should still be sure to specify what depths it can decompress to by using the `capabilities` field. A codec might use this field if if was capable of decompressing to several depths, but was faster decompressing to a particular depth.

`syncFrameTime`

Discussion
A pointer to an `ICMFrameTimeInfo` structure. This structure contains timing information about the display of the frame.

`needUpdateOnTimeChange`

Discussion
Undocumented

`enableBlackLining`

Discussion
If TRUE, indicates that the client has requested blacklining (displaying every other line of the image). Blacklining increases the speed of movie playback while decreasing the image quality.

`needUpdateOnSourceChange`

Discussion
Undocumented

`pad`

Discussion
Unused.

`unused`

Discussion
Unused.

`finalDestinationPort`

Discussion
Undocumented

`requestedBufferWidth`

Discussion
Specifies the width of the image buffer to use, in pixels. For this value to be used, the `codecWantsSpecialScaling` flag in `CodecCapabilities` must be set.

`requestedBufferHeight`

Discussion
Specifies the height of the image buffer to use, in pixels. For this value to be used, the `codecWantsSpecialScaling` flag in `CodecCapabilities` must be set.

`displayableAreaOfRequestedBuffer`

Discussion
This field can be used to prevent parts of the requested buffer from being displayed. When the `codecWantsSpecialScaling` flag is set, this rectangle can be filled in to indicate what portion of the requested buffer's width and height should be used. The buffer rectangle created by the requested buffer is always based at (0,0), so this coordinate system is also used by `displayableAreaOfRequestedBuffer`. If this field is not filled in, a default value of (0,0,0,0) is used, and the entire buffer is displayed. Use this field if you are experiencing edge problems with FlashPix images.

`requestedSingleField`

Discussion
Undocumented

`needUpdateOnNextIdle`

Discussion
Undocumented

`pad2`

Discussion
Unused.

`bufferGammaLevel`

Discussion
The gamma level of the data buffer.

`taskWeight`

Discussion
The preferred weight for multiprocessing tasks implementing this operation. You should assign a value by means of the Mac OS function `MPSetTaskWeight`.

`taskName`

Discussion
The preferred type for multiprocessing tasks implementing this operation. You should assign a value by means of the Mac OS function `MPSetTaskType`.

Discussion
The Image Compression Manager creates the decompression parameters structure, and your image decompressor component is required only to provide values for the `wantedDestinationPixelSize` and `wantedDestinationPixelTypes` fields of the structure. Your image decompressor component can also modify other fields if necessary. For example, if it can scale images, it must set the `codecCapabilityCanScale` flag in the `capabilities` field of the structure.

Version Notes
Some of the fields in `CodecDecompressParams` were added for various versions of QuickTime starting with version 2.1. See comments in the C interface file for details.

Related Functions
`ImageCodecBandDecompress` (page 26)
`ImageCodecBeginBand` (page 27)
`ImageCodecEffectBegin` (page 40)
`ImageCodecEffectSetup` (page 45)
`ImageCodecNewImageBufferMemory` (page 67)
`ImageCodecNewImageGWorld` (page 68)
`ImageCodecPreDecompress` (page 70)
`ImageCodecPreflight` (page 71)

Declared In
`ImageCodec.h`

ComponentMPWorkFunctionUPP

Represents a type used by the Image Codec API.

```
typedef STACK_UPP_TYPE(ComponentMPWorkFunctionProcPtr) ComponentMPWorkFunctionUPP;
```

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

Declared In
`Components.h`

EffectsFrameParams

Contains information about the current frame of a video effect.

```
struct EffectsFrameParams {
     ICMFrameTimeRecord    frameTime;
     long                  effectDuration;
     Boolean               doAsync;
     unsigned char         pad[3];
     EffectSourcePtr       source;
     void *                refCon;
 };
```

Fields

`frameTime`

Discussion
Timing data for the current frame. This structure includes information such as the total number of frames being rendered in this sequence, and the current frame number.

`effectDuration`

Discussion
The duration of a single effect frame.

`doAsync`

Discussion
This field contains TRUE if the effect can process asynchronously.

`pad`

Discussion
Unused.

`source`

Discussion
A pointer to the input sources; see the `EffectSource` structure.

`refCon`

Discussion
A pointer to storage for this instantiation of the effect.

Related Functions

`ImageCodecEffectBegin` (page 40)
`ImageCodecEffectCancel` (page 40)
`ImageCodecEffectRenderFrame` (page 43)

Declared In
`ImageCodec.h`

EffectsFrameParamsPtr

Represents a type used by the Image Codec API.

```
typedef EffectsFrameParams * EffectsFrameParamsPtr;
```

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

Declared In
`ImageCodec.h`

EffectSource

Provides data for the EffectsFrameParams structure.

```
struct EffectSource {
     long               effectType;
     Ptr                data;
     SourceData         source;
     EffectSourcePtr    next;
```

Fields

`effectType`

Discussion

The type of the effect or a default effect type constant (see below). Enter `kEffectRawSource` if the source is raw image compression manager data. See these constants:

`kEffectRawSource`

`kEffectGenericType`

`data`

Discussion

A pointer to the track data for the effect.

`source`

Discussion

The source itself.

`next`

Discussion

A pointer to the next source in the input chain.

`lastTranslatedFrameTime`

Discussion

The start frame time of last converted frame; this value may be -1.

`lastFrameDuration`

Discussion

The duration of the last converted frame; this value may be 0.

`lastFrameTimeScale`

Discussion

The time scale of this source frame; this field has meaning only if the `lastTranslatedFrameTime` and `lastFrameDuration` fields are valid.

Related Functions

`ImageCodecEffectConvertEffectSourceToFormat` (page 41)

Declared In

`ImageCodec.h`

EffectSourcePtr

Represents a type used by the Image Codec API.

```
typedef EffectSource * EffectSourcePtr;
```

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

Declared In
`ImageCodec.h`

gxPaths

Encapsulates a multiple-path geometry.

```
struct gxPaths {
     long      contours;
     gxPath    contour[1];
 };
```

Fields
`contours`

Discussion
The number of path contours.

`contour`

Discussion
The path contours; see `gxPath`.

Discussion
The contours field indicates the total number of contours (in other words, the total number of separate paths), and the `contour` field is an array that contains the path geometries. Since a `gxPaths` structure is of variable length and every element in it is of type `long`, you can define a path geometry as an array of long integer values.

Related Functions
`CurveCountPointsInPath` (page 14)
`CurveGetLength` (page 17)
`CurveGetNearestPathPoint` (page 17)
`CurveGetPathPoint` (page 18)
`CurveLengthToPoint` (page 20)
`CurvePathPointToLength` (page 22)
`CurveSetPathPoint` (page 23)

Declared In
`ImageCodec.h`

gxPoint

Defines a point in vector graphics.

```
struct gxPoint {
     Fixed    x;
     Fixed    y;
 };
```

Fields
x

Discussion
A horizontal distance. Greater values of the x field indicate distances further to the right.

y

Discussion
A vertical distance. Greater values of the y field indicate distances further down.

Discussion
The location of the origin depends on the context where you use the point; for example, it might be the upper-left corner of a view port. Notice that the x and y fields are of type Fixed. QuickDraw GX allows you to specify fractional coordinate positions.

Related Functions
CurveGetPathPoint (page 18)
CurveInsertPointIntoPath (page 19)
CurveSetPathPoint (page 23)

Declared In
ImageCodec.h

ImageCodecMPDrawBandUPP

Represents a type used by the Image Codec API.

```
typedef STACK_UPP_TYPE(ImageCodecMPDrawBandProcPtr) ImageCodecMPDrawBandUPP;
```

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

Declared In
ImageCodec.h

ImageCodecTimeTriggerUPP

Represents a type used by the Image Codec API.

```
typedef STACK_UPP_TYPE(ImageCodecTimeTriggerProcPtr) ImageCodecTimeTriggerUPP;
```

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

Declared In
ImageCodec.h

ImageSubCodecDecompressCapabilities

Returned by an image decompressor component in response to ImageCodecInitialize.

```
struct ImageSubCodecDecompressCapabilities {
     long       recordSize;
     long       decompressRecordSize;
     Boolean    canAsync;
     UInt8      pad0;
```

Fields

`recordSize`

Discussion

The size of this structure in bytes.

`decompressRecordSize`

Discussion

The size of the `ImageSubCodecDecompressRecord` structure that your image decompressor component requires. This structure is used to pass information from `ImageCodecBeginBand` (page 27) to `ImageCodecDrawBand` (page 38) and `ImageCodecEndBand` (page 46).

`canAsync`

Discussion

Specifies whether your image decompressor component can perform asynchronous scheduled decompression. This should be TRUE unless your image decompressor component calls functions that cannot be called during interrupt time.

`pad0`

Discussion

Unused.

Discussion

The first function call that your image decompressor component receives from the base image decompressor is always a call to `ImageCodecInitialize` (page 64). In response to this call, your image decompressor component returns an `ImageSubCodecDecompressCapabilities` structure that specifies its capabilities.

Related Functions

`ImageCodecInitialize` (page 64)

Declared In

`ImageCodec.h`

ImageSubCodecDecompressRecord

Contains information needed for decompressing a frame.

```
struct ImageSubCodecDecompressRecord {
     Ptr                    baseAddr;
     long                   rowBytes;
     Ptr                    codecData;
     ICMProgressProcRecord  progressProcRecord;
     ICMDataProcRecord      dataProcRecord;
     void *                 userDecompressRecord;
     UInt8                  frameType;
     UInt8                  pad[3];
     long                   priv[2];
 };
```

Fields

`baseAddr`

Discussion

The address of the destination pixel map, which includes adjustment for the offset. Note that if the bit depth of the pixel map is less than 8, your image decompressor component must adjust for the bit offset.

`rowBytes`

Discussion

The offset in bytes from one row of the destination pixel map to the next. The value of the `rowBytes` field must be less than 0x4000.

`codecData`

Discussion

A pointer to the data to be decompressed.

`progressProcRecord`

Discussion

An `ICMProgressProcRecord` structure that specifies a progress function. This function reports on the progress of a decompression operation. If there is no progress function, the Image Compression Manager sets the `progressProc` field in the `ICMProgressProcRecord` structure to `NIL`.

`dataProcRecord`

Discussion

An `ICMDataProcRecord` structure that specifies a data-loading function. If the data to be decompressed is not all in memory, your component can call this function to load more data. If there is no data-loading function, the Image Compression Manager sets the `dataProc` field in the `ICMDataProcRecord` structure to `NIL`, and the entire image must be in memory at the location specified by the `codecData` field of the `ImageSubCodecDecompressRecord` structure.

`userDecompressRecord`

Discussion

A pointer to storage for the decompression operation. The storage is allocated by the base image decompressor after it calls `ImageCodecInitialize` (page 64). The size of the storage is determined by the `decompressRecordSize` field of the `ImageSubCodecDecompressCapabilities` structure that is returned by `ImageCodecInitialize`. Your image decompressor component should use this storage to store any additional information needed about the frame in order to decompress it.

`frameType`

Discussion
A constant (see below) that indicates the `frame` type. See these constants:

```
kCodecFrameTypeUnknown
kCodecFrameTypeKey
kCodecFrameTypeDifference
kCodecFrameTypeDroppableDifference
```

`pad`

Discussion
Unused.

`priv`

Discussion
Private to QuickTime; do not use.

Related Functions
`ImageCodecBeginBand` (page 27)
`ImageCodecDrawBand` (page 38)
`ImageCodecEndBand` (page 46)
`ImageCodecMPDrawBandProc`

Declared In
`ImageCodec.h`

QTParameterValidationOptions

Represents a type used by the Image Codec API.

```
typedef long QTParameterValidationOptions;
```

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

Declared In
`ImageCodec.h`

SMPTEFlags

Represents a type used by the Image Codec API.

```
typedef long SMPTEFlags;
```

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

Declared In
`ImageCodec.h`

SMPTEFrameReference

Represents a type used by the Image Codec API.

```
typedef long SMPTEFrameReference;
```

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

Declared In
`ImageCodec.h`

SMPTEWipeType

Represents a type used by the Image Codec API.

```
typedef unsigned long SMPTEWipeType;
```

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

Declared In
`ImageCodec.h`

Constants

Codec Properties

Constants that represent the properties of codecs.

```
enum {
                                        /* The minimum data size for spooling in
or out data */
  codecMinimumDataSize          = 32768L
};
enum {
  codecConditionFirstBand       = 1L << 0,
  codecConditionLastBand        = 1L << 1,
  codecConditionFirstFrame      = 1L << 2,
  codecConditionNewDepth        = 1L << 3,
  codecConditionNewTransform    = 1L << 4,
  codecConditionNewSrcRect      = 1L << 5,
  codecConditionNewMask         = 1L << 6,
  codecConditionNewMatte        = 1L << 7,
  codecConditionNewTransferMode = 1L << 8,
  codecConditionNewClut         = 1L << 9,
  codecConditionNewAccuracy     = 1L << 10,
  codecConditionNewDestination  = 1L << 11,
  codecConditionFirstScreen     = 1L << 12,
  codecConditionDoCursor        = 1L << 13,
  codecConditionCatchUpDiff     = 1L << 14,
  codecConditionMaskMayBeChanged = 1L << 15,
  codecConditionToBuffer        = 1L << 16,
  codecConditionCodecChangedMask = 1L << 31
};
enum {
  codecInfoResourceType         = 'cdci', /* codec info resource type */
  codecInterfaceVersion       = 2    /* high word returned in component GetVersion
 */
};
enum {
  codecSuggestedBufferSentinel  = 'sent' /* codec public resource containing
suggested data pattern to put past end of data buffer */
};
enum {
  codecUsesOverlaySurface       = 1L << 0, /* codec uses overlay surface */
  codecImageBufferIsOverlaySurface = 1L << 1, /* codec image buffer is overlay
surface, the bits in the buffer are on the screen */
  codecSrcMustBeImageBuffer     = 1L << 2, /* codec can only source data from an
image buffer */
  codecImageBufferIsInAGPMemory = 1L << 4, /* codec image buffer is in AGP space,
 byte writes are OK */
  codecImageBufferIsInPCIMemory = 1L << 5, /* codec image buffer is across a PCI
bus; byte writes are bad */
  codecImageBufferMemoryFlagsValid = 1L << 6, /* set by
ImageCodecNewImageBufferMemory/NewImageGWorld to indicate that it set the AGP/PCI
 flags (supported in QuickTime 6.0 and later) */
  codecDrawsHigherQualityScaled = 1L << 7, /* codec will draw higher-quality image
 if it performs scaling (eg, wipe effect with border) */
  codecSupportsOutOfOrderDisplayTimes = 1L << 8, /* codec supports frames queued
in one order for display in a different order, eg, IPB content */
  codecSupportsScheduledBackwardsPlaybackWithDifferenceFrames = 1L << 9 /* codec
can use additional buffers to minimise redecoding during backwards playback */
};
```

Constants

`codecConditionFirstBand`
: An input flag that indicates if this is the first band in the frame. If this flag is set to 1, then your component is being called for the first time for the current frame.

Available in Mac OS X v10.0 and later.

Declared in `ImageCodec.h`.

`codecConditionLastBand`
: An input flag that indicates if this is the last band in the frame. If this flag is set to 1, then your component is being called for the last time for the current frame. If the `codecConditionFirstBand` flag is also set to 1, this is the only time the Image Compression Manager is calling your component for the current frame.

Available in Mac OS X v10.0 and later.

Declared in `ImageCodec.h`.

`codecConditionFirstFrame`
: An input flag that indicates that this is the first frame to be decompressed for this image sequence.

Available in Mac OS X v10.0 and later.

Declared in `ImageCodec.h`.

`codecConditionNewDepth`
: An input flag that indicates that the depth of the destination has changed for this image sequence.

Available in Mac OS X v10.0 and later.

Declared in `ImageCodec.h`.

`codecConditionNewTransform`
: An input flag that indicates that the transformation matrix has changed for this sequence.

Available in Mac OS X v10.0 and later.

Declared in `ImageCodec.h`.

`codecConditionNewSrcRect`
: An input flag that indicates that the source rectangle has changed for this sequence.

Available in Mac OS X v10.0 and later.

Declared in `ImageCodec.h`.

`codecConditionNewMatte`
: An input flag that indicates that the matte pixel map has changed for this sequence.

Available in Mac OS X v10.0 and later.

Declared in `ImageCodec.h`.

`codecConditionNewTransferMode`
: An input flag that indicates that the transfer mode has changed for this sequence.

Available in Mac OS X v10.0 and later.

Declared in `ImageCodec.h`.

`codecConditionNewClut`
: An input flag that indicates that the color lookup table has changed for this sequence.

Available in Mac OS X v10.0 and later.

Declared in `ImageCodec.h`.

`codecConditionNewAccuracy`
: An input flag that indicates to the component that the `accuracy` parameter has changed for this sequence.

  Available in Mac OS X v10.0 and later.

  Declared in `ImageCodec.h`.

`codecConditionNewDestination`
: An input flag that indicates to the component that the destination pixel map has changed for this sequence.

  Available in Mac OS X v10.0 and later.

  Declared in `ImageCodec.h`.

`codecConditionFirstScreen`
: Indicates when the codec is decompressing an image to the first of multiple screens. That is, if the decompressed image crosses multiple screens, then the codec can look at this flag to determine if this is the first time an image is being decompressed for each of the screens to which it is being decompressed. A codec that depends on the `maskBits` field of this structure being a valid `RgnHandle` on `ImageCodecPreDecompress` (page 70) needs to know that in this case it is not able to clip images since the region handle is only passed for the first of the screens; clipping would be incorrect for the subsequent screen for that image.

  Available in Mac OS X v10.0 and later.

  Declared in `ImageCodec.h`.

`codecConditionDoCursor`
: Set to 1 if the decompressor component should shield and unshield the cursor for the current decompression operation. This flag should be set only if the codec has indicated its ability to handle cursor shielding by setting the `codecCanShieldCursor` flag in the `capabilities` field during `ImageCodecPreDecompress` (page 70).

  Available in Mac OS X v10.0 and later.

  Declared in `ImageCodec.h`.

`codecConditionCatchUpDiff`
: Indicates if the current frame is a "catch-up" frame. Set this flag to 1 if the current frame is a catch-up frame. Note that you must also set the `codecFlagCatchUpDiff` flag to 1. This may be useful to decompressors that can drop frames when playback is falling behind.

  Available in Mac OS X v10.0 and later.

  Declared in `ImageCodec.h`.

`codecConditionMaskMayBeChanged`
: The Image Compression Manager has always included support for decompressors that could provide a bit mask of pixels that were actually drawn when a particular frame was decompressed. If a decompressor can provide a bit mask of pixels that changed, the Image Compression Manager transfers to the screen only the pixels that actually changed. QuickTime 2.1 extended this capability by adding this new condition flag. The decompressor should write back the mask only if this flag is set. This flag is used only by `ImageCodecFlush` (page 49).

  Available in Mac OS X v10.0 and later.

  Declared in `ImageCodec.h`.

`codecConditionToBuffer`
: Set to 1 if the current decompression operation is decompressing into an offscreen buffer.

  Available in Mac OS X v10.0 and later.

  Declared in `ImageCodec.h`.

`codecConditionCodecChangedMask`
: An output flag that indicates that the component has changed the mask bits. If your image decompressor component can mask decompressed images and if some of the image pixels should not be written to the screen, set to 0 the corresponding bits in the mask defined by the `maskBits` field in the decompression parameter structure. In addition, set this flag to 1. Otherwise, set this flag to 0.

Available in Mac OS X v10.0 and later.

Declared in `ImageCodec.h`.

`codecInfoResourceType`
: Codec info resource type.

Available in Mac OS X v10.0 and later.

Declared in `ImageCodec.h`.

`codecInterfaceVersion`
: High word returned in component `GetVersion`.

Available in Mac OS X v10.0 and later.

Declared in `ImageCodec.h`.

`codecSuggestedBufferSentinel`
: Codec public resource containing suggested data pattern to put past end of data buffer.

Available in Mac OS X v10.2 and later.

Declared in `ImageCodec.h`.

`codecUsesOverlaySurface`
: *Undocumented*

Available in Mac OS X v10.0 and later.

Declared in `ImageCodec.h`.

`codecImageBufferIsOverlaySurface`
: Indicates that the codec's image buffer is an overlay surface; the bits in the buffer appear on the screen.

Available in Mac OS X v10.0 and later.

Declared in `ImageCodec.h`.

`codecSrcMustBeImageBuffer`
: Indicates that the codec can accept source data only from an image buffer.

Available in Mac OS X v10.0 and later.

Declared in `ImageCodec.h`.

`codecImageBufferIsInAGPMemory`
: Indicates that the codec's image buffer resides in AGP address space and accepts byte writes.

Available in Mac OS X v10.0 and later.

Declared in `ImageCodec.h`.

`codecImageBufferIsInPCIMemory`
: Codec image buffer is across a PCI bus; byte writes are bad.

Available in Mac OS X v10.0 and later.

Declared in `ImageCodec.h`.

`codecImageBufferMemoryFlagsValid`
: Set by `ImageCodecNewImageBufferMemory` (page 67) or `NewImageGWorld` to indicate that the `codecImageBufferIsInAGPMemory` and `codecImageBufferIsInPCIPMemory` flags have been set correctly.

Available in Mac OS X v10.2 and later.

Declared in `ImageCodec.h`.

`codecDrawsHigherQualityScaled`
: Indicates that the codec will draw a higher quality image if it performs scaling; for example, while drawing a wipe effect with a border.

Available in Mac OS X v10.2 and later.

Declared in `ImageCodec.h`.

`codecSupportsOutOfOrderDisplayTimes`
: Codec supports frames queued in one order for display in a different order, for example IPB content.

Available in Mac OS X v10.3 and later.

Declared in `ImageCodec.h`.

Declared In
`ImageCodec.h`

ImageSubCodecDecompressRecord Values

Constants passed to ImageSubCodecDecompressRecord.

```
enum {
  kCodecFrameTypeUnknown        = 0,
  kCodecFrameTypeKey            = 1,
  kCodecFrameTypeDifference     = 2,
  kCodecFrameTypeDroppableDifference = 3
};
```

Constants
`kCodecFrameTypeUnknown`
: The frame type is unknown.

Available in Mac OS X v10.0 and later.

Declared in `ImageCodec.h`.

`kCodecFrameTypeKey`
: This is a key frame.

Available in Mac OS X v10.0 and later.

Declared in `ImageCodec.h`.

`kCodecFrameTypeDifference`
: This is a difference frame.

Available in Mac OS X v10.0 and later.

Declared in `ImageCodec.h`.

Declared In
`ImageCodec.h`

EffectSource Values

Constants passed to EffectSource.

```
enum {
  kEffectRawSource              = 0,    /* the source is raw image data*/
  kEffectGenericType            = 'geff' /* generic effect for combining others*/
};
```

Constants

`kEffectRawSource`

The source is raw Image Compression Manager data.

Available in Mac OS X v10.0 and later.

Declared in `ImageCodec.h`.

Declared In

`ImageCodec.h`

ImageCodecValidateParameters Values

Constants passed to ImageCodecValidateParameters.

```
enum {
  kParameterValidationNoFlags   = 0x00000000,
  kParameterValidationFinalValidation = 0x00000001
};
```

Declared In

`ImageCodec.h`

CodecDecompressParams Values

Constants passed to CodecDecompressParams.

```
enum {
  kScreenFloodMethodNone        = 0,
  kScreenFloodMethodKeyColor    = 1,
  kScreenFloodMethodAlpha       = 2
};
enum {
  matrixFlagScale2x             = 1L << 7,
  matrixFlagScale1x             = 1L << 6,
  matrixFlagScaleHalf           = 1L << 5
};
```

Constants

`kScreenFloodMethodNone`

No method; value is 0.

Available in Mac OS X v10.0 and later.

Declared in `ImageCodec.h`.

`kScreenFloodMethodKeyColor`
: Key color method; value is 1.

  Available in Mac OS X v10.0 and later.

  Declared in `ImageCodec.h`.

`kScreenFloodMethodAlpha`
: Alpha channel method; value is 2.

  Available in Mac OS X v10.0 and later.

  Declared in `ImageCodec.h`.

`matrixFlagScale2x`
: Double-scale; value is `1L<<7`.

  Available in Mac OS X v10.0 and later.

  Declared in `ImageCodec.h`.

`matrixFlagScale1x`
: Single-scale; value is `1L<<6`.

  Available in Mac OS X v10.0 and later.

  Declared in `ImageCodec.h`.

Declared In
`ImageCodec.h`

Document Revision History

This table describes the changes to *Image Codec Reference for QuickTime*.

Date	Notes
2006-05-23	New document, based on previously published material, that describes the API for QuickTime image codecs.

Document Revision History

Index

C

CDSequenceDataSource structure 91
CDSequenceDataSourcePtr data type 94
Codec Properties 115
CodecCompressParams structure 94
codecConditionCatchUpDiff constant 118
codecConditionCodecChangedMask constant 119
codecConditionDoCursor constant 118
codecConditionFirstBand constant 117
codecConditionFirstFrame constant 117
codecConditionFirstScreen constant 118
codecConditionLastBand constant 117
codecConditionMaskMayBeChanged constant 118
codecConditionNewAccuracy constant 118
codecConditionNewClut constant 117
codecConditionNewDepth constant 117
codecConditionNewDestination constant 118
codecConditionNewMatte constant 117
codecConditionNewSrcRect constant 117
codecConditionNewTransferMode constant 117
codecConditionNewTransform constant 117
codecConditionToBuffer constant 118
CodecDecompressParams structure 99
CodecDecompressParams Values 121
codecDrawsHigherQualityScaled constant 120
codecImageBufferIsInAGPMemory constant 119
codecImageBufferIsInPCIMemory constant 119
codecImageBufferIsOverlaySurface constant 119
codecImageBufferMemoryFlagsValid constant 120
codecInfoResourceType constant 119
codecInterfaceVersion constant 119
codecSrcMustBeImageBuffer constant 119
codecSuggestedBufferSentinel constant 119
codecSupportsOutOfOrderDisplayTimes constant 120
codecUsesOverlaySurface constant 119
ComponentMPWorkFunctionProc callback 90
ComponentMPWorkFunctionUPP data type 107
CurveAddAtomToVectorStream function 12
CurveAddPathAtomToVectorStream function 13
CurveAddZeroAtomToVectorStream function 14
CurveCountPointsInPath function 14
CurveCreateVectorStream function 15
CurveGetAtomDataFromVectorStream function 16
CurveGetLength function 17
CurveGetNearestPathPoint function 17
CurveGetPathPoint function 18
CurveInsertPointIntoPath function 19
CurveLengthToPoint function 20
CurveNewPath function 21
CurvePathPointToLength function 22
CurveSetPathPoint function 23

D

DisposeImageCodecDrawBandCompleteUPP function 24
DisposeImageCodecMPDrawBandUPP function 24
DisposeImageCodecTimeTriggerUPP function 25

E

EffectsFrameParams structure 107
EffectsFrameParamsPtr data type 108
EffectSource structure 109
EffectSource Values 121
EffectSourcePtr data type 109

G

gxPaths structure 110
gxPoint structure 110

I

ImageCodecBandCompress function 25

ImageCodecBandDecompress function 26
ImageCodecBeginBand function 27
ImageCodecBeginPass function 28
ImageCodecBusy function 29
ImageCodecCancelTrigger function 30
ImageCodecCompleteFrame function 30
ImageCodecCreateStandardParameterDialog function 31
ImageCodecDecodeBand function 32
ImageCodecDismissStandardParameterDialog function 33
ImageCodecDisposeImageGWorld function 34
ImageCodecDisposeMemory function 34
ImageCodecDITLEvent function 35
ImageCodecDITLInstall function 36
ImageCodecDITLItem function 36
ImageCodecDITLRemove function 37
ImageCodecDITLValidateInput function 38
ImageCodecDrawBand function 38
ImageCodecDroppingFrame function 39
ImageCodecEffectBegin function 40
ImageCodecEffectCancel function 40
ImageCodecEffectConvertEffectSourceToFormat function 41
ImageCodecEffectDisposeSMPTEFrame function 42
ImageCodecEffectGetSpeed function 42
ImageCodecEffectPrepareSMPTEFrame function 43
ImageCodecEffectRenderFrame function 43
ImageCodecEffectRenderSMPTEFrame function 44
ImageCodecEffectSetup function 45
ImageCodecEncodeFrame function 46
ImageCodecEndBand function 46
ImageCodecExtractAndCombineFields function 47
ImageCodecFlush function 49
ImageCodecFlushFrame function 50
ImageCodecGetBaseMPWorkFunction function 50
ImageCodecGetCodecInfo function 51
ImageCodecGetCompressedImageSize function 52
ImageCodecGetCompressionTime function 53
ImageCodecGetDecompressLatency function 54
ImageCodecGetDITLForSize function 55
ImageCodecGetMaxCompressionSize function 56
ImageCodecGetMaxCompressionSizeWithSources function 57
ImageCodecGetParameterList function 58
ImageCodecGetParameterListHandle function 59
ImageCodecGetSettings function 59
ImageCodecGetSettingsAsText function 60
ImageCodecGetSimilarity function 60
ImageCodecGetSourceDataGammaLevel function 61
ImageCodecHitTestData function 62
ImageCodecHitTestDataWithFlags function 63
ImageCodecInitialize function 64
ImageCodecIsImageDescriptionEquivalent function 64
ImageCodecIsStandardParameterDialogEvent function 65
ImageCodecMergeFloatingImageOntoWindow function 66
ImageCodecMPDrawBandProc callback 90
ImageCodecMPDrawBandUPP data type 111
ImageCodecNewImageBufferMemory function 67
ImageCodecNewImageGWorld function 68
ImageCodecNewMemory function 69
ImageCodecPreCompress function 70
ImageCodecPreDecompress function 70
ImageCodecPreflight function 71
ImageCodecPrepareToCompressFrames function 72
ImageCodecProcessBetweenPasses function 73
ImageCodecQueueStarting function 74
ImageCodecQueueStopping function 74
ImageCodecRemoveFloatingImage function 75
ImageCodecRequestGammaLevel function 76
ImageCodecRequestSettings function 77
ImageCodecScheduleFrame function 78
ImageCodecSetSettings function 78
ImageCodecSetTimeBase function 79
ImageCodecSetTimeCode function 80
ImageCodecSourceChanged function 80
ImageCodecStandardParameterDialogDoAction function 81
ImageCodecTimeTriggerProc callback 91
ImageCodecTimeTriggerUPP data type 111
ImageCodecTrimImage function 83
ImageCodecValidateParameters function 84
ImageCodecValidateParameters Values 121
ImageSubCodecDecompressCapabilities structure 112
ImageSubCodecDecompressRecord structure 112
ImageSubCodecDecompressRecord Values 120

K

kCodecFrameTypeDifference constant 120
kCodecFrameTypeKey constant 120
kCodecFrameTypeUnknown constant 120
kEffectRawSource constant 121
kScreenFloodMethodAlpha constant 122
kScreenFloodMethodKeyColor constant 122
kScreenFloodMethodNone constant 121

M

matrixFlagScale1x constant 122
matrixFlagScale2x constant 122

N

NewImageCodecDrawBandCompleteUPP function 85
NewImageCodecMPDrawBandUPP function 85
NewImageCodecTimeTriggerUPP function 86

Q

QTParameterValidationOptions data type 114
QTPhotoDefineHuffmanTable function 86
QTPhotoDefineQuantizationTable function 87
QTPhotoSetRestartInterval function 88
QTPhotoSetSampling function 89

S

SMPTEFlags data type 114
SMPTEFrameReference data type 115
SMPTEWipeType data type 115