Core Video Reference

Graphics & Imaging > Video

2007-03-22

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

Simultaneously published in the United States and Canada.

Contents

Core Video Reference 7

Overview 7
Functions by Task 7
- CVBuffer Functions 7
- CVDisplayLink Functions 8
- CVHostTime Functions 9
- CVImageBuffer Functions 9
- CVOpenGLBuffer Functions 10
- CVOpenGLBufferPool Functions 10
- CVOpenGLTexture Functions 10
- CVOpenGLTextureCache Functions 11
- CVPixelBuffer Functions 11
- CVPixelBufferPool Functions 12
- CVPixelFormatDescription Functions 13

Functions 13
- CVBufferGetAttachment 13
- CVBufferGetAttachments 14
- CVBufferPropagateAttachments 14
- CVBufferRelease 15
- CVBufferRemoveAllAttachments 15
- CVBufferRemoveAttachment 16
- CVBufferRetain 16
- CVBufferSetAttachment 17
- CVBufferSetAttachments 18
- CVDisplayLinkCreateWithActiveCGDisplays 18
- CVDisplayLinkCreateWithCGDisplay 19
- CVDisplayLinkCreateWithCGDisplays 20
- CVDisplayLinkCreateWithOpenGLDisplayMask 20
- CVDisplayLinkGetActualOutputVideoRefreshPeriod 21
- CVDisplayLinkGetCurrentCGDisplay 21
- CVDisplayLinkGetCurrentTime 22
- CVDisplayLinkGetNominalOutputVideoRefreshPeriod 22
- CVDisplayLinkGetOutputVideoLatency 23
- CVDisplayLinkGetTypeID 23
- CVDisplayLinkIsRunning 23
- CVDisplayLinkRelease 24
- CVDisplayLinkRetain 24
- CVDisplayLinkSetCurrentCGDisplay 25
- CVDisplayLinkSetCurrentCGDisplayFromOpenGLContext 25
- CVDisplayLinkSetOutputCallback 26
- CVDisplayLinkStart 27

CVDisplayLinkStop 27
CVDisplayLinkTranslateTime 28
CVGetCurrentHostTime 29
CVGetHostClockFrequency 29
CVGetHostClockMinimumTimeDelta 29
CVImageBufferGetCleanRect 30
CVImageBufferGetColorSpace 30
CVImageBufferGetDisplaySize 31
CVImageBufferGetEncodedSize 31
CVOpenGLBufferAttach 32
CVOpenGLBufferCreate 32
CVOpenGLBufferGetAttributes 33
CVOpenGLBufferGetTypeID 33
CVOpenGLBufferPoolCreate 34
CVOpenGLBufferPoolCreateOpenGLBuffer 34
CVOpenGLBufferPoolGetAttributes 35
CVOpenGLBufferPoolGetOpenGLBufferAttributes 35
CVOpenGLBufferPoolGetTypeID 36
CVOpenGLBufferPoolRelease 36
CVOpenGLBufferPoolRetain 37
CVOpenGLBufferRelease 37
CVOpenGLBufferRetain 37
CVOpenGLTextureCacheCreate 38
CVOpenGLTextureCacheCreateTextureFromImage 39
CVOpenGLTextureCacheFlush 39
CVOpenGLTextureCacheGetTypeID 40
CVOpenGLTextureCacheRelease 40
CVOpenGLTextureCacheRetain 41
CVOpenGLTextureGetCleanTexCoords 41
CVOpenGLTextureGetName 42
CVOpenGLTextureGetTarget 43
CVOpenGLTextureGetTypeID 43
CVOpenGLTextureIsFlipped 43
CVOpenGLTextureRelease 44
CVOpenGLTextureRetain 44
CVPixelBufferCreate 45
CVPixelBufferCreateResolvedAttributesDictionary 46
CVPixelBufferCreateWithBytes 46
CVPixelBufferCreateWithPlanarBytes 48
CVPixelBufferFillExtendedPixels 49
CVPixelBufferGetBaseAddress 49
CVPixelBufferGetBaseAddressOfPlane 50
CVPixelBufferGetBytesPerRow 51
CVPixelBufferGetBytesPerRowOfPlane 51
CVPixelBufferGetDataSize 52
CVPixelBufferGetExtendedPixels 52

CVPixelBufferGetHeight 53
CVPixelBufferGetHeightOfPlane 53
CVPixelBufferGetPixelFormatType 54
CVPixelBufferGetPlaneCount 54
CVPixelBufferGetTypeID 54
CVPixelBufferGetWidth 55
CVPixelBufferGetWidthOfPlane 55
CVPixelBufferIsPlanar 56
CVPixelBufferLockBaseAddress 56
CVPixelBufferPoolCreate 57
CVPixelBufferPoolCreatePixelBuffer 57
CVPixelBufferPoolGetAttributes 58
CVPixelBufferPoolGetPixelBufferAttributes 59
CVPixelBufferPoolGetTypeID 59
CVPixelBufferPoolRelease 59
CVPixelBufferPoolRetain 60
CVPixelBufferRelease 60
CVPixelBufferRetain 61
CVPixelBufferUnlockBaseAddress 61
CVPixelFormatDescriptionArrayCreateWithAllPixelFormatTypes 62
CVPixelFormatDescriptionCreateWithPixelFormatType 62
CVPixelFormatDescriptionRegisterDescriptionWithPixelFormatType 63
Callbacks 63
CVDisplayLinkOutputCallback 63
CVFillExtendedPixelsCallBack 65
CVPixelBufferReleaseBytesCallback 65
CVPixelBufferReleasePlanarBytesCallback 66
Data Types 67
CVBufferRef 67
CVDisplayLinkRef 67
CVFillExtendedPixelsCallbackData 67
CVImageBufferRef 68
CVOptionFlags 68
CVOpenGLBufferRef 69
CVOpenGLBufferPoolRef 69
CVOpenGLTextureRef 69
CVOpenGLTextureCacheRef 69
CVPixelBufferRef 70
CVPixelBufferPoolRef 70
CVReturn 70
CVSMPTETime 71
CVTime 71
CVTimeStamp 72
Constants 73
CVBuffer Attachment Keys 73
CVBuffer Attachment Modes 74

CVBuffer Attribute Keys 74
CVTime Constants 75
CVTime Values 75
CVTimeStamp Flags 75
Image Buffer Attachment Keys 77
OpenGL Buffer Attribute Keys 80
OpenGL Buffer Pool Attribute Keys 81
Pixel Buffer Attribute Keys 81
Pixel Buffer Pool Attribute Keys 83
Pixel Format Description Keys 83
SMPTE State Flags 86
SMPTE Time Types 87
Result Codes 88

Document Revision History 91

Index 93

Core Video Reference

Framework:	QuartzCore/QuartzCore.h
Companion guide	Core Video Programming Guide
Declared in	CVBase.h CVBuffer.h CVDisplayLink.h CVHostTime.h CVImageBuffer.h CVOpenGLBuffer.h CVOpenGLBufferPool.h CVOpenGLTexture.h CVOpenGLTextureCache.h CVPixelBuffer.h CVPixelBufferPool.h CVPixelFormatDescription.h CVReturn.h

Overview

Core Video is a new pipeline model for digital video in Mac OS X. Partitioning the processing into discrete steps makes it simpler for developers to access and manipulate individual frames without having to worry about translating between data types (QuickTime, OpenGL, and so on) or display synchronization issues.

Core Video is available in:

- Mac OS X v10.4 and later
- Mac OS X v10.3 when QuickTime 7.0 or later is installed

Functions by Task

CVBuffer Functions

Core Video buffer functions operate on all Core Video buffer types, including pixel buffers and OpenGL buffers, as well as OpenGL textures.

`CVBufferGetAttachment` (page 13)
Returns a specific attachment of a Core Video buffer.

`CVBufferGetAttachments` (page 14)
Returns all attachments of a Core Video buffer.

`CVBufferPropagateAttachments` (page 14)
Copies all propagatable attachments from one Core Video buffer to another.

`CVBufferRelease` (page 15)
Releases a Core Video buffer.

`CVBufferRemoveAllAttachments` (page 15)
Removes all attachments of a Core Video buffer.

`CVBufferRemoveAttachment` (page 16)
Removes a specific attachment of a Core Video buffer.

`CVBufferRetain` (page 16)
Retains a Core Video buffer.

`CVBufferSetAttachment` (page 17)
Sets or adds an attachment of a Core Video buffer.

`CVBufferSetAttachments` (page 18)
Sets a set of attachments for a Core Video buffer.

CVDisplayLink Functions

The main purpose of the CoreVideo display link to provide a separate high-priority thread to notify your application when a given display will need each frame. How often a frame is requested is based on the refresh rate of the display device currently associated with the display link. A CoreVideo display link is represented in code by the `CVDisplayLinkRef` type. The display link API uses the Core Foundation class system internally to provide reference counting behaviour and other useful properties.

`CVDisplayLinkCreateWithCGDisplay` (page 19)
Creates a display link for a single display.

`CVDisplayLinkCreateWithActiveCGDisplays` (page 18)
Creates a display link capable of being used with all active displays.

`CVDisplayLinkCreateWithCGDisplays` (page 20)
Creates a display link for an array of displays.

`CVDisplayLinkCreateWithOpenGLDisplayMask` (page 20)
Creates a display link from an OpenGL display mask.

`CVDisplayLinkGetActualOutputVideoRefreshPeriod` (page 21)
Retrieves the actual output refresh period of a display as measured by the host time base.

`CVDisplayLinkGetCurrentCGDisplay` (page 21)
Gets the current display associated with a display link.

`CVDisplayLinkGetCurrentTime` (page 22)
Retrieves the current (“now”) time of a given display link.

`CVDisplayLinkGetNominalOutputVideoRefreshPeriod` (page 22)
Retrieves the nominal refresh period of a display link.

`CVDisplayLinkGetOutputVideoLatency` (page 23)
Retrieves the nominal latency of a display link.

`CVDisplayLinkGetTypeID` (page 23)
Obtains the Core Foundation ID for the display link data type.

`CVDisplayLinkIsRunning` (page 23)
Indicates whether a given display link is running.

`CVDisplayLinkRelease` (page 24)
Releases a display link.

`CVDisplayLinkRetain` (page 24)
Retains a display link.

`CVDisplayLinkSetCurrentCGDisplay` (page 25)
Sets the current display of a display link.

`CVDisplayLinkSetCurrentCGDisplayFromOpenGLContext` (page 25)
Selects the display link most optimal for the current renderer of an OpenGL context.

`CVDisplayLinkSetOutputCallback` (page 26)
Set the renderer output callback function.

`CVDisplayLinkStart` (page 27)
Activates a display link.

`CVDisplayLinkStop` (page 27)
Stops a display link.

`CVDisplayLinkTranslateTime` (page 28)
Translates the time in the display link's time base from one representation to another.

CVHostTime Functions

`CVGetCurrentHostTime` (page 29)
Retrieves the current value of the host time base.

`CVGetHostClockFrequency` (page 29)
Retrieve the frequency of the host time base.

`CVGetHostClockMinimumTimeDelta` (page 29)
Retrieve the smallest possible increment in the host time base.

CVImageBuffer Functions

The functions in this section operate on Core Video buffers derived from the CVImageBuffer abstract type (`CVImageBufferRef`); specifically, pixel buffers, OpenGL buffers, and OpenGL textures.

`CVImageBufferGetCleanRect` (page 30)
Returns the source rectangle of a Core Video image buffer that represents the clean aperture of the buffer in encoded pixels.

`CVImageBufferGetColorSpace` (page 30)
Returns the color space of a Core Video image buffer.

`CVImageBufferGetDisplaySize` (page 31)
Returns the nominal output display size, in square pixels, of a Core Video image buffer.

`CVImageBufferGetEncodedSize` (page 31)
Returns the full encoded dimensions of a Core Video image buffer.

CVOpenGLBuffer Functions

The Core Video OpenGL buffer (type `CVOpenGLBufferRef` is a wrapper around the standard OpenGL pbuffer.

`CVOpenGLBufferAttach` (page 32)
Attaches an OpenGL context to a Core Video OpenGL buffer.

`CVOpenGLBufferCreate` (page 32)
Create a new Core Video OpenGL buffer that can be used for OpenGL rendering purposes

`CVOpenGLBufferGetAttributes` (page 33)
Obtains the attributes of a Core Video OpenGL buffer.

`CVOpenGLBufferGetTypeID` (page 33)
Obtains the Core Foundation type ID for the OpenGL buffer type.

`CVOpenGLBufferRelease` (page 37)
Releases a Core Video OpenGL buffer.

`CVOpenGLBufferRetain` (page 37)
Retains a Core Video OpenGL buffer.

CVOpenGLBufferPool Functions

An OpenGL buffer pool is a utility object for managing a set of OpenGL buffer objects for recycling.

`CVOpenGLBufferPoolCreate` (page 34)
Creates a new OpenGL buffer pool.

`CVOpenGLBufferPoolCreateOpenGLBuffer` (page 34)
Creates a new OpenGL buffer from an OpenGL buffer pool.

`CVOpenGLBufferPoolGetAttributes` (page 35)
Returns the pool attributes dictionary for an Open GL buffer pool.

`CVOpenGLBufferPoolGetOpenGLBufferAttributes` (page 35)
Returns the attributes of OpenGL buffers that will be created from a buffer pool.

`CVOpenGLBufferPoolGetTypeID` (page 36)
Obtains the Core Foundation ID for the OpenGL buffer pool type.

`CVOpenGLBufferPoolRelease` (page 36)
Releases an OpenGL buffer pool.

`CVOpenGLBufferPoolRetain` (page 37)
Retains an OpenGL buffer pool.

CVOpenGLTexture Functions

The Core Video OpenGL texture is a wrapper around the standard OpenGL texture type.

`CVOpenGLTextureGetCleanTexCoords` (page 41)
Returns the texture coordinates for the part of the image that should be displayed.

`CVOpenGLTextureGetName` (page 42)
Returns the texture target name of a CoreVideo OpenGL texture.

`CVOpenGLTextureGetTarget` (page 43)
Returns the texture target (for example, `GL_TEXTURE_2D`) of an OpenGL texture.

`CVOpenGLTextureGetTypeID` (page 43)
Obtains the Core Foundation ID for the Core Video OpenGL texture type.

`CVOpenGLTextureIsFlipped` (page 43)
Determines whether or not an OpenGL texture is flipped vertically.

`CVOpenGLTextureRelease` (page 44)
Releases a Core Video OpenGL texture.

`CVOpenGLTextureRetain` (page 44)
Retains a Core Video OpenGL texture.

CVOpenGLTextureCache Functions

`CVOpenGLTextureCacheCreate` (page 38)
Creates an OpenGL texture cache.

`CVOpenGLTextureCacheCreateTextureFromImage` (page 39)
Creates an OpenGL texture object from an existing image buffer.

`CVOpenGLTextureCacheFlush` (page 39)
Flushes the OpenGL texture cache.

`CVOpenGLTextureCacheGetTypeID` (page 40)
Returns the Core Foundation ID of the texture cache type.

`CVOpenGLTextureCacheRelease` (page 40)
Releases an OpenGL texture cache.

`CVOpenGLTextureCacheRetain` (page 41)
Retains an OpenGL texture cache.

CVPixelBuffer Functions

A pixel buffer stores an image in main memory

`CVPixelBufferCreate` (page 45)
Creates a single pixel buffer for a given size and pixel format.

`CVPixelBufferCreateResolvedAttributesDictionary` (page 46)
Takes an array of `CFDictionary` objects describing various pixel buffer attributes and tries to resolve them into a single dictionary.

`CVPixelBufferCreateWithBytes` (page 46)
Creates a pixel buffer for a given size and pixel format containing data specified by a memory location.

`CVPixelBufferCreateWithPlanarBytes` (page 48)
Creates a single pixel buffer in planar format for a given size and pixel format containing data specified by a memory location.

`CVPixelBufferFillExtendedPixels` (page 49)
Fills the extended pixels of the pixel buffer.

`CVPixelBufferGetBaseAddress` (page 49)
Returns the base address of the pixel buffer.

`CVPixelBufferGetBaseAddressOfPlane` (page 50)
Returns the base address of the plane at the specified plane index.

CVPixelBufferGetBytesPerRow (page 51)
Returns the number of bytes per row of the pixel buffer.

CVPixelBufferGetBytesPerRowOfPlane (page 51)
Returns the number of bytes per row for a plane at the specified index in the pixel buffer.

CVPixelBufferGetDataSize (page 52)
Returns the data size for contiguous planes of the pixel buffer.

CVPixelBufferGetExtendedPixels (page 52)
Returns the amount of extended pixel padding in the pixel buffer.

CVPixelBufferGetHeight (page 53)
Returns the height of the pixel buffer.

CVPixelBufferGetHeightOfPlane (page 53)
Returns the height of the plane at planeIndex in the pixel buffer.

CVPixelBufferGetPixelFormatType (page 54)
Returns the pixel format type of the pixel buffer.

CVPixelBufferGetPlaneCount (page 54)
Returns number of planes of the pixel buffer.

CVPixelBufferGetTypeID (page 54)
Returns the Core Foundation ID of the pixel buffer type.

CVPixelBufferGetWidth (page 55)
Returns the width of the pixel buffer.

CVPixelBufferGetWidthOfPlane (page 55)
Returns the width of the plane at a given index in the pixel buffer.

CVPixelBufferIsPlanar (page 56)
Determine if the pixel buffer is planar.

CVPixelBufferLockBaseAddress (page 56)
Locks the base address of the pixel buffer.

CVPixelBufferRelease (page 60)
Releases a pixel buffer.

CVPixelBufferRetain (page 61)
Retains a pixel buffer.

CVPixelBufferUnlockBaseAddress (page 61)
Unlocks the base address of the pixel buffer.

CVPixelBufferPool Functions

CVPixelBufferPoolCreate (page 57)
Creates a pixel buffer pool.

CVPixelBufferPoolCreatePixelBuffer (page 57)
Creates a pixel buffer from a pixel buffer pool.

CVPixelBufferPoolGetAttributes (page 58)
Returns the pool attributes dictionary for a pixel buffer pool.

CVPixelBufferPoolGetPixelBufferAttributes (page 59)
Returns the attributes of pixel buffers that will be created from this pool.

`CVPixelBufferPoolGetTypeID` (page 59)
Returns the Core Foundation ID of the pixel buffer pool type.

`CVPixelBufferPoolRelease` (page 59)
Releases a pixel buffer pool.

`CVPixelBufferPoolRetain` (page 60)
Retains a pixel buffer pool.

CVPixelFormatDescription Functions

Used only if you are defining a custom pixel format.

`CVPixelFormatDescriptionRegisterDescriptionWithPixelFormatType` (page 63)
Registers a pixel format description with Core Video.

`CVPixelFormatDescriptionCreateWithPixelFormatType` (page 62)
Creates a pixel format description from a given `OSType` identifier.

`CVPixelFormatDescriptionArrayCreateWithAllPixelFormatTypes` (page 62)
Returns all the pixel format descriptions known to Core Video.

Functions

CVBufferGetAttachment

Returns a specific attachment of a Core Video buffer.

```
CFTypeRef CVBufferGetAttachment (
   CVBufferRef buffer,
   CFStringRef key,
   CVAttachmentMode *attachmentMode
);
```

Parameters

buffer
The Core Video buffer whose attachment you want to obtain.

key
A key in the form of a Core Foundation string identifying the desired attachment.

attachmentMode
On return, `attachmentMode` points to the mode of the attachment. See "CVBuffer Attachment Modes" (page 74) for possible values. If the attachment mode is not defined, this parameter returns `NULL`.

Return Value
If found, the specified attachment.

Discussion
You can attach any Core Foundation object to a Core Video buffer to store additional information by calling `CVBufferSetAttachment` (page 17) or `CVBufferSetAttachments` (page 18).

You can find predefined attachment keys in "CVBuffer Attachment Keys" (page 73) and "Image Buffer Attachment Keys" (page 77).

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

Declared In
`CVBuffer.h`

CVBufferGetAttachments

Returns all attachments of a Core Video buffer.

```
CFDictionaryRef CVBufferGetAttachments (
   CVBufferRef buffer,
   CVAttachmentMode attachmentMode
);
```

Parameters

`buffer`

The Core Video buffer whose attachments you want to obtain.

`attachmentMode`

The mode of the attachments you want to obtain. See "CVBuffer Attachment Modes" (page 74) for possible values.

Return Value
A Core Foundation dictionary with all buffer attachments identified by keys. If no attachment is present, the dictionary is empty. Returns `NULL` for an invalid attachment mode.

Discussion
`CVBufferGetAttachments` is a convenience call that returns all attachments with their corresponding keys in a Core Foundation dictionary.

You can find predefined attachment keys in "CVBuffer Attachment Keys" (page 73) and "Image Buffer Attachment Keys" (page 77).

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

Declared In
`CVBuffer.h`

CVBufferPropagateAttachments

Copies all propagatable attachments from one Core Video buffer to another.

```
void CVBufferPropagateAttachments (
   CVBufferRef sourceBuffer,
   CVBufferRef destinationBuffer
);
```

Parameters

sourceBuffer

The buffer to copy attachments from.

destinationBuffer

The buffer to copy attachments to.

Discussion

`CVBufferPropagateAttachments` is a convenience call that copies all attachments with a mode of `kCVAttachmentMode_ShouldPropagate` from one buffer to another.

Availability

Available in Mac OS X v10.3 and later.

Declared In

`CVBuffer.h`

CVBufferRelease

Releases a Core Video buffer.

```
void CVBufferRelease (
   CVBufferRef buffer
);
```

Parameters

buffer

The Core Video buffer that you want to release.

Discussion

Like `CFRelease` `CVBufferRelease` decrements the retain count of a Core Video buffer. If that count consequently becomes zero the memory allocated to the object is deallocated and the object is destroyed. Unlike `CFRelease`, you can pass `NULL` to `CVBufferRelease` without causing a crash.

Availability

Available in Mac OS X v10.3 and later.

Related Sample Code

CaptureAndCompressIPBMovie

MovieVideoChart

VideoViewer

Declared In

`CVBuffer.h`

CVBufferRemoveAllAttachments

Removes all attachments of a Core Video buffer.

```
void CVBufferRemoveAllAttachments (
   CVBufferRef buffer
);
```

Parameters

buffer

The Core Video buffer whose attachments you want to remove.

Discussion

`CVBufferRemoveAllAttachments` removes all attachments of a buffer and decrements their reference counts.

Availability

Available in Mac OS X v10.3 and later.

Declared In

`CVBuffer.h`

CVBufferRemoveAttachment

Removes a specific attachment of a Core Video buffer.

```
void CVBufferRemoveAttachment (
   CVBufferRef buffer,
   CFStringRef key
);
```

Parameters

buffer

The Core Video buffer containing the attachment to remove.

key

A key in the form of a Core Foundation string identifying the desired attachment.

Discussion

`CVBufferRemoveAttachment` removes an attachment identified by a key. If found the attachment is removed and the retain count decremented.

You can find predefined attachment keys in "CVBuffer Attachment Keys" (page 73) and "Image Buffer Attachment Keys" (page 77).

Availability

Available in Mac OS X v10.3 and later.

Declared In

`CVBuffer.h`

CVBufferRetain

Retains a Core Video buffer.

```
CVBufferRef CVBufferRetain (
   CVBufferRef buffer
);
```

Parameters

buffer

The Core Video buffer that you want to retain.

Return Value

For convenience, the same Core Video buffer you wanted to retain.

Discussion

Like `CFRetain`, `CVBufferRetain` increments the retain count of a Core Video buffer. Unlike `CFRetain`, you can pass `NULL` to `CVBufferRetain` without causing a crash.

Availability

Available in Mac OS X v10.3 and later.

Related Sample Code

CaptureAndCompressIPBMovie

MovieVideoChart

Declared In

`CVBuffer.h`

CVBufferSetAttachment

Sets or adds an attachment of a Core Video buffer.

```
void CVBufferSetAttachment (
   CVBufferRef buffer,
   CFStringRef key,
   CFTypeRef value,
   CVAttachmentMode attachmentMode
);
```

Parameters

buffer

The Core Video buffer to which to add or set the attachment.

key

The key, in the form of a Core Foundation string, identifying the desired attachment.

value

The attachment in the form of a Core Foundation object. If this parameter is `NULL`, the function returns an error.

attachmentMode

Specifies the attachment mode for this attachment. See "CVBuffer Attachment Modes" (page 74) for possible values. Any given attachment key may exist in only one mode at a time.

Discussion

You can attach any Core Foundation object to a Core Video buffer to store additional information. If the key doesn't currently exist for the buffer object when you call this function, the new attachment will be added. If the key does exist, the existing attachment will be replaced. In both cases the retain count of the attachment will be incremented. The value can be any CFType. You can find predefined attachment keys in "CVBuffer Attachment Keys" (page 73) and "Image Buffer Attachment Keys" (page 77).

You can also set attachments when creating a buffer by specifying them in the `kCVBufferPropagatedAttachmentsKey` or `kkCVBufferNonpropagatedAttachmentsKey` attributes when creating the buffer.

To retrieve attachments, use the `CVBufferGetAttachment` (page 13) or `CVBufferGetAttachments` (page 14) functions.

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

Declared In
`CVBuffer.h`

CVBufferSetAttachments

Sets a set of attachments for a Core Video buffer.

```
void CVBufferSetAttachments (
   CVBufferRef buffer,
   CFDictionaryRef theAttachments,
   CVAttachmentMode attachmentMode
);
```

Parameters

buffer

The Core Video buffer to which to set the attachments.

theAttachments

The attachments to set, in the form of a Core Foundation dictionary array.

attachmentMode

Specifies which attachment mode is desired for this attachment. A particular attachment key may only exist in a single mode at a time.

Discussion
`CVBufferSetAttachments` is a convenience call that in turn calls `CVBufferSetAttachment` (page 17) for each key and value in the given dictionary. All key-value pairs must be in the root level of the dictionary.

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

Declared In
`CVBuffer.h`

CVDisplayLinkCreateWithActiveCGDisplays

Creates a display link capable of being used with all active displays.

```
CVReturn CVDisplayLinkCreateWithActiveCGDisplays (
   CVDisplayLinkRef *displayLinkOut
);
```

Parameters

displayLinkOut

On return, `displayLinkOut` points to the newly created display link.

Return Value
A Core Video result code. See "Result Codes" (page 88) for possible values.

Discussion
`CVDisplayLinkCreateWithActiveCGDisplays` determines the displays actively used by the host computer and creates a display link compatible with all of them. For most applications, calling this function is the most convenient way to create a display link. After creation, you can assign the display link to any active display by calling `CVDisplayLinkSetCurrentCGDisplay` (page 25).

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

Related Sample Code
QTQuartzPlayer
VideoViewer

Declared In
`CVDisplayLink.h`

CVDisplayLinkCreateWithCGDisplay

Creates a display link for a single display.

```
CVReturn CVDisplayLinkCreateWithCGDisplay (
   CGDirectDisplayID displayID,
   CVDisplayLinkRef *displayLinkOut
);
```

Parameters

displayID
The Core Graphics ID of the target display.

displayLinkOut
On return, `displayLinkOut` points to the newly created display link.

Return Value
A Core Video result code. See "Result Codes" (page 88) for possible values.

Discussion
Use this call to create a display link for a single display.

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

Related Sample Code
OpenGLCaptureToMovie
QTCoreImage101
QTCoreVideo102
QTCoreVideo201
QTCoreVideo301

Declared In
`CVDisplayLink.h`

CVDisplayLinkCreateWithCGDisplays

Creates a display link for an array of displays.

```
CVReturn CVDisplayLinkCreateWithCGDisplays (
   CGDirectDisplayID *displayArray,
   CFIndex count,
   CVDisplayLinkRef *displayLinkOut
);
```

Parameters

displayArray

A pointer to an array of Core Graphics display IDs representing all the active monitors you want to use with this display link.

count

The number of displays in the display array.

displayLisk

On return, `displayLinkOut` points to the newly created display link.

Return Value

A Core Video result code. See "Result Codes" (page 88) for possible values.

Discussion

Use this call to create a display link for a set of displays identified by the Core Graphics display IDs.

Availability

Available in Mac OS X v10.3 and later.

Declared In

`CVDisplayLink.h`

CVDisplayLinkCreateWithOpenGLDisplayMask

Creates a display link from an OpenGL display mask.

```
CVReturn CVDisplayLinkCreateWithOpenGLDisplayMask (
   CGOpenGLDisplayMask mask,
   CVDisplayLinkRef *displayLinkOut
);
```

Parameters

mask

The OpenGL display mask describing the available displays.

displayLinkOut

On return, `displayLinkOut` points to the newly created display link.

Return Value

A Core Video result code. See "Result Codes" (page 88) for possible values.

Discussion

Using this function avoids having to call the Core Graphics function `CGOpenGLDisplayMaskToDisplayID`.

Availability

Available in Mac OS X v10.3 and later.

Related Sample Code
CIVideoDemoGL
LiveVideoMixer2
LiveVideoMixer3

Declared In
`CVDisplayLink.h`

CVDisplayLinkGetActualOutputVideoRefreshPeriod

Retrieves the actual output refresh period of a display as measured by the host time base.

```
double CVDisplayLinkGetActualOutputVideoRefreshPeriod (
   CVDisplayLinkRef displayLink
);
```

Parameters

displayLink
The display link to get the refresh period from.

Return Value
A double-precision floating-point value representing the actual refresh period. This value may be zero if the device is not running or is otherwise unavailable.

Discussion
This call returns the actual output refresh period (in seconds) as computed relative to the host time base.

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

Declared In
`CVDisplayLink.h`

CVDisplayLinkGetCurrentCGDisplay

Gets the current display associated with a display link.

```
CGDirectDisplayID CVDisplayLinkGetCurrentCGDisplay (
   CVDisplayLinkRef displayLink
);
```

Parameters

displayLink
The display link whose current display you want obtain.

Return Value
A `CGDirectDisplayID` representing the current display.

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

Declared In
`CVDisplayLink.h`

CVDisplayLinkGetCurrentTime

Retrieves the current ("now") time of a given display link.

```
CVReturn CVDisplayLinkGetCurrentTime (
   CVDisplayLinkRef displayLink,
   CVTimeStamp *outTime
);
```

Parameters

displayLink

The display link whose current time you want to obtain.

outTime

A pointer to a `CVTimeStamp` structure. Note that yout must set the version in the structure (currently 0) before calling to indicate which version of the timestamp structure you want.

Return Value

A Core Video result code. See "Result Codes" (page 88) for possible values.

Discussion

You use this call to obtain the timestamp of the frame that is currently being displayed.

Availability

Available in Mac OS X v10.3 and later.

Declared In

`CVDisplayLink.h`

CVDisplayLinkGetNominalOutputVideoRefreshPeriod

Retrieves the nominal refresh period of a display link.

```
CVTime CVDisplayLinkGetNominalOutputVideoRefreshPeriod (
   CVDisplayLinkRef displayLink
);
```

Parameters

displayLink

The display link whose refresh period you want to obtain.

Return Value

A `CVTime` structure that holds the nominal refresh period. This value is indefinite if an invalid display link was specified.

Discussion

This call allows one to retrieve the device's ideal refresh period. For example, an NTSC output device might report 1001/60000 to represent the exact NTSC vertical refresh rate.

Availability

Available in Mac OS X v10.3 and later.

Declared In

`CVDisplayLink.h`

CVDisplayLinkGetOutputVideoLatency

Retrieves the nominal latency of a display link.

```
CVTime CVDisplayLinkGetOutputVideoLatency (
   CVDisplayLinkRef displayLink
);
```

Parameters

displayLink

The display link whose latency value you want to obtain.

Return Value

A `CVTime` structure that holds the latency value. This value may be indefinite.

Discussion

This call allows you to retrieve the device's built-in output latency. For example, an NTSC device with one frame of latency might report back 1001/30000 or 2002/60000.

Availability

Available in Mac OS X v10.3 and later.

Declared In

`CVDisplayLink.h`

CVDisplayLinkGetTypeID

Obtains the Core Foundation ID for the display link data type.

```
CFTypeID CVDisplayLinkGetTypeID (
   void
);
```

Return Value

The Core Foundation ID for this type.

Availability

Available in Mac OS X v10.3 and later.

Declared In

`CVDisplayLink.h`

CVDisplayLinkIsRunning

Indicates whether a given display link is running.

```
Boolean CVDisplayLinkIsRunning (
   CVDisplayLinkRef displayLink
);
```

Parameters

displayLink

The display link whose run state you want to determine.

Return Value
Returns `true` if the display link is running, `false` otherwise.

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

Related Sample Code
CIVideoDemoGL
QTCoreImage101
QTCoreVideo102
QTCoreVideo201
QTCoreVideo301

Declared In
`CVDisplayLink.h`

CVDisplayLinkRelease

Releases a display link.

```
void CVDisplayLinkRelease (
   CVDisplayLinkRef displayLink
);
```

Parameters

displayLink
The display link to release. This function is *NULL*-safe.

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

Related Sample Code
LiveVideoMixer3
QTCoreImage101
QTCoreVideo102
QTCoreVideo201
QTCoreVideo301

Declared In
`CVDisplayLink.h`

CVDisplayLinkRetain

Retains a display link.

```
CVDisplayLinkRef CVDisplayLinkRetain (
   CVDisplayLinkRef displayLink
);
```

Parameters

displayLink
The display link to retain. This function is *NULL*-safe.

Return Value
For convenience, this function returns the retained display link if successful.

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

Declared In
`CVDisplayLink.h`

CVDisplayLinkSetCurrentCGDisplay

Sets the current display of a display link.

```
CVReturn CVDisplayLinkSetCurrentCGDisplay (
   CVDisplayLinkRef displayLink,
   CGDirectDisplayID displayID
);
```

Parameters

displayLink
The display link whose display you want to set.

displayID
The ID of the display to be set.

Return Value
A Core Video result code. See "Result Codes" (page 88) for possible values.

Discussion
Although it is safe to call this function on a running display link, a discontinuity may appear in the video timestamp.

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

Related Sample Code
LiveVideoMixer3
QTCoreImage101
QTCoreVideo102
QTCoreVideo201
QTCoreVideo301

Declared In
`CVDisplayLink.h`

CVDisplayLinkSetCurrentCGDisplayFromOpenGLContext

Selects the display link most optimal for the current renderer of an OpenGL context.

```
CVReturn CVDisplayLinkSetCurrentCGDisplayFromOpenGLContext (
   CVDisplayLinkRef displayLink,
   CGLContextObj cglContext,
   CGLPixelFormatObj cglPixelFormat
);
```

Parameters

displayLink

The display link for which you want to set the current display.

cglContext

The OpenGL context to retrieve the current renderer from.

cglPixelFormat

The OpenGL pixel format used to create the passed-in OpenGL context.

Return Value

A Core Video result code. See "Result Codes" (page 88) for possible values.

Discussion

This function chooses the display with the lowest refresh rate.

Availability

Available in Mac OS X v10.3 and later.

Related Sample Code

VideoViewer

Declared In

`CVDisplayLink.h`

CVDisplayLinkSetOutputCallback

Set the renderer output callback function.

```
CVReturn CVDisplayLinkSetOutputCallback (
   CVDisplayLinkRef displayLink,
   CVDisplayLinkOutputCallback callback,
   void *userInfo
);
```

Parameters

displayLink

The display link whose output callback you want to set.

callback

The callback function to set for this display link. See `CVDisplayLinkOutputCallback` (page 63) for more information about implementing this function.

userInfo

A pointer to user data.

Return Value

A Core Video result code. See "Result Codes" (page 88) for possible values.

Discussion

The display link invokes this callback whenever it wants you to output a frame.

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

Related Sample Code
LiveVideoMixer3
QTCoreImage101
QTCoreVideo102
QTCoreVideo201
QTCoreVideo301

Declared In
`CVDisplayLink.h`

CVDisplayLinkStart

Activates a display link.

```
CVReturn CVDisplayLinkStart (
   CVDisplayLinkRef displayLink
);
```

Parameters

displayLink
The display link to activate.

Return Value
A Core Video result code. See "Result Codes" (page 88) for possible values.

Discussion
Calling this function starts the display link thread, which then periodically calls back to your application to request that you display frames. If the specified display link is already running, `CVDisplayLinkStart` returns an error.

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

Related Sample Code
QTCoreVideo102
QTCoreVideo103
QTCoreVideo201
QTCoreVideo202
QTCoreVideo301

Declared In
`CVDisplayLink.h`

CVDisplayLinkStop

Stops a display link.

```
CVReturn CVDisplayLinkStop (
   CVDisplayLinkRef displayLink
);
```

Parameters

displayLink

The display link to stop.

Return Value

A Core Video result code. See "Result Codes" (page 88) for possible values.

Discussion

If the specified display link is already stopped, `CVDisplayLinkStop` returns an error.

In Mac OS X v.10.4 and later, the display link thread is automatically stopped if the user employs Fast User Switching. The display link is restarted when switching back to the original user.

Availability

Available in Mac OS X v10.3 and later.

Related Sample Code

QTCoreImage101

QTCoreVideo102

QTCoreVideo201

QTCoreVideo202

QTCoreVideo301

Declared In

`CVDisplayLink.h`

CVDisplayLinkTranslateTime

Translates the time in the display link's time base from one representation to another.

```
CVReturn CVDisplayLinkTranslateTime (
   CVDisplayLinkRef displayLink,
   const CVTimeStamp *inTime,
   CVTimeStamp *outTime
);
```

Parameters

displayLink

The display link whose time base should be used to do the translation.

inTime

A pointer to a `CVTimeStamp` structure containing the source time to translate.

outTime

A pointer to a `CVTimeStamp` structure into which the target time is written. Before calling, you must set the version field (currently `0`) to indicate which version of the structure you want. You should also set the `flags` field to specify which representations to translate to.

Return Value

A Core Video result code. See "Result Codes" (page 88) for possible values.

Discussion
Note that the display link has to be running for this call to succeed.

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

Declared In
`CVDisplayLink.h`

CVGetCurrentHostTime

Retrieves the current value of the host time base.

```
uint64_t CVGetCurrentHostTime
```

Return Value
The current host time.

Discussion
In Mac OS X, the host time base for CoreVideo and CoreAudio are identical, so the values returned from either API can be used interchangeably.

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

Declared In
`CVHostTime.h`

CVGetHostClockFrequency

Retrieve the frequency of the host time base.

```
double CVGetHostClockFrequency
```

Return Value
The current host frequency.

Discussion
In Mac OS X, the host time base for CoreVideo and CoreAudio are identical, and the values returned from either API can be used interchangeably.

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

Declared In
`CVHostTime.h`

CVGetHostClockMinimumTimeDelta

Retrieve the smallest possible increment in the host time base.

```
uint32_t CVGetHostClockMinimumTimeDelta
```

Return Value
The smallest valid increment in the host time base.

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

Declared In
`CVHostTime.h`

CVImageBufferGetCleanRect

Returns the source rectangle of a Core Video image buffer that represents the clean aperture of the buffer in encoded pixels.

```
CGRect CVImageBufferGetCleanRect (
   CVImageBufferRef imageBuffer
);
```

Parameters
`imageBuffer`
The image buffer that you want to retrieve the display size from.

Return Value
A `CGRect` structure returning the nominal display size of the buffer. Returns a rectangle of zero size if called with either a non-`CVImageBufferRef` type or `NULL`.

Discussion
The clean aperture size is smaller than the full size of the image. For example, an NTSC DV frame would return a `CGRect` structure with an origin of (8,0) and a size of (704,480). Note that the origin of this rectangle is always in the lower-left corner. This is the same coordinate system as that used by Quartz and Core Image.

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

Declared In
`CVImageBuffer.h`

CVImageBufferGetColorSpace

Returns the color space of a Core Video image buffer.

```
CGColorSpaceRef CVImageBufferGetColorSpace (
   CVImageBufferRef imageBuffer
);
```

Parameters
`imageBuffer`
The image buffer that you want to retrieve the color space from.

Return Value
The color space of the buffer. Returns `NULL` if called with either a non-`CVImageBufferRef` type or `NULL`.

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

Declared In
`CVImageBuffer.h`

CVImageBufferGetDisplaySize

Returns the nominal output display size, in square pixels, of a Core Video image buffer.

```
CGSize CVImageBufferGetDisplaySize (
   CVImageBufferRef imageBuffer
);
```

Parameters

`imageBuffer`

The image buffer that you want to retrieve the display size from.

Return Value
A `CGSize` structure defining the nominal display size of the buffer Returns zero size if called with a non-`CVImageBufferRef` type or `NULL`.

Discussion
For example, for an NTSC DV frame this would be 640 x 480.

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

Declared In
`CVImageBuffer.h`

CVImageBufferGetEncodedSize

Returns the full encoded dimensions of a Core Video image buffer.

```
CGSize CVImageBufferGetEncodedSize (
   CVImageBufferRef imageBuffer
);
```

Parameters

`imageBuffer`

The image buffer that you want to retrieve the encoded size from.

Return Value
A `CGSize` structure defining the full encoded size of the buffer. Returns zero size if called with either a non-`CVImageBufferRef` type or `NULL`.

Discussion
For example, for an NTSC DV frame, the encoded size would be 720 x 480. Note: When creating a Core Image image from a Core Video image buffer, you use this call to retrieve the image size.

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

Declared In
`CVImageBuffer.h`

CVOpenGLBufferAttach

Attaches an OpenGL context to a Core Video OpenGL buffer.

```
CVReturn CVOpenGLBufferAttach (
   CVOpenGLBufferRef openGLBuffer,
   CGLContextObj cglContext,
   GLenum face,
   GLint level,
   GLint screen
);
```

Parameters

openGLBuffer
The buffer you want to attach an OpenGL context to.

cglContext
The OpenGL context you want to attach.

face
The OpenGL face enumeration (`0` for noncube maps.)

level
The mipmap level for drawing in the OpenGL context. This value cannot exceed the maximum mipmap level for this buffer.

screen
The virtual screen number you want to use for this context.

Return Value
A Core Video result code. See "Result Codes" (page 88) for possible values.

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

Declared In
`CVOpenGLBuffer.h`

CVOpenGLBufferCreate

Create a new Core Video OpenGL buffer that can be used for OpenGL rendering purposes

```
CVReturn CVOpenGLBufferCreate (
   CFAllocatorRef allocator,
   size_t width,
   size_t height,
   CFDictionaryRef attributes,
   CVOpenGLBufferRef *bufferOut
);
```

Parameters

allocator
The allocator to use to create the Core Video OpenGL buffer. Pass `NULL` to specify the default allocator.

width

The width of the buffer in pixels.

height

The height of the buffer in pixels.

attributes

A Core Foundation dictionary containing other desired attributes of the buffer (texture target, internal format, max mipmap level, etc.). May be `NULL`. The following attribute values are assumed if you do not explicitly define them:

- `kCVOpenGLBufferTarget` = `GL_TEXTURE_RECTANGLE_EXT`
- `kCVOpenGLBufferInternalFormat` = `GL_RGBA`
- `kCVOpenGLBufferMaximumMipmapLevel` = 0

bufferOut

On return, `bufferOut` points to the newly created OpenGL buffer.

Return Value
A Core Video result code. See "Result Codes" (page 88) for possible values.

Discussion

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

Declared In
`CVOpenGLBuffer.h`

CVOpenGLBufferGetAttributes

Obtains the attributes of a Core Video OpenGL buffer.

```
CFDictionaryRef CVOpenGLBufferGetAttributes (
   CVOpenGLBufferRef openGLBuffer
);
```

Parameters

openGLBuffer

The OpenGL buffer whose attributes you want to obtain.

Return Value
A Core Foundation dictionary containing the OpenGL buffer attributes, or `NULL` if no attributes exist.

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

Declared In
`CVOpenGLBuffer.h`

CVOpenGLBufferGetTypeID

Obtains the Core Foundation type ID for the OpenGL buffer type.

```
CFTypeID CVOpenGLBufferGetTypeID (
   void
);
```

Return Value
The Core Foundation ID for this data type.

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

Declared In
`CVOpenGLBuffer.h`

CVOpenGLBufferPoolCreate

Creates a new OpenGL buffer pool.

```
CVReturn CVOpenGLBufferPoolCreate (
   CFAllocatorRef allocator,
   CFDictionaryRef poolAttributes,
   CFDictionaryRef openGLBufferAttributes,
   CVOpenGLBufferPoolRef *poolOut
);
```

Parameters

allocator
The allocator to use for allocating this buffer pool. Pass `NULL` to specify the default allocator.

poolAttributes
A Core Foundation dictionary containing the attributes to be used for the pool itself.

openGLBufferAttributes
A Core Foundation dictionary containing the attributes to be used for creating new OpenGL buffers within the pool.

poolOut
On return, `poolOut` points to the new OpenGL buffer pool.

Return Value
A Core Video result code. See "Result Codes" (page 88) for possible values.

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

Declared In
`CVOpenGLBufferPool.h`

CVOpenGLBufferPoolCreateOpenGLBuffer

Creates a new OpenGL buffer from an OpenGL buffer pool.

```
CVReturn CVOpenGLBufferPoolCreateOpenGLBuffer (
   CFAllocatorRef allocator,
   CVOpenGLBufferPoolRef openGLBufferPool,
   CVOpenGLBufferRef *openGLBufferOut
);
```

Parameters

allocator

The allocator to use for creating the buffer. May be `NULL` to specify the default allocator.

openGLBufferPool

The OpenGL buffer pool that should create the new OpenGL buffer.

openGLBufferOut

On return, `OpenGLBufferOut` points to the new OpenGL buffer.

Return Value
A Core Video result code. See "Result Codes" (page 88) for possible values.

Discussion
The function creates a new OpenGL buffer using the OpenGL buffer attributes specified in the `CVOpenGLBufferPoolCreate` (page 34) call. This buffer has default attachments as specified in the `openGLBufferAttributes` parameter of `CVOpenGLBufferPoolCreate` (page 34) (using either the `kCVBufferPropagatedAttachmentsKey` or `kCVBufferNonPropagatedAttachmentsKey` attributes).

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

Declared In
`CVOpenGLBufferPool.h`

CVOpenGLBufferPoolGetAttributes

Returns the pool attributes dictionary for an Open GL buffer pool.

```
CFDictionaryRef CVOpenGLBufferPoolGetAttributes (
   CVOpenGLBufferPoolRef pool
);
```

Parameters

pool

The OpenGL buffer pool to retrieve the attributes from.

Return Value
The buffer-pool attributes Core Foundation dictionary, or `NULL` on failure.

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

Declared In
`CVOpenGLBufferPool.h`

CVOpenGLBufferPoolGetOpenGLBufferAttributes

Returns the attributes of OpenGL buffers that will be created from a buffer pool.

```
CFDictionaryRef CVOpenGLBufferPoolGetOpenGLBufferAttributes (
   CVOpenGLBufferPoolRef pool
);
```

Parameters

pool

The OpenGL buffer pool to retrieve the attributes from.

Return Value
The OpenGL buffer attributes Core Foundation dictionary, or `NULL` on failure.

Discussion
You can use this function to obtain information about the OpenGL buffers that will be created from the buffer pool.

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

Declared In
`CVOpenGLBufferPool.h`

CVOpenGLBufferPoolGetTypeID

Obtains the Core Foundation ID for the OpenGL buffer pool type.

```
CFTypeID CVOpenGLBufferPoolGetTypeID (
   void
);
```

Return Value
The Core Foundation ID for this data type.

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

Declared In
`CVOpenGLBufferPool.h`

CVOpenGLBufferPoolRelease

Releases an OpenGL buffer pool.

```
void CVOpenGLBufferPoolRelease (
   CVOpenGLBufferPoolRef openGLBufferPool
);
```

Parameters

openGLBufferPool

The OpenGL buffer pool that you want to release.

Discussion
This function is equivalent to `CFRelease`, but `NULL` safe.

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

Declared In
`CVOpenGLBufferPool.h`

CVOpenGLBufferPoolRetain

Retains an OpenGL buffer pool.

```
CVOpenGLBufferPoolRef CVOpenGLBufferPoolRetain (
   CVOpenGLBufferPoolRef openGLBufferPool
);
```

Parameters

openGLBufferPool
The OpenGL buffer pool that you want to retain.

Return Value
For convenience, the same buffer pool object you wanted to retain.

Discussion
This function is equivalent to `CFRetain`, but `NULL` safe.

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

Declared In
`CVOpenGLBufferPool.h`

CVOpenGLBufferRelease

Releases a Core Video OpenGL buffer.

```
void CVOpenGLBufferRelease (
   CVOpenGLBufferRef buffer
);
```

Parameters

buffer
The OpenGL buffer that you want to release.

Discussion
This function is equivalent to `CFRelease`, but `NULL` safe.

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

Declared In
`CVOpenGLBuffer.h`

CVOpenGLBufferRetain

Retains a Core Video OpenGL buffer.

```
CVOpenGLBufferRef CVOpenGLBufferRetain (
   CVOpenGLBufferRef buffer
);
```

Parameters

buffer

The OpenGL Buffer that you want to retain.

Return Value

For convenience, the OpenGL buffer that was retained.

Discussion

This function is equivalent to `CFRetain`, but `NULL` safe.

Availability

Available in Mac OS X v10.3 and later.

Declared In

`CVOpenGLBuffer.h`

CVOpenGLTextureCacheCreate

Creates an OpenGL texture cache.

```
CVReturn CVOpenGLTextureCacheCreate (
   CFAllocatorRef allocator,
   CFDictionaryRef cacheAttributes,
   CGLContextObj cglContext,
   CGLPixelFormatObj cglPixelFormat,
   CFDictionaryRef textureAttributes,
   CVOpenGLTextureCacheRef *cacheOut
);
```

Parameters

allocator

The allocator to use for allocating the cache. Pass `NULL` to specify the default allocator.

cacheAttributes

A Core Foundation dictionary containing the attributes of the cache itself. Pass `NULL` to specify no attributes.

cglContext

The OpenGL context into which the texture objects will be created.

cglPixelFormat

The OpenGL pixel format used to create the OpenGL context specified in `cglContext`.

textureAttributes

A Core Foundation dictionary containing the attributes to be used for creating the OpenGL texture objects. Pass `NULL` to specify no attributes.

cacheOut

On return, `cacheOut` points to the newly created texture cache.

Return Value

A Core Video result code. See "Result Codes" (page 88) for possible values.

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

Declared In
`CVOpenGLTextureCache.h`

CVOpenGLTextureCacheCreateTextureFromImage

Creates an OpenGL texture object from an existing image buffer.

```
CVReturn CVOpenGLTextureCacheCreateTextureFromImage (
   CFAllocatorRef allocator,
   CVOpenGLTextureCacheRef textureCache,
   CVImageBufferRef sourceImage,
   CFDictionaryRef attributes,
   CVOpenGLTextureRef *textureOut
);
```

Parameters

allocator
: The allocator to use for allocating the OpenGL texture object. May be `NULL` to specify the default allocator.

textureCache
: The OpenGL texture cache to be used to manage the texture.

sourceImage
: The image buffer from which you want to create an OpenGL texture.

attributes
: The desired buffer attributes for the OpenGL texture.

textureOut
: On return, `textureOut` points to the newly created texture object.

Return Value
A Core Video result code. See "Result Codes" (page 88) for possible values.

Discussion
This function copies all image buffer attachments designated as propagatable to the newly-created texture.

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

Declared In
`CVOpenGLTextureCache.h`

CVOpenGLTextureCacheFlush

Flushes the OpenGL texture cache.

```
void CVOpenGLTextureCacheFlush (
   CVOpenGLTextureCacheRef textureCache,
   CVOptionFlags options
);
```

Parameters

textureCache

The texture cache to flush.

options

Currently unused; pass `0`.

Return Value

A Core Video result code. See "Result Codes" (page 88) for possible values.

Discussion

You must call this function periodically to allow the texture cache to perform housekeeping operations.

Availability

Available in Mac OS X v10.3 and later.

Declared In

`CVOpenGLTextureCache.h`

CVOpenGLTextureCacheGetTypeID

Returns the Core Foundation ID of the texture cache type.

```
CFTypeID CVOpenGLTextureCacheGetTypeID (
   void
);
```

Return Value

The Core Foundation ID for this type.

Availability

Available in Mac OS X v10.3 and later.

Declared In

`CVOpenGLTextureCache.h`

CVOpenGLTextureCacheRelease

Releases an OpenGL texture cache.

```
void CVOpenGLTextureCacheRelease (
   CVOpenGLTextureCacheRef textureCache
);
```

Parameters

textureCache

The OpenGL texture cache that you want to release.

Discussion

This function is equivalent to `CFRelease`, but `NULL` safe.

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

Declared In
`CVOpenGLTextureCache.h`

CVOpenGLTextureCacheRetain

Retains an OpenGL texture cache.

```
CVOpenGLTextureCacheRef CVOpenGLTextureCacheRetain (
   CVOpenGLTextureCacheRef textureCache
);
```

Parameters

textureCache

The OpenGL texture cache that you want to retain.

Return Value
For convenience, the return value is the buffer you wanted to retain.

Discussion
This function is equivalent to `CFRetain`, but `NULL` safe.

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

Declared In
`CVOpenGLTextureCache.h`

CVOpenGLTextureGetCleanTexCoords

Returns the texture coordinates for the part of the image that should be displayed.

```
void CVOpenGLTextureGetCleanTexCoords (
   CVOpenGLTextureRef image,
   GLfloat lowerLeft[2],
   GLfloat lowerRight[2],
   GLfloat upperRight[2],
   GLfloat upperLeft[2]
);
```

Parameters

image

The Core Video OpenGL texture whose clean tex coordinates you want to obtain.

lowerLeft

On return, the `GLFloat` array hold the *s* and *t* texture coordinates of the lower-left corner of the image.

lowerRight

On return, the `GLFloat` array hold the *s* and *t* texture coordinates of the lower-right corner of the image.

upperRight

On return, the `GLFloat` array hold the *s* and *t* texture coordinates of the upper-right corner of the image.

upperLeft

On return, the `GLFloat` array hold the *s* and *t* texture coordinates of the upper-left corner of the image.

Discussion
This function automatically takes into account whether or not the texture is flipped.

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

Related Sample Code
LiveVideoMixer3
QTCoreVideo102
QTCoreVideo201
QTCoreVideo301
QTQuartzPlayer

Declared In
`CVOpenGLTexture.h`

CVOpenGLTextureGetName

Returns the texture target name of a CoreVideo OpenGL texture.

```
GLuint CVOpenGLTextureGetName (
   CVOpenGLTextureRef image
);
```

Parameters

image

The Core Video OpenGL texture whose texture target name you want to obtain.

Return Value
The target name of the texture.

Discussion
See the OpenGL specification for more information about texture targets.

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

Related Sample Code
LiveVideoMixer2
LiveVideoMixer3
QTCoreVideo102
QTCoreVideo201
QTCoreVideo301

Declared In
`CVOpenGLTexture.h`

CVOpenGLTextureGetTarget

Returns the texture target (for example, `GL_TEXTURE_2D`) of an OpenGL texture.

```
GLenum CVOpenGLTextureGetTarget (
   CVOpenGLTextureRef image
);
```

Parameters

image

The Core Video OpenGL texture whose target you want to obtain.

Return Value

The OpenGL texture target.

Discussion

See the OpenGL specification for more information about texture targets.

Availability

Available in Mac OS X v10.3 and later.

Related Sample Code

LiveVideoMixer2

LiveVideoMixer3

QTCoreVideo102

QTCoreVideo201

QTCoreVideo301

Declared In

`CVOpenGLTexture.h`

CVOpenGLTextureGetTypeID

Obtains the Core Foundation ID for the Core Video OpenGL texture type.

```
CFTypeID CVOpenGLTextureGetTypeID (
   void
);
```

Return Value

The Core Foundation ID for this type.

Availability

Available in Mac OS X v10.3 and later.

Related Sample Code

QTQuartzPlayer

Declared In

`CVOpenGLTexture.h`

CVOpenGLTextureIsFlipped

Determines whether or not an OpenGL texture is flipped vertically.

```
Boolean CVOpenGLTextureIsFlipped (
   CVOpenGLTextureRef image
);
```

Parameters

image

The Core Video OpenGL texture whose orientation you want to determine.

Return Value
Returns `true` if (0,0) in the texture is in the upper-left corner, `false` if (0,0) is in the lower left corner.

Discussion
Quartz assumes a lower-left origin.

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

Declared In
`CVOpenGLTexture.h`

CVOpenGLTextureRelease

Releases a Core Video OpenGL texture.

```
void CVOpenGLTextureRelease (
   CVOpenGLTextureRef texture
);
```

Parameters

texture

The Core Video OpenGL texture that you want to release.

Discussion
This function is equivalent to `CFRelease`, but `NULL` safe.

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

Related Sample Code
CIVideoDemoGL
LiveVideoMixer3
QTCoreImage101
QTCoreVideo102
QTCoreVideo201

Declared In
`CVOpenGLTexture.h`

CVOpenGLTextureRetain

Retains a Core Video OpenGL texture.

```
CVOpenGLTextureRef CVOpenGLTextureRetain (
   CVOpenGLTextureRef texture
);
```

Parameters

texture

The Core Video OpenGL texture that you want to retain.

Return Value

For convenience, the Core Video OpenGL texture you want to retain.

Discussion

This function is equivalent to `CFRetain`, but `NULL` safe.

Availability

Available in Mac OS X v10.3 and later.

Declared In

`CVOpenGLTexture.h`

CVPixelBufferCreate

Creates a single pixel buffer for a given size and pixel format.

```
CVReturn CVPixelBufferCreate (
   CFAllocatorRef allocator,
   size_t width,
   size_t height,
   OSType pixelFormatType,
   CFDictionaryRef pixelBufferAttributes,
   CVPixelBufferRef *pixelBufferOut
);
```

Parameters

allocator

The allocator to use to create the pixel buffer. Pass `NULL` to specify the default allocator.

width

Width of the pixel buffer, in pixels.

height

Height of the pixel buffer, in pixels.

pixelFormatType

The pixel format identified by its respective four-character code (type `OSType`).

pixelBufferAttributes

A dictionary with additonal attributes for a pixel buffer. This parameter is optional. See "Pixel Buffer Attribute Keys" (page 81) for more details.

pixelBufferOut

On return, `pixelBufferOut` points to the newly created pixel buffer.

Return Value

A Core Video result code. See "Result Codes" (page 88) for possible values.

Discussion
This function allocates the necessary memory based on the pixel dimensions, format, and extended pixels described in the pixel buffer's attributes.

Some of the parameters specified in this call override equivalent pixel buffer attributes. For example, if you define the `kCVPixelBufferWidth` and `kCVPixelBufferHeight` keys in the pixel buffer attributes parameter (`pixelBufferAttributes`), these values are overriden by the `width` and `height` parameters.

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

Declared In
`CVPixelBuffer.h`

CVPixelBufferCreateResolvedAttributesDictionary

Takes an array of `CFDictionary` objects describing various pixel buffer attributes and tries to resolve them into a single dictionary.

```
CVReturn CVPixelBufferCreateResolvedAttributesDictionary (
   CFAllocatorRef allocator,
   CFArrayRef attributes,
   CFDictionaryRef *resolvedDictionaryOut
);
```

Parameters

`allocator`
The allocator to use to create the pixel buffer. Pass `NULL` to specify the default allocator.

`attributes`
An array of Core Foundation dictionaries containing pixel buffer attribute key-value pairs.

`resolvedDictionaryOut`
On return, `resolvedDictionaryOut` points to the consolidated dictionary.

Return Value
A Core Video result code. See "Result Codes" (page 88) for possible values.

Discussion
This call is useful when you need to resolve requirements between several potential clients of a buffer.

If two or more dictionaries contain the same key but different values, errors may occur. For example, the width and height attributes must match, but if the bytes-per-row (rowBytes) attributes differ, the least common multiple is taken. Mismatches in pixel format allocators or callbacks also cause an error.

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

Declared In
`CVPixelBuffer.h`

CVPixelBufferCreateWithBytes

Creates a pixel buffer for a given size and pixel format containing data specified by a memory location.

```
CVReturn CVPixelBufferCreateWithBytes (
   CFAllocatorRef allocator,
   size_t width,
   size_t height,
   OSType pixelFormatType,
   void *baseAddress,
   size_t bytesPerRow,
   CVPixelBufferReleaseBytesCallback releaseCallback,
   void *releaseRefCon,
   CFDictionaryRef pixelBufferAttributes,
   CVPixelBufferRef *pixelBufferOut
);
```

Parameters

allocator

The allocator to use to create this buffer. Pass `NULL` to specify the default allocator.

width

Width of the pixel buffer, in pixels.

height

Height of the pixel buffer, in pixels.

pixelFormatType

Pixel format indentified by its respective four character code (type `OSType`).

baseAddress

A pointer to the base address of the memory storing the pixels.

bytesPerRow

Row bytes of the pixel storage memory.

releaseCallback

The callback function to be called when the pixel buffer is destroyed. This callback allows the owner of the pixels to free the memory. See `CVPixelBufferReleaseBytesCallback` (page 65) for more information.

releaseRefCon

User data identifying the pixel buffer. This value is passed to your pixel buffer release callback.

pixelBufferAttributes

A Core Foundation dictionary with additonal attributes for a a pixel buffer. This parameter is optional. See "Pixel Buffer Attribute Keys" (page 81) for more details.

pixelBufferOut

On return, `pixelBufferOut` points to the newly created pixel buffer.

Return Value

A Core Video result code. See "Result Codes" (page 88) for possible values.

Discussion

Some of the parameters specified in this call override equivalent pixel buffer attributes. For example, if you define the `kCVPixelBufferWidth` and `kCVPixelBufferHeight` keys in the pixel buffer attributes parameter (`pixelBufferAttributes`), these values are overriden by the `width` and `height` parameters.

Availability

Available in Mac OS X v10.3 and later.

Declared In

`CVPixelBuffer.h`

CVPixelBufferCreateWithPlanarBytes

Creates a single pixel buffer in planar format for a given size and pixel format containing data specified by a memory location.

```
CVReturn CVPixelBufferCreateWithPlanarBytes (
   CFAllocatorRef allocator,
   size_t width,
   size_t height,
   OSType pixelFormatType,
   void *dataPtr,
   size_t dataSize,
   size_t numberOfPlanes,
   void *planeBaseAddress[],
   size_t planeWidth[],
   size_t planeHeight[],
   size_t planeBytesPerRow[],
   CVPixelBufferReleasePlanarBytesCallback releaseCallback,
   void *releaseRefCon,
   CFDictionaryRef pixelBufferAttributes,
   CVPixelBufferRef *pixelBufferOut
);
```

Parameters

allocator
: The allocator to use to create this buffer. Pass `NULL` to specify the default allocator.

width
: Width of the pixel buffer, in pixels.

height
: Height of the pixel buffer, in pixels.

pixelFormatType
: Pixel format indentified by its respective four-character code (type `OSType`).

dataPtr
: A pointer to a plane descriptor block if applicable, or `NULL` if not.

dataSize
: The size of the memory if the planes are contiguous, or `NULL` if not.

numberOfPlanes
: The number of planes.

planeBaseAddress
: The array of base addresses for the planes.

planeWidth
: The array of plane widths.

planeHeight
: The array of plane heights.

planeBytesPerRow
: Thje array of plane bytes-per-row values.

releaseCallback
: The callback function that gets called when the pixel buffer is destroyed. This callback allows the owner of the pixels to free the memory. See `CVPixelBufferReleaseBytesCallback` (page 65) for more information.

releaseRefCon

A pointer to user data identifying the pixel buffer. This value is passed to your pixel buffer release callback.

pixelBufferAttributes

A dictionary with additonal attributes for a a pixel buffer. This parameter is optional. See "Pixel Buffer Attribute Keys" (page 81) for more details.

pixelBufferOut

On return, `pixelBufferOut` points to the newly created pixel buffer.

Return Value
A Core Video result code. See "Result Codes" (page 88) for possible values.

Discussion
Some of the parameters specified in this call override equivalent pixel buffer attributes. For example, if you define the `kCVPixelBufferWidth` and `kCVPixelBufferHeight` keys in the pixel buffer attributes parameter (`pixelBufferAttributes`), these values are overriden by the `width` and `height` parameters.

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

Declared In
`CVPixelBuffer.h`

CVPixelBufferFillExtendedPixels

Fills the extended pixels of the pixel buffer.

```
CVReturn CVPixelBufferFillExtendedPixels (
   CVPixelBufferRef pixelBuffer
);
```

Parameters

pixelBuffer

The pixel buffer whose extended pixels you want to fill.

Discussion
This function replicates edge pixels to fill the entire extended region of the image.

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

Declared In
`CVPixelBuffer.h`

CVPixelBufferGetBaseAddress

Returns the base address of the pixel buffer.

```
void * CVPixelBufferGetBaseAddress (
   CVPixelBufferRef pixelBuffer
);
```

Parameters

pixelBuffer

The pixel buffer whose base address you want to obtain.

Return Value

The base address of the pixels. For chunky buffers, this returns a pointer to the pixel at (0,0) in the buffer For planar buffers this returns a pointer to a `PlanarComponentInfo` structure (as defined by QuickTime in `ImageCodec.h`).

Discussion

Retrieving the base address for a pixel buffer requires that the buffer base address be locked via a successful call to `CVPixelBufferLockBaseAddress` (page 56).

Availability

Available in Mac OS X v10.3 and later.

Related Sample Code

CaptureAndCompressIPBMovie

MovieVideoChart

OpenGLCaptureToMovie

QTPixelBufferVCToCGImage

Quartz Composer QCTV

Declared In

`CVPixelBuffer.h`

CVPixelBufferGetBaseAddressOfPlane

Returns the base address of the plane at the specified plane index.

```
void * CVPixelBufferGetBaseAddressOfPlane (
   CVPixelBufferRef pixelBuffer,
   size_t planeIndex
);
```

Parameters

pixelBuffer

The pixel buffer containing the plane whose base address you want to obtain.

planeIndex

The index of the plane.

Return Value

The base address of the plane, or `NULL` for nonplanar pixel buffers.

Discussion

Retrieving the base address for a pixel buffer requires that the buffer base address be locked by a successful call to `CVPixelBufferLockBaseAddress` (page 56).

Availability

Available in Mac OS X v10.3 and later.

Declared In
`CVPixelBuffer.h`

CVPixelBufferGetBytesPerRow

Returns the number of bytes per row of the pixel buffer.

```
size_t CVPixelBufferGetBytesPerRow (
   CVPixelBufferRef pixelBuffer
);
```

Parameters

pixelBuffer

The pixel buffer whose bytes-per-row value you want to obtain.

Return Value
The number of bytes per row of the image data. For planar buffers this function returns a `rowBytes` value such that `bytesPerRow` * `height` covers the entire image including all planes.

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

Related Sample Code
CaptureAndCompressIPBMovie
MovieVideoChart
OpenGLCaptureToMovie
QTPixelBufferVCToCGImage
Quartz Composer QCTV

Declared In
`CVPixelBuffer.h`

CVPixelBufferGetBytesPerRowOfPlane

Returns the number of bytes per row for a plane at the specified index in the pixel buffer.

```
size_t CVPixelBufferGetBytesPerRowOfPlane (
   CVPixelBufferRef pixelBuffer,
   size_t planeIndex
);
```

Parameters

pixelBuffer

The pixel buffer containing the plane.

planeIndex

The index of the plane whose bytes-per-row value you want to obtain.

Return Value
The number of row bytes of the plane, or `NULL` for nonplanar pixel buffers.

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

Declared In
CVPixelBuffer.h

CVPixelBufferGetDataSize

Returns the data size for contiguous planes of the pixel buffer.

```
size_t CVPixelBufferGetDataSize (
   CVPixelBufferRef pixelBuffer
);
```

Parameters

pixelBuffer
The pixel buffer whose data size you want to obtain.

Return Value
The data size as specified in the call to CVPixelBufferCreateWithPlanarBytes (page 48).

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

Declared In
CVPixelBuffer.h

CVPixelBufferGetExtendedPixels

Returns the amount of extended pixel padding in the pixel buffer.

```
void CVPixelBufferGetExtendedPixels (
   CVPixelBufferRef pixelBuffer,
   size_t *extraColumnsOnLeft,
   size_t *extraColumnsOnRight,
   size_t *extraRowsOnTop,
   size_t *extraRowsOnBottom
);
```

Parameters

pixelBuffer
The pixel buffer whose extended pixel size you want to obtain.

extraColumnsOnLeft
Returns the pixel row padding to the left. May be NULL.

extraColumnsOnRight
Returns the pixel row padding to the right. May be NULL.

extraRowsOnTop
Returns the pixel row padding to the top. May be NULL.

extraRowsOnBottom
The pixel row padding to the bottom. May be NULL.

Discussion

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

Declared In
`CVPixelBuffer.h`

CVPixelBufferGetHeight

Returns the height of the pixel buffer.

```
size_t CVPixelBufferGetHeight (
   CVPixelBufferRef pixelBuffer
);
```

Parameters

pixelBuffer
The pixel buffer whose height you want to obtain.

Return Value
The buffer height, in pixels.

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

Related Sample Code
CaptureAndCompressIPBMovie
MovieVideoChart
QTPixelBufferVCToCGImage

Declared In
`CVPixelBuffer.h`

CVPixelBufferGetHeightOfPlane

Returns the height of the plane at planeIndex in the pixel buffer.

```
size_t CVPixelBufferGetHeightOfPlane (
   CVPixelBufferRef pixelBuffer,
   size_t planeIndex
);
```

Parameters

pixelBuffer
The pixel buffer whose plane height you want to obtain.

planeIndex
The index of the plane.

Return Value
The height of the buffer, in pixels, or `0` for nonplanar pixel buffers.

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

Declared In
`CVPixelBuffer.h`

CVPixelBufferGetPixelFormatType

Returns the pixel format type of the pixel buffer.

```
OSType CVPixelBufferGetPixelFormatType (
   CVPixelBufferRef pixelBuffer
);
```

Parameters

pixelBuffer

The pixel buffer whose format type you want to obtain.

Return Value

A four-character code `OSType` identifier for the pixel format.

Discussion

Availability

Available in Mac OS X v10.3 and later.

Related Sample Code

QTPixelBufferVCToCGImage

Declared In

`CVPixelBuffer.h`

CVPixelBufferGetPlaneCount

Returns number of planes of the pixel buffer.

```
size_t CVPixelBufferGetPlaneCount (
   CVPixelBufferRef pixelBuffer
);
```

Parameters

pixelBuffer

The pixel buffer whose plane count you want to obtain..

Return Value

The number of planes. Returns `0` for nonplanar pixel buffers.

Availability

Available in Mac OS X v10.3 and later.

Declared In

`CVPixelBuffer.h`

CVPixelBufferGetTypeID

Returns the Core Foundation ID of the pixel buffer type.

```
CFTypeID CVPixelBufferGetTypeID (
   void
);
```

Return Value
The Core Foundation ID for this type.

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

Declared In
`CVPixelBuffer.h`

CVPixelBufferGetWidth

Returns the width of the pixel buffer.

```
size_t CVPixelBufferGetWidth (
   CVPixelBufferRef pixelBuffer
);
```

Parameters

pixelBuffer
The pixel buffer whose width you want to obtain.

Return Value
The width of the buffer, in pixels.

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

Related Sample Code
CaptureAndCompressIPBMovie
MovieVideoChart
QTPixelBufferVCToCGImage

Declared In
`CVPixelBuffer.h`

CVPixelBufferGetWidthOfPlane

Returns the width of the plane at a given index in the pixel buffer.

```
size_t CVPixelBufferGetWidthOfPlane (
   CVPixelBufferRef pixelBuffer,
   size_t planeIndex
);
```

Parameters

pixelBuffer
The pixel buffer whose plane width you want to obtain.

planeIndex
The index of the plane at which to obtain the width.

Return Value
The width of the plane, in pixels, or `0` for nonplanar pixel buffers.

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

Declared In
`CVPixelBuffer.h`

CVPixelBufferIsPlanar

Determine if the pixel buffer is planar.

```
Boolean CVPixelBufferIsPlanar (
   CVPixelBufferRef pixelBuffer
);
```

Parameters

`pixelBuffer`
The pixel buffer to check.

Return Value
Returns `true` if the pixel buffer was created using `CVPixelBufferCreateWithPlanarBytes` (page 48), `false` otherwise.

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

Declared In
`CVPixelBuffer.h`

CVPixelBufferLockBaseAddress

Locks the base address of the pixel buffer.

```
CVReturn CVPixelBufferLockBaseAddress (
   CVPixelBufferRef pixelBuffer,
   CVOptionFlags lockFlags
);
```

Parameters

`pixelBuffer`
The pixel buffer whose base address you want to lock.

`lockFlags`
No options currently defined; pass `0`.

Return Value
A Core Video result code. See "Result Codes" (page 88) for possible values.

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

Related Sample Code
CaptureAndCompressIPBMovie

MovieVideoChart
OpenGLCaptureToMovie
QTPixelBufferVCToCGImage
Quartz Composer QCTV

Declared In
`CVPixelBuffer.h`

CVPixelBufferPoolCreate

Creates a pixel buffer pool.

```
CVReturn CVPixelBufferPoolCreate (
   CFAllocatorRef allocator,
   CFDictionaryRef poolAttributes,
   CFDictionaryRef pixelBufferAttributes,
   CVPixelBufferPoolRef *poolOut
);
```

Parameters

allocator
The allocator to use for allocating this buffer pool. Pass `NULL` to specify the default allocator.

poolAttributes
A Core Foundation dictionary containing the attributes for this pixel buffer pool.

pixelBufferAttributes
A Core Foundation dictionary containing the attributes to be used for creating new pixel buffers within the pool.

poolOut
On return, `poolOut` points to the newly created pixel buffer pool.

Return Value
A Core Video result code. See "Result Codes" (page 88) for possible values.

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

Related Sample Code
OpenGLCaptureToMovie
Quartz Composer QCTV

Declared In
`CVPixelBufferPool.h`

CVPixelBufferPoolCreatePixelBuffer

Creates a pixel buffer from a pixel buffer pool.

```
CVReturn CVPixelBufferPoolCreatePixelBuffer (
   CFAllocatorRef allocator,
   CVPixelBufferPoolRef pixelBufferPool,
   CVPixelBufferRef *pixelBufferOut
);
```

Parameters

allocator

The allocator to use for creating the pixel buffer. Pass `NULL` to specify the default allocator.

pixelBufferPool

The pixel buffer pool for creating the new pixel buffer.

pixelBufferOut

On return, `pixelBufferOut` points to the newly created pixel buffer.

Return Value
A Core Video result code. See "Result Codes" (page 88) for possible values.

Discussion
This function creates a new pixel buffer using the pixel buffer attributes specifed during pool creation. This buffer has default attachments as specified in the `pixelBufferAttributes` parameter of `CVPixelBufferPoolCreate` (page 57) (using either the `kCVBufferPropagatedAttachmentsKey` or `kCVBufferNonPropagatedAttachmentsKey` attributes).

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

Related Sample Code
OpenGLCaptureToMovie
Quartz Composer QCTV

Declared In
`CVPixelBufferPool.h`

CVPixelBufferPoolGetAttributes

Returns the pool attributes dictionary for a pixel buffer pool.

```
CFDictionaryRef CVPixelBufferPoolGetAttributes (
   CVPixelBufferPoolRef pool
);
```

Parameters

pool

The pixel buffer pool to retrieve the attributes from.

Return Value
A Core Foundation dictionary containing the pool attributes, or `NULL` on failure.

Discussion

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

Declared In
`CVPixelBufferPool.h`

CVPixelBufferPoolGetPixelBufferAttributes

Returns the attributes of pixel buffers that will be created from this pool.

```
CFDictionaryRef CVPixelBufferPoolGetPixelBufferAttributes (
   CVPixelBufferPoolRef pool
);
```

Parameters

pool

The pixel buffer pool to retrieve the attributes from.

Return Value

A Core Foundation dictionary containing the pixel buffer attributes, or `NULL` on failure.

Discussion

This function is provided for those cases where you may need to know some information about the buffers that will be created for you .

Availability

Available in Mac OS X v10.3 and later.

Declared In

`CVPixelBufferPool.h`

CVPixelBufferPoolGetTypeID

Returns the Core Foundation ID of the pixel buffer pool type.

```
CFTypeID CVPixelBufferPoolGetTypeID (
   void
);
```

Return Value

The Core Foundation ID for this type.

Availability

Available in Mac OS X v10.3 and later.

Declared In

`CVPixelBufferPool.h`

CVPixelBufferPoolRelease

Releases a pixel buffer pool.

```
void CVPixelBufferPoolRelease (
   CVPixelBufferPoolRef pixelBufferPool
);
```

Parameters

pixelBufferPool

The pixel buffer pool that you want to release.

Discussion
This function is equivalent to `CFRelease`, but `NULL` safe.

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

Related Sample Code
OpenGLCaptureToMovie
Quartz Composer QCTV

Declared In
`CVPixelBufferPool.h`

CVPixelBufferPoolRetain

Retains a pixel buffer pool.

```
CVPixelBufferPoolRef CVPixelBufferPoolRetain (
   CVPixelBufferPoolRef pixelBufferPool
);
```

Parameters

buffer

The pixel buffer pool that you want to retain.

Return Value
For convenience, the same pixel buffer pool that you wanted to retain.

Discussion
This function is equivalent to `CFRetain`, but `NULL` safe.

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

Declared In
`CVPixelBufferPool.h`

CVPixelBufferRelease

Releases a pixel buffer.

```
void CVPixelBufferRelease (
   CVPixelBufferRef texture
);
```

Parameters

buffer

The pixel buffer that you want to release.

Discussion
This function is equivalent to `CFRelease`, but `NULL` safe.

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

Related Sample Code
OpenGLCaptureToMovie
QTCoreVideo103
QTCoreVideo202
QTPixelBufferVCToCGImage

Declared In
`CVPixelBuffer.h`

CVPixelBufferRetain

Retains a pixel buffer.

```
CVPixelBufferRef CVPixelBufferRetain (
   CVPixelBufferRef texture
);
```

Parameters

buffer

The pixel buffer that you want to retain.

Return Value
For convenience, the same pixel buffer you want to retain.

Discussion
This function is equivalent to `CFRetain`, but `NULL` safe.

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

Declared In
`CVPixelBuffer.h`

CVPixelBufferUnlockBaseAddress

Unlocks the base address of the pixel buffer.

```
CVReturn CVPixelBufferUnlockBaseAddress (
   CVPixelBufferRef pixelBuffer,
   CVOptionFlags unlockFlags
);
```

Parameters

pixelBuffer

The pixel buffer whose base address you want to unlock.

unlockFlags

No options currently defined; pass `0`.

Return Value
A Core Video result code. See "Result Codes" (page 88) for possible values.

Discussion

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

Related Sample Code
CaptureAndCompressIPBMovie
MovieVideoChart
QTCoreVideo202
QTPixelBufferVCToCGImage
Quartz Composer QCTV

Declared In
`CVPixelBuffer.h`

CVPixelFormatDescriptionArrayCreateWithAllPixelFormatTypes

Returns all the pixel format descriptions known to Core Video.

```
CFArrayRef CVPixelFormatDescriptionArrayCreateWithAllPixelFormatTypes (
   CFAllocatorRef allocator
);
```

Parameters

allocator
The allocator to use when creating the description. Pass `NULL` to specify the default allocator.

Return Value
An array of Core Foundation dictionaries, each containing a pixel format description. See "Pixel Format Description Keys" (page 83) for a list of keys relevant to the format description.

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

Declared In
`CVPixelFormatDescription.h`

CVPixelFormatDescriptionCreateWithPixelFormatType

Creates a pixel format description from a given `OSType` identifier.

```
CFDictionaryRef CVPixelFormatDescriptionCreateWithPixelFormatType (
   CFAllocatorRef allocator,
   OSType pixelFormat
);
```

Parameters

allocator
The allocator to use when creating the description. Pass `NULL` to specify the default allocator.

pixelFormat
A four-character code that identifies the pixel format you want to obtain.

Return Value
A Core Foundation dictionary containing the pixel format description. See "Pixel Format Description Keys" (page 83) for a list of keys relevant to the format description.

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

Declared In
`CVPixelFormatDescription.h`

CVPixelFormatDescriptionRegisterDescriptionWithPixelFormatType

Registers a pixel format description with Core Video.

```
void CVPixelFormatDescriptionRegisterDescriptionWithPixelFormatType (
   CFDictionaryRef description,
   OSType pixelFormat
);
```

Parameters

description

A Core Foundation dictionary containing the pixel format description. See "Pixel Format Description Keys" (page 83) for a list of required and optional keys.

pixelFormat

The four-character code (type `OSType`) identifier for this pixel format.

Discussion
If you are using a custom pixel format, you must register the format with Core Video using this function. See Technical Q&A 1401: Registering Custom Pixel Formats with QuickTime and Core Video for more details.

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

Declared In
`CVPixelFormatDescription.h`

Callbacks

CVDisplayLinkOutputCallback

Defines a pointer to a display link output callback function, which is called whenever the display link wants the application to output a frame.

```
typedef CVReturn (*CVDisplayLinkOutputCallback)(
   CVDisplayLinkRef displayLink,
   const CVTimeStamp *inNow,
   const CVTimeStamp *inOutputTime,
   CVOptionFlags flagsIn,
   CVOptionFlags *flagsOut,
   void *displayLinkContext
   );
```

You would declare a display link output callback function named `MyDisplayLinkCallback` like this:

```
CVReturn MyDisplayLinkCallback (
   CVDisplayLinkRef displayLink,
   const CVTimeStamp *inNow,
   const CVTimeStamp *inOutputTime,
   CVOptionFlags flagsIn,
   CVOptionFlags *flagsOut,
   void *displayLinkContext
   );
```

Parameters

displayLink
: The display link requesting the frame.

inNow
: A pointer to the current time.

inOutputTime
: A pointer to the time that the frame will be displayed.

flagsIn
: Currently unused. Pass 0.

flagsOut
: Currently unused. Pass 0.

displayLinkContext
: A pointer to application-defined data. This is the pointer you passed into the `CVDisplayLinkSetOutputCallback` (page 26) function when registering your callback.

Discussion

For a given display link, you must register a display link output callback using `CVDisplayLinkSetOutputCallback` (page 26) so that you can process and output the requested frame.

You callback must retrieve the frame with the timestamp specified by the (`inOutputTime` parameter, manipulate it if desired (for example, apply color correction or map into onto a surface), and then output it to the display.

Availability

Available in Mac OS X v10.3 and later.

Declared In

`CVDisplayLink.h`

CVFillExtendedPixelsCallBack

Defines a pointer to a custom extended pixel-fill function, which is called whenever the system needs to pad a buffer holding your custom pixel format.

```
typedef Boolean (*CVFillExtendedPixelsCallBack)(
   CVPixelBufferRef pixelBuffer,
   void *refCon
   );
```

Here is how you would declare a custom fill function named `MyExtendedPixelFillFunc`

```
Boolean MyExtendedPixelFillFunc (
   CVPixelBufferRef pixelBuffer,
   void *refCon
   );
```

Parameters

pixelBuffer

The pixel buffer to be padded.

refCon

A pointer to application-defined data. This is the same value you stored in the `CVFillExtendedPixelsCallbackData` (page 67) structure.

Return Value

Return `true` if the padding was successful, `false` otherwise.

Discussion

For more information on implementing a custom extended pixel-fill callback, see Technical Q&A 1440: Implementing a CVFillExtendedPixelsCallback.

Availability

Available in Mac OS X v10.3 and later.

Declared In

`CVPixelFormatDescription.h`

CVPixelBufferReleaseBytesCallback

Defines a pointer to a pixel buffer release callback function, which is called when a pixel buffer created by `CVPixelBufferCreateWithBytes` (page 46) is released.

```
typedef void (*CVPixelBufferReleaseBytesCallback)(
   void *releaseRefCon,
   const void *baseAddress
   );
```

You would declare a pixel buffer release callback named `MyPixelBufferReleaseCallback` like this:

```
void MyPixelBufferReleaseCallback(
   void *releaseRefCon,
   const void *baseAddress
   );
```

Parameters

releaseRefCon

A pointer to application-defined data. This pointer is the same as that passed in the `releaseRefCon` parameter of `CVPixelBufferCreateWithBytes` (page 46).

baseAddress

A pointer to the base address of the memory holding the pixels. This pointer is the same as that passed in the `baseAddress` parameter of `CVPixelBufferCreateWithBytes` (page 46).

Discussion

You use this callback to release the pixels and perform any other cleanup when a pixel buffer is released.

Availability

Available in Mac OS X v10.3 and later.

Declared In

`CVPixelBuffer.h`

CVPixelBufferReleasePlanarBytesCallback

Defines a pointer to a pixel buffer release callback function, which is called when a pixel buffer created by `CVPixelBufferCreateWithPlanarBytes` (page 48) is released.

```
typedef void (*CVPixelBufferReleasePlanarBytesCallback)(
   void *releaseRefCon,
   const void *dataPtr,
   size_t dataSize,
   size_t numberOfPlanes,
   const void *planeAddresses[]
   );
```

You would declare a callback named `MyPixelBufferReleasePlanarBytes` like this:

```
void MyPixelBufferReleasePlanarBytes)(
   void *releaseRefCon,
   const void *dataPtr,
   size_t dataSize,
   size_t numberOfPlanes,
   const void *planeAddresses[]
   );
```

Parameters

releaseRefCon

A pointer to application-defined data. This pointer is the same as that passed in the `releaseRefCon` parameter of `CVPixelBufferCreateWithPlanarBytes` (page 48).

dataPtr

A pointer to a plane descriptor block. This is the same pointer you passed to `CVPixelBufferCreateWithPlanarBytes` (page 48) in the `dataPtr` parameter.

dataSize

The size value you passed to `CVPixelBufferCreateWithPlanarBytes` (page 48) in the `dataSize` parameter.

numberOfPlanes

The number of planes value you passed to `CVPixelBufferCreateWithPlanarBytes` (page 48) in the `numberOfPlanes` parameter.

planeAddresses

A pointer to the base plane address you passed to `CVPixelBufferCreateWithPlanarBytes` (page 48) in the `basePlaneAddress` parameter.

Discussion
You use this callback to release the pixels and perform any other cleanup when a pixel buffer is released.

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

Declared In
`CVPixelBuffer.h`

Data Types

CVBufferRef

Defines the base type for all Core Video buffers.

```
typedef struct __CVBuffer *CVBufferRef;
```

Discussion
CVBuffers represent an abstract type from which all Core Video buffers derive.

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

Declared In
`CVBuffer.h`

CVDisplayLinkRef

Defines a display link.

```
typedef struct __CVDisplayLink *CVDisplayLinkRef;
```

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

Declared In
`CVDisplayLink.h`

CVFillExtendedPixelsCallbackData

Holds information describing a custom extended pixel fill algorithm.

```
typedef struct {
    CFIndex version;
    CVFillExtendedPixelsCallBack    fillCallBack;
    void *refCon;
} CVFillExtendedPixelsCallBackData;
```

Fields

`version`

The version of this fill algorithm.

`fillCallback`

A pointer to a custom pixel fill function.

`refCon`

A pointer to application-defined data that is passed to your custom pixel fill function.

Discussion

You must fill out this structure and store it as part of your pixel format description Core Foundation dictionary (key: `kCVPixelFormatFillExtendedPixelsCallback`, type: `CFData`). However, if your custom pixel format never needs the functionality of `CVPixelBufferFillExtendedPixels` (page 49), you don't need to add this key or implement the associated callback.

For more information about defining a custom pixel format, see "Pixel Format Description Keys" (page 83).

Availability

Available in Mac OS X v10.3 and later.

Declared In

`CVPixelFormatDescription.h`

CVImageBufferRef

Defines a Core Video image buffer.

```
typedef CVBufferRef CVImageBufferRef;
```

Discussion

An image buffer is an abstract type representing Core Video buffers that hold images. In Core Video, pixel buffers, OpenGL buffers, and OpenGL textures all derive from the image buffer type.

Availability

Available in Mac OS X v10.3 and later.

Declared In

`CVImageBuffer.h`

CVOptionFlags

Define flags to be used for the display link output callback function.

```
typedef uint64_t CVOptionFlags;
```

Discussion

No flags are currently defined.

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

Declared In
`CVBase.h`

CVOpenGLBufferRef

Defines a Core Video OpenGL buffer.

```
typedef CVImageBufferRef CVOpenGLBufferRef;
```

Discussion
The Core Video OpenGL buffer (type `CVOpenGLBufferRef` is a wrapper around the standard OpenGL pbuffer.

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

Declared In
`CVOpenGLBuffer.h`

CVOpenGLBufferPoolRef

Defines an OpenGL buffer pool.

```
typedef struct _CVOpenGLBufferPool *CVOpenGLBufferPoolRef;
```

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

Declared In
`CVOpenGLBufferPool.h`

CVOpenGLTextureRef

Defines an OpenGL texture-based image buffer.

```
typedef CVImageBufferRef CVOpenGLTextureRef;
```

Discussion
The Core Video OpenGL texture (type `CVOpenGLTextureRef` is a wrapper around the standard OpenGL texture.

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

Declared In
`CVOpenGLTexture.h`

CVOpenGLTextureCacheRef

Defines a CoreVideo OpenGL texture cache.

```
typedef struct __CVOpenGLTextureCache *CVOpenGLTextureCacheRef;
```

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

Declared In
`CVOpenGLTextureCache.h`

CVPixelBufferRef

Defines a Core Video pixel buffer.

```
typedef CVImageBufferRef CVPixelBufferRef;
```

Discussion
The pixel buffer stores an image in main memory.

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

Declared In
`CVPixelBuffer.h`

CVPixelBufferPoolRef

Defines a pixel buffer pool.

```
typedef struct _CVPixelBufferPool *CVPixelBufferPoolRef;
```

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

Declared In
`CVPixelBufferPool.h`

CVReturn

Defines the return error code for Core Video functions.

```
typedef int32_t CVReturn;
```

Discussion
See "Result Codes" (page 88) for possible values.

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

Declared In
`CVReturn.h`

CVSMPTETime

A structure for holding a SMPTE time.

```
struct CVSMPTETime    {
    SInt16  subframes;
    SInt16  subframeDivisor;
    UInt32  counter;
    UInt32  type;
    UInt32  flags;
    SInt16  hours;
    SInt16  minutes;
    SInt16  seconds;
    SInt16  frames;
    ;}
typedef struct CVSMPTETime CVSMPTETime;
```

Fields

`subframes`
The number of subframes in the full message.

`subframeDivisor`
The number of subframes per frame (typically, 80).

`counter`
The total number of messages received.

`type`
The kind of SMPTE time type. See "SMPTE Time Types" (page 87) for a list of possible values.

`flags`
A set of flags that indicate the SMPTE state. See "SMPTE State Flags" (page 86) for possible values.

`hours`
The number of hours in the full message.

`minutes`
The number of minutes in the full message.

`seconds`
The number of seconds in the full message.

`frames`
The number of frames in the full message.

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

Declared In
`CVBase.h`

CVTime

A structure for reporting Core Video time values.

```
typedef struct {
    int64_t     timeValue;
    int64_t     timeScale;
    int32_t     flags;
} CVTime;
```

Fields

`timeValue`

The time value.

`timeScale`

The time scale for this value.

`flags`

Flags associated with the `CVTime` value. See "CVTime Constants" (page 75) for possible values. If `kCVTimeIsIndefinite` is set, you should not use any of the other fields in this structure.

Discussion

This structure is equivalent to the QuickTime `QTTime` structure.

Availability

Available in Mac OS X v10.3 and later.

Declared In

`CVBase.h`

CVTimeStamp

A structure for defining a display timestamp.

```
typedef struct {
    uint32_t    version;
    int32_t     videoTimeScale;
    int64_t     videoTime;
    uint64_t    hostTime;
    double      rateScalar;
    int64_t     videoRefreshPeriod;
    CVSMPTETime smpteTime;
    uint64_t    flags;
    uint64_t    reserved;
} CVTimeStamp;
```

Fields

`version`

The current `CVTimeStamp` structure is version 0. Some functions require you to specify a version when passing in a timestamp structure to be filled.

`videoTimeScale`

The scale (in units per second) of the `videoTimeScale` and `videoRefreshPeriod` fields.

`videoTime`

The start of a frame (or field for interlaced video).

`hostTime`

The host root time base time.

`rateScalar`

The current rate of the device as measured by the timestamps, divided by the nominal rate

`videoPeriod`
: The nominal update period of the current output device.

`smpteTime`
: The SMPTE time representation of the timestamp.

`flags`
: A bit field containing additional information about the timestamp. See "CVTimeStamp Flags" (page 75) for a list of possible values. .

`reserved`
: Reserved. Do not use.

Discussion
This structure is designed to be very similar to the audio time stamp defined in the Core Audio framework. However, unlike the audio timestamps, floating-point values are not used to represent the video equivalent of sample times. This was done partly to avoid precision issues, and partly because QuickTime still uses integers for time values and time scales. In the actual implementation it has turned out to be very convenient to use integers, and we can represent frame rates like NTSC (30000/1001 fps) exactly. The `mHostTime` structure field uses the same Mach absolute time base used in Core Audio, so that clients of the Core Video API can synchronize between the two subsystems.

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

Declared In
`CVBase.h`

Constants

CVBuffer Attachment Keys

Specify an attachment type for a Core Video buffer.

```
const CFStringRef kCVBufferMovieTimeKey;
const CFStringRef kCVBufferTimeValueKey;
const CFStringRef kCVBufferTimeScaleKey;
```

Constants
`kCVBufferMovieTimeKey`
: The movie time associated with the buffer. Generally only available for frames emitted by QuickTime (type `CFDictionary` containing the `kCVBufferTimeValueKey` and `kCVBufferTimeScaleKey` keys).

  Available in Mac OS X v10.3 and later.

  Declared in `CVBuffer.h`.

`kCVBufferTimeValueKey`
: The actual time value associated with the movie.

  Available in Mac OS X v10.3 and later.

  Declared in `CVBuffer.h`.

`kCVBufferTimeScaleKey`
The time scale associated with the movie.

Available in Mac OS X v10.3 and later.

Declared in `CVBuffer.h`.

CVBuffer Attachment Modes

Specify the propagation mode of a Core Video buffer attachment.

```
enum {
    kCVAttachmentMode_ShouldNotPropagate    = 0,
    kCVAttachmentMode_ShouldPropagate       = 1,
};
typedef uint32_t CVAttachmentMode;
```

Constants

`kCVAttachmentMode_ShouldNotPropagate`
Do not propagate this attachment.

Available in Mac OS X v10.3 and later.

Declared in `CVBuffer.h`.

`kCVAttachmentMode_ShouldPropagate`
Copy this attachment when using the `CVBufferPropagateAttachments` (page 14) function. For example, in most cases, you would want to propagate an attachment bearing a timestamp to each successive buffer.

Available in Mac OS X v10.3 and later.

Declared in `CVBuffer.h`.

Discussion

You set these attributes when adding attachments to a `CVBuffer` object.

CVBuffer Attribute Keys

Specify attributes associated with Core Video buffers.

```
const CFStringRef kCVBufferPropagatedAttachmentsKey;
const CFStringRef kCVBufferNonPropagatedAttachmentsKey;
```

Constants

`kCVBufferPropagatedAttachmentsKey`
Attachments that should be copied when using the `CVBufferPropagateAttachments` (page 14) function (type `CFDictionary`, containing a list of attachments as key-value pairs).

Available in Mac OS X v10.3 and later.

Declared in `CVBuffer.h`.

`kCVBufferNonPropagatedAttachmentsKey`
Attachments that should not be copied when using the `CVBufferPropagateAttachments` (page 14) function (type `CFDictionary`, containing a list of attachments as key-value pairs).

Available in Mac OS X v10.3 and later.

Declared in `CVBuffer.h`.

Discussion
These attributes let you set multiple attachments at the time of buffer creation, rather than having to call `CVBufferSetAttachment` (page 17) for each attachment.

CVTime Constants

Specify flags for the `CVTime` structure.

```
enum {
kCVTimeIsIndefinite = 1 << 0
};
```

Constants

`kCVTimeIsIndefinite`
The time value is unknown.

Available in Mac OS X v10.3 and later.

Declared in `CVBase.h`.

CVTime Values

Indicate specific `CVTime` values.

```
const CVTime kCVZeroTime;
const CVTime kCVIndefiniteTime;
```

Constants

`kCVZeroTime`
Zero time or duration. For example, `CVDisplayLinkGetOutputVideoLatency` (page 23) returns `kCVZeroTime` for zero video latency.

Available in Mac OS X v10.3 and later.

Declared in `CVBase.h`.

`kCVIndefiniteTime`
An unknown or indefinite time. For example, `CVDisplayLinkGetNominalOutputVideoRefreshPeriod` (page 22) returns `kCVIndefiniteTime` if the display link specified is not valid.

Available in Mac OS X v10.3 and later.

Declared in `CVBase.h`.

CVTimeStamp Flags

Specify flags for the `CVTimeStamp` structure.

```
enum
{
    kCVTimeStampVideoTimeValid              = (1L << 0),
    kCVTimeStampHostTimeValid               = (1L << 1),
    kCVTimeStampSMPTETimeValid              = (1L << 2),
    kCVTimeStampVideoRefreshPeriodValid     = (1L << 3),
    kCVTimeStampRateScalarValid             = (1L << 4),
    kCVTimeStampTopField                    = (1L << 16),
    kCVTimeStampBottomField                 = (1L << 17)
};
enum
{
    kCVTimeStampVideoHostTimeValid   =
            (kCVTimeStampVideoTimeValid | kCVTimeStampHostTimeValid),
    kCVTimeStampIsInterlaced            =
            (kCVTimeStampTopField | kCVTimeStampBottomField)
};
```

Constants

`kCVTimeStampVideoTimeValid`

The value in the video time field is valid.

Available in Mac OS X v10.3 and later.

Declared in `CVBase.h`.

`kCVTimeStampHostTimeValid`

The value in the host time field is valid.

Available in Mac OS X v10.3 and later.

Declared in `CVBase.h`.

`kCVTimeStampSMPTETimeValid`

The value in the SMPTE time field is valid.

Available in Mac OS X v10.3 and later.

Declared in `CVBase.h`.

`kCVTimeStampVideoRefreshPeriodValid`

The value in the video refresh period field is valid.

Available in Mac OS X v10.3 and later.

Declared in `CVBase.h`.

`kCVTimeStampRateScalarValid`

The value in the rate scalar field is valid.

Available in Mac OS X v10.3 and later.

Declared in `CVBase.h`.

`kCVTimeStampTopField`

The timestamp represents the top lines of an interlaced image.

Available in Mac OS X v10.3 and later.

Declared in `CVBase.h`.

`kCVTimeStampBottomField`

The timestamp represents the bottom lines of an interlaced image.

Available in Mac OS X v10.3 and later.

Declared in `CVBase.h`.

`kCVTimeStampVideoHostTimeValid`
A convenience constant indicating that both the video time and host time fields are valid.

Available in Mac OS X v10.3 and later.

Declared in `CVBase.h`.

`kCVTimeStampIsInterlaced`
A convenience constant indicating that the timestamp is for an interlaced image.

Available in Mac OS X v10.3 and later.

Declared in `CVBase.h`.

Discussion
These flags indicate which fields in the `CVTimeStamp` (page 72) structure contain valid information.

Image Buffer Attachment Keys

Specify attachment types associated with image buffers.

```
const CFStringRef   kCVImageBufferCGColorSpaceKey;
const CFStringRef   kCVImageBufferGammaLevelKey;
const CFStringRef   kCVImageBufferCleanApertureKey;
const CFStringRef   kCVImageBufferPreferredCleanApertureKey;
const CFStringRef   kCVImageBufferCleanApertureWidthKey;
const CFStringRef   kCVImageBufferCleanApertureHeightKey;
const CFStringRef   kCVImageBufferCleanApertureHorizontalOffsetKey;
const CFStringRef   kCVImageBufferCleanApertureVerticalOffsetKey;
const CFStringRef   kCVImageBufferFieldCountKey;
const CFStringRef   kCVImageBufferFieldDetailKey;
const CFStringRef   kCVImageBufferFieldDetailTemporalTopFirst;
const CFStringRef   kCVImageBufferFieldDetailTemporalBottomFirst;
const CFStringRef   kCVImageBufferFieldDetailSpatialFirstLineEarly;
const CFStringRef   kCVImageBufferFieldDetailSpatialFirstLineLate;
const CFStringRef   kCVImageBufferPixelAspectRatioKey;
const CFStringRef   kCVImageBufferPixelAspectRatioHorizontalSpacingKey;
const CFStringRef   kCVImageBufferPixelAspectRatioVerticalSpacingKey;
const CFStringRef   kCVImageBufferDisplayDimensionsKey;
const CFStringRef   kCVImageBufferDisplayWidthKey;
const CFStringRef   kCVImageBufferDisplayHeightKey;
const CFStringRef   kCVImageBufferYCbCrMatrixKey;
const CFStringRef   kCVImageBufferYCbCrMatrix_ITU_R_709_2;
const CFStringRef   kCVImageBufferYCbCrMatrix_ITU_R_601_4;
const CFStringRef   kCVImageBufferYCbCrMatrix_SMPTE_240M_1995;
```

Constants
`kCVImageBufferCGColorSpaceKey`
The color space for the buffer (type `CGColorSpaceRef`).

Available in Mac OS X v10.3 and later.

Declared in `CVImageBuffer.h`.

`kCVImageBufferGammaLevelKey`
The gamma level for this buffer (type `CFNumber`).

Available in Mac OS X v10.3 and later.

Declared in `CVImageBuffer.h`.

`kCVImageBufferCleanApertureKey`
The clean aperture for the buffer (type `CFDictionary` , containing the clean aperture width, height, and horizontal and vertical offset key-value pairs).

Available in Mac OS X v10.3 and later.

Declared in `CVImageBuffer.h`.

`kCVImageBufferPreferredCleanApertureKey`
The preferred clean aperture for the buffer (type `CFDictionary` , containing the clean aperture width, height, and horizontal and vertical offset key-value pairs).

Available in Mac OS X v10.3 and later.

Declared in `CVImageBuffer.h`.

`kCVImageBufferCleanApertureWidthKey`
The clean aperture width (type `CFNumber`).

Available in Mac OS X v10.3 and later.

Declared in `CVImageBuffer.h`.

`kCVImageBufferCleanApertureHeightKey`
The clean aperture height (type `CFNumber`).

Available in Mac OS X v10.3 and later.

Declared in `CVImageBuffer.h`.

`kCVImageBufferCleanApertureHorizontalOffsetKey`
The clean aperture horizontal offset (type `CFNumber`).

Available in Mac OS X v10.3 and later.

Declared in `CVImageBuffer.h`.

`kCVImageBufferCleanApertureVerticalOffsetKey`
The clean aperture vertical offset (type `CFNumber`).

Available in Mac OS X v10.3 and later.

Declared in `CVImageBuffer.h`.

`kCVImageBufferFieldCountKey`
The field count for the buffer (type `CFNumber`).

Available in Mac OS X v10.3 and later.

Declared in `CVImageBuffer.h`.

`kCVImageBufferFieldDetailKey`
Specific information about the field of a video frame in the buffer (type `CFDictionary`, containing the temporal bottom first and top first and spacial first-line-early and first-line-late keys).

Available in Mac OS X v10.3 and later.

Declared in `CVImageBuffer.h`.

`kCVImageBufferFieldDetailTemporalTopFirst`
(type `CFString`).

Available in Mac OS X v10.3 and later.

Declared in `CVImageBuffer.h`.

`kCVImageBufferFieldDetailTemporalBottomFirst`
(type `CFString`).

Available in Mac OS X v10.3 and later.

Declared in `CVImageBuffer.h`.

`kCVImageBufferFieldDetailSpatialFirstLineEarly`
(type `CFString`).

Available in Mac OS X v10.3 and later.

Declared in `CVImageBuffer.h`.

`kCVImageBufferFieldDetailSpatialFirstLineLate`
(type `CFString`).

Available in Mac OS X v10.3 and later.

Declared in `CVImageBuffer.h`.

`kCVImageBufferPixelAspectRatioKey`
The pixel aspect ratio of the buffer (type `CFDictionary`, containing the horizontal and vertical spacing keys).

Available in Mac OS X v10.3 and later.

Declared in `CVImageBuffer.h`.

`kCVImageBufferPixelAspectRatioHorizontalSpacingKey`
The horizontal component of the buffer aspect ratio (type `CFNumber`).

Available in Mac OS X v10.3 and later.

Declared in `CVImageBuffer.h`.

`kCVImageBufferPixelAspectRatioVerticalSpacingKey`
The vertical component of the buffer aspect ratio (type `CFNumber`).

Available in Mac OS X v10.3 and later.

Declared in `CVImageBuffer.h`.

`kCVImageBufferDisplayDimensionsKey`
The buffer display dimensions (type `CFDictionary` containing the buffer display width and height keys).

Available in Mac OS X v10.3 and later.

Declared in `CVImageBuffer.h`.

`kCVImageBufferDisplayWidthKey`
The buffer display width (type `CFNumber`).

Available in Mac OS X v10.3 and later.

Declared in `CVImageBuffer.h`.

`kCVImageBufferDisplayHeightKey`
The buffer display height (type `CFNumber`).

Available in Mac OS X v10.3 and later.

Declared in `CVImageBuffer.h`.

`kCVImageBufferYCbCrMatrixKey`
The type of conversion matrix used for this buffer when converting from YCbCr to RGB images (type `CFString`). The value for this key should be one of the following constants: `kCVImageBufferYCbCrMatrix_ITU_R_709_2`, `kCVImageBufferYCbCrMatrix_ITU_R_601_4`, or `kCVImageBufferYCbCrMatrix_SMPTE_240M_1995`.

Available in Mac OS X v10.3 and later.

Declared in `CVImageBuffer.h`.

`kCVImageBufferYCbCrMatrix_ITU_R_709_2`
Specifies the YCbCr to RGB conversion matrix for HDTV digital television (ITU R 709) images.

Available in Mac OS X v10.3 and later.

Declared in `CVImageBuffer.h`.

`kCVImageBufferYCbCrMatrix_ITU_R_601_4`
Specifies the YCbCr to RGB conversion matrix for standard digital television (ITU R 601) images.

Available in Mac OS X v10.3 and later.

Declared in `CVImageBuffer.h`.

`kCVImageBufferYCbCrMatrix_SMPTE_240M_1995`
Specifies the YCbCR to RGB conversion matrix for 1920 x 1135 HDTV (SMPTE 240M 1995).

Available in Mac OS X v10.3 and later.

Declared in `CVImageBuffer.h`.

Discussion
Image buffer attachment keys are stored in a Core Foundation dictionary associated with an image buffer. Note that some of these keys are stored in subdictionaries keyed by a higher-level attribute. For example, the `kCVImageBufferDisplayWidthKey` and `kCVImageBufferDisplayHeightKey` attributes are stored in a Core Foundation dictionary keyed to the `kCVImageBufferDisplayDimensionsKey` attribute.

OpenGL Buffer Attribute Keys

Specify attributes of an OpenGL buffer.

```
const CFStringRef kCVOpenGLBufferWidth;
const CFStringRef kCVOpenGLBufferHeight;
const CFStringRef kCVOpenGLBufferTarget;
const CFStringRef kCVOpenGLBufferInternalFormat;
const CFStringRef kCVOpenGLBufferMaximumMipmapLevel;
```

Constants
`kCVOpenGLBufferWidth`
The width of the buffer.

Available in Mac OS X v10.3 and later.

Declared in `CVOpenGLBuffer.h`.

`kCVOpenGLBufferHeight`
The height of the buffer.

Available in Mac OS X v10.3 and later.

Declared in `CVOpenGLBuffer.h`.

`kCVOpenGLBufferTarget`
The OpenGL target for this buffer.

Available in Mac OS X v10.3 and later.

Declared in `CVOpenGLBuffer.h`.

`kCVOpenGLBufferInternalFormat`
The OpenGL internal format of this buffer.

Available in Mac OS X v10.3 and later.

Declared in `CVOpenGLBuffer.h`.

`kCVOpenGLBufferMaximumMipmapLevel`
The maximum mipmap level for this buffer.

Available in Mac OS X v10.3 and later.

Declared in `CVOpenGLBuffer.h`.

OpenGL Buffer Pool Attribute Keys

Specify attributes associated with an OpenGL buffer pool.

```
const CFStringRef kCVOpenGLBufferPoolMinimumBufferCountKey;
const CFStringRef kCVOpenGLBufferPoolMaximumBufferAgeKey;
```

Constants

`kCVOpenGLBufferPoolMinimumBufferCountKey`
Indicates the minimum number of buffers to keep in the pool (type `CFNumber`).

Available in Mac OS X v10.3 and later.

Declared in `CVOpenGLBufferPool.h`.

`kCVOpenGLBufferPoolMaximumBufferAgeKey`
Indicates how long unused buffers should be kept before they are deallocated (type `CFAbsoluteTime`).

Available in Mac OS X v10.3 and later.

Declared in `CVOpenGLBufferPool.h`.

Discussion
You specify these keys in a Core Foundation dictionary when calling functions such as `CVOpenGLBufferPoolCreate` (page 34).

Pixel Buffer Attribute Keys

Specify attributes associated with a pixel buffer.

```
const CFStringRef kCVPixelBufferPixelFormatTypeKey;
 const CFStringRef kCVPixelBufferMemoryAllocatorKey;
 const CFStringRef kCVPixelBufferWidthKey;
 const CFStringRef kCVPixelBufferHeightKey;
 const CFStringRef kCVPixelBufferExtendedPixelsLeftKey;
 const CFStringRef kCVPixelBufferExtendedPixelsTopKey;
 const CFStringRef kCVPixelBufferExtendedPixelsRightKey;
 const CFStringRef kCVPixelBufferExtendedPixelsBottomKey;
 const CFStringRef kCVPixelBufferBytesPerRowAlignmentKey;
 const CFStringRef kCVPixelBufferCGBitmapContextCompatibilityKey;
 const CFStringRef kCVPixelBufferCGImageCompatibilityKey;
 const CFStringRef kCVPixelBufferOpenGLCompatibilityKey;
```

Constants

`kCVPixelBufferPixelFormatTypeKey`
The pixel format for this buffer (type `CFNumber`, or type `CFArray` containing an array of `CFNumber` types (actually type `OSType`)). For a listing of common pixel formats, see the QuickTime Ice Floe Dispatch 20.

Available in Mac OS X v10.3 and later.

Declared in `CVPixelBuffer.h`.

`kCVPixelBufferMemoryAllocatorKey`
The allocator used with this buffer (type `CFAllocatorRef`).

Available in Mac OS X v10.3 and later.

Declared in `CVPixelBuffer.h`.

`kCVPixelBufferWidthKey`
The width of the pixel buffer (type `CFNumber`).

Available in Mac OS X v10.3 and later.

Declared in `CVPixelBuffer.h`.

`kCVPixelBufferHeightKey`
The height of the pixel buffer (type `CFNumber`).

Available in Mac OS X v10.3 and later.

Declared in `CVPixelBuffer.h`.

`kCVPixelBufferExtendedPixelsLeftKey`
The number of pixels padding the left of the image (type `CFNumber`).

Available in Mac OS X v10.3 and later.

Declared in `CVPixelBuffer.h`.

`kCVPixelBufferExtendedPixelsTopKey`
The number of pixels padding the top of the image (type `CFNumber`).

Available in Mac OS X v10.3 and later.

Declared in `CVPixelBuffer.h`.

`kCVPixelBufferExtendedPixelsRightKey`
The number of pixels padding the right of the image (type `CFNumber`).

Available in Mac OS X v10.3 and later.

Declared in `CVPixelBuffer.h`.

`kCVPixelBufferExtendedPixelsBottomKey`
The number of pixels padding the bottom of the image (type `CFNumber`).

Available in Mac OS X v10.3 and later.

Declared in `CVPixelBuffer.h`.

`kCVPixelBufferBytesPerRowAlignmentKey`
Indicates the number of bytes per row in the pixel buffer (type `CFNumber`).

Available in Mac OS X v10.3 and later.

Declared in `CVPixelBuffer.h`.

`kCVPixelBufferCGBitmapContextCompatibilityKey`
Indicates whether the pixel buffer is compatible with Core Graphics bitmap contexts (type `CFBoolean`).

Available in Mac OS X v10.3 and later.

Declared in `CVPixelBuffer.h`.

`kCVPixelBufferCGImageCompatibilityKey`
Indicates whether the pixel buffer is compatible with `CGImage` types (type `CFBoolean`).

Available in Mac OS X v10.3 and later.

Declared in `CVPixelBuffer.h`.

`kCVPixelBufferOpenGLCompatibilityKey`
Indicates whether the pixel buffer is compatible with OpenGL contexts (type `CFBoolean`).

Available in Mac OS X v10.3 and later.

Declared in `CVPixelBuffer.h`.

Discussion
You specify these keys in a Core Foundation dictionary when calling functions such as `CVPixelBufferCreate` (page 45).

Pixel Buffer Pool Attribute Keys

Specify attributes associated with a pixel buffer pool.

```
const CFStringRef kCVPixelBufferPoolMinimumBufferCountKey;
const CFStringRef kCVPixelBufferPoolMaximumBufferAgeKey;
```

Constants
`kCVPixelBufferPoolMinimumBufferCountKey`
The minimum number of buffers allowed in the pixel buffer pool (type `CFNumber`).

Available in Mac OS X v10.3 and later.

Declared in `CVPixelBufferPool.h`.

`kCVPixelBufferPoolMaximumBufferAgeKey`
The maximum allowable age for a buffer in the pixel buffer pool (type `CFAbsoluteTime`).

Available in Mac OS X v10.3 and later.

Declared in `CVPixelBufferPool.h`.

Discussion
You specify these keys in a Core Foundation dictionary when calling functions such as `CVPixelBufferPoolCreate` (page 57).

Pixel Format Description Keys

Specify attributes of a pixel format.

```
const CFStringRef kCVPixelFormatName;
const CFStringRef kCVPixelFormatConstant;
const CFStringRef kCVPixelFormatCodecType;
const CFStringRef kCVPixelFormatFourCC;
const CFStringRef kCVPixelFormatPlanes;
const CFStringRef kCVPixelFormatBlockWidth;
const CFStringRef kCVPixelFormatBlockHeight;
const CFStringRef kCVPixelFormatBitsPerBlock;
const CFStringRef kCVPixelFormatBlockHorizontalAlignment;
const CFStringRef kCVPixelFormatBlockVerticalAlignment;
const CFStringRef kCVPixelFormatHorizontalSubsampling;
const CFStringRef kCVPixelFormatVerticalSubsampling;

const CFStringRef kCVPixelFormatOpenGLFormat;
const CFStringRef kCVPixelFormatOpenGLType;
const CFStringRef kCVPixelFormatOpenGLInternalFormat;
```

```
const CFStringRef kCVPixelFormatCGBitmapInfo;

const CFStringRef kCVPixelFormatQDCompatibility;
const CFStringRef kCVPixelFormatCGBitmapContextCompatibility;
const CFStringRef kCVPixelFormatCGImageCompatibility;
const CFStringRef kCVPixelFormatOpenGLCompatibility;

const CFStringRef kCVPixelFormatFillExtendedPixelsCallback;
```

Constants

`kCVPixelFormatName`

The name of the pixel format (type `CFString`). This should be the same as the codec name you would use in QuickTime.

Available in Mac OS X v10.3 and later.

Declared in `CVPixelFormatDescription.h`.

`kCVPixelFormatConstant`

The pixel format constant for QuickTime.

Available in Mac OS X v10.3 and later.

Declared in `CVPixelFormatDescription.h`.

`kCVPixelFormatCodecType`

The codec type (type `CFString`). For example, `'2vuy'` or `k422YpCbCr8CodecType`.

Available in Mac OS X v10.3 and later.

Declared in `CVPixelFormatDescription.h`.

`kCVPixelFormatFourCC`

The Microsoft FourCC equivalent code for this pixel format (type `CFString`).

Available in Mac OS X v10.3 and later.

Declared in `CVPixelFormatDescription.h`.

`kCVPixelFormatPlanes`

The number of image planes associated with this format (type `CFNumber`. Each plane may contain a single component or an interleaved set of components. Note that if your pixel format is not planar, you can put the required format keys at the top-level dictionary.

Available in Mac OS X v10.3 and later.

Declared in `CVPixelFormatDescription.h`.

`kCVPixelFormatBlockWidth`

The width, in pixels, of the smallest byte-addressable group of pixels (type `CFNumber`. Used to assist with allocating memory for pixel formats that don't have an integer value for bytes per pixel. Assumed to be 1 if this key is not present. Here are some examples of block widths for standard pixel formats:

- 8-bit luminance only, block width is 1, the bits per block value is 8.
- 16-bit 1555 RGB, block width is 1, the bits per block value is 16.
- 32-bit 8888 ARGB, block width is 1, the bits per block value is 32.
- 2vuy (CbYCrY), block width is 2, the bits per block value is 32.
- 1-bit bitmap, block width is 8, the bits per block value is 8.
- v210, block width is 6, the bits per block value is 128 .

Available in Mac OS X v10.3 and later.

Declared in `CVPixelFormatDescription.h`.

`kCVPixelFormatBlockHeight`
The height, in pixels, of the smallest byte-addressable group of pixels (type `CFNumber`). Assumed to be one if this key is not present.

Available in Mac OS X v10.3 and later.

Declared in `CVPixelFormatDescription.h`.

`kCVPixelFormatBitsPerBlock`
The number of bits per block. For simple pixel formats, this value is the same as the traditional bits-per-pixel value. This key is mandatory in pixel format descriptions. See the description for `kCVPixelFormatBlockWidth` for examples of bits-per-block values.

Available in Mac OS X v10.3 and later.

Declared in `CVPixelFormatDescription.h`.

`kCVPixelFormatBlockHorizontalAlignment`
The horizontal alignment requirements of this format (type `CFNumber`). For example,the alignment for v210 would be '8' here for the horizontal case to match the standard v210 row alignment value of 48. Assumed to be 1 if this key is not present.

Available in Mac OS X v10.3 and later.

Declared in `CVPixelFormatDescription.h`.

`kCVPixelFormatBlockVerticalAlignment`
The vertical alignment requirements of this format (type `CFNumber`). Assumed to be 1 if this key is not present.

Available in Mac OS X v10.3 and later.

Declared in `CVPixelFormatDescription.h`.

`kCVPixelFormatHorizontalSubsampling`
Horizontal subsampling information for this plane (type `CFNumber`). Assumed to be 1 if this key is not present.

Available in Mac OS X v10.3 and later.

Declared in `CVPixelFormatDescription.h`.

`kCVPixelFormatVerticalSubsampling`
Vertical subsampling information for this plane (type `CFNumber`). Assumed to be 1 if this key is not present.

Available in Mac OS X v10.3 and later.

Declared in `CVPixelFormatDescription.h`.

`kCVPixelFormatOpenGLFormat`
The OpenGL format used to describe this image plane (if applicable). See the OpenGL specification for possible values.

Available in Mac OS X v10.3 and later.

Declared in `CVPixelFormatDescription.h`.

`kCVPixelFormatOpenGLType`
The OpenGL type to describe this image plane (if applicable). See the OpenGL specification for possible values.

Available in Mac OS X v10.3 and later.

Declared in `CVPixelFormatDescription.h`.

`kCVPixelFormatOpenGLInternalFormat`
The OpenGL internal format for this pixel format (if applicable). See the OpenGL specification for possible values.

Available in Mac OS X v10.3 and later.

Declared in `CVPixelFormatDescription.h`.

`kCVPixelFormatCGBitmapInfo`
The Core Graphics bitmap information for this pixel format (if applicable).

Available in Mac OS X v10.3 and later.

Declared in `CVPixelFormatDescription.h`.

`kCVPixelFormatQDCompatibility`
Indicates whether this format is compatible with QuickDraw (type `CFBoolean`).

Available in Mac OS X v10.3 and later.

Declared in `CVPixelFormatDescription.h`.

`kCVPixelFormatCGBitmapContextCompatibility`
Indicates whether this format is compatible with Core Graphics bitmap contexts(type `CFBoolean`).

Available in Mac OS X v10.3 and later.

Declared in `CVPixelFormatDescription.h`.

`kCVPixelFormatCGImageCompatibility`
Indicates whether this format is compatible with the `CGImage` type (type `CFBoolean`).

Available in Mac OS X v10.3 and later.

Declared in `CVPixelFormatDescription.h`.

`kCVPixelFormatOpenGLCompatibility`
Indicates whether this format is compatible with OpenGL (type `CFBoolean`).

Available in Mac OS X v10.3 and later.

Declared in `CVPixelFormatDescription.h`.

`kCVPixelFormatFillExtendedPixelsCallback`
Specifies a custom extended pixel fill algorithm (type `CFData`). See `CVFillExtendedPixelsCallBack` (page 65) and `CVFillExtendedPixelsCallbackData` (page 67) for more information.

Available in Mac OS X v10.3 and later.

Declared in `CVPixelFormatDescription.h`.

Discussion
If you need to define a custom pixel format, you must specify these keys in a Core Foundation dictionary. For information about registering your pixel format, see Technical Q&A 1401: Registering Custom Pixel Formats with QuickTime and Core Video.

In most cases you do not need to specify your own pixel format.

SMPTE State Flags

Flags that describe the SMPTE time state.

```
enum{
    kCVSMPTETimeValid     = (1L << 0),
    kCVSMPTETimeRunning   = (1L << 1)
};
```

Constants

`kCVSMPTETimeValid`

The full time is valid.

Available in Mac OS X v10.3 and later.

Declared in `CVBase.h`.

`kCVSMPTETimeRunning`

Time is running.

Available in Mac OS X v10.3 and later.

Declared in `CVBase.h`.

Discussion

You use these values in the `CVSMPTETime` (page 71) structure.

SMPTE Time Types

Constants that describe the type of SMPTE time.

```
enum{
    kCVSMPTETimeType24        = 0,
    kCVSMPTETimeType25        = 1,
    kCVSMPTETimeType30Drop    = 2,
    kCVSMPTETimeType30        = 3,
    kCVSMPTETimeType2997      = 4,
    kCVSMPTETimeType2997Drop  = 5,
    kCVSMPTETimeType60        = 6,
    kCVSMPTETimeType5994      = 7
};
```

Constants

`kCVSMPTETimeType24`

24 frames per second (standard film).

Available in Mac OS X v10.3 and later.

Declared in `CVBase.h`.

`kCVSMPTETimeType25`

25 frames per second (standard PAL).

Available in Mac OS X v10.3 and later.

Declared in `CVBase.h`.

`kCVSMPTETimeType30Drop`

30 drop frame.

Available in Mac OS X v10.3 and later.

Declared in `CVBase.h`.

`kCVSMPTETimeType30`
30 frames per second.

Available in Mac OS X v10.3 and later.

Declared in `CVBase.h`.

`kCVSMPTETimeType2997`
29.97 frames per second (standard NTSC).

Available in Mac OS X v10.3 and later.

Declared in `CVBase.h`.

`kCVSMPTETimeType2997Drop`
29.97 drop frame.

Available in Mac OS X v10.3 and later.

Declared in `CVBase.h`.

`kCVSMPTETimeType60`
60 frames per second.

Available in Mac OS X v10.3 and later.

Declared in `CVBase.h`.

`kCVSMPTETimeType5994`
59.94 frames per second.

Available in Mac OS X v10.3 and later.

Declared in `CVBase.h`.

Discussion
You use these values in the `CVSMPTETime` (page 71) structure.

Result Codes

The table below lists the result codes returned for Core Video. Note that these result codes are of type `CVReturn`, not type `OSErr`.

Result Code	Value	Description
`kCVReturnSuccess`	0	No error Available in Mac OS X v10.3 and later.
`kCVReturnFirst`	-6660	Placeholder to mark the beginning of Core Video result codes (not returned by any functions). Available in Mac OS X v10.3 and later.
`kCVReturnError`	-6660	An otherwise undefined error occurred. Available in Mac OS X v10.3 and later.
`kCVReturnInvalidArgument`	-6661	Invalid function parameter. For example, out of range or the wrong type. Available in Mac OS X v10.3 and later.

Result Code	Value	Description
kCVReturnAllocationFailed	-6662	Memory allocation for a buffer or buffer pool failed. Available in Mac OS X v10.3 and later.
kCVReturnInvalidDisplay	-6670	The display specified when creating a display link is invalid. Available in Mac OS X v10.3 and later.
kCVReturnDisplayLinkAlreadyRunning	-6671	The specified display link is already running. Available in Mac OS X v10.3 and later.
kCVReturnDisplayLinkNotRunning	-6672	The specified display link is not running. Available in Mac OS X v10.3 and later.
kCVReturnDisplayLinkCallbacksNotSet	-6673	No callback registered for the specified display link. You must set either the output callback or both the render and display callbacks. Available in Mac OS X v10.3 and later.
kCVReturnInvalidPixelFormat	-6680	The buffer does not support the specified pixel format. Available in Mac OS X v10.3 and later.
kCVReturnInvalidSize	-6681	The buffer cannot support the requested buffer size (usually too big). Available in Mac OS X v10.3 and later.
kCVReturnInvalidPixelBufferAttributes	-6682	A buffer cannot be created with the specified attributes. Available in Mac OS X v10.3 and later.
kCVReturnPixelBufferNotOpenGLCompatible	-6683	The pixel buffer is not compatible with OpenGL due to an unsupported buffer size, pixel format, or attribute. Available in Mac OS X v10.3 and later.
kCVReturnPoolAllocationFailed	-6690	Allocation for a buffer pool failed, most likely due to a lack of resources. Check to make sure your parameters are in range. Available in Mac OS X v10.3 and later.
kCVReturnInvalidPoolAttributes	-6691	A buffer pool cannot be created with the specified attributes. Available in Mac OS X v10.3 and later.

Result Code	Value	Description
kCVReturnLast	-6699	Placeholder to mark the end of Core Video result codes (not returned by any functions). Available in Mac OS X v10.3 and later.

Document Revision History

This table describes the changes to *Core Video Reference*.

Date	Notes
2007-03-22	Made minor formatting changes.
2005-11-09	Fixed error in CVPixelBufferFillExtendedPixels abstract. Added links to related Technical Q&As for custom CVFillExtendedPixels callbacks and CVPixelFormatDescriptionRegisterDescriptionWithPixelFormatType.
2005-07-07	Fixed links to Apple documentation.
2005-04-29	New document that describes the C API for obtaining and manipulating individual video frames.

Document Revision History

Index

C

CVBuffer Attachment Keys 73
CVBuffer Attachment Modes 74
CVBuffer Attribute Keys 74
CVBufferGetAttachment function 13
CVBufferGetAttachments function 14
CVBufferPropagateAttachments function 14
CVBufferRef data type 67
CVBufferRelease function 15
CVBufferRemoveAllAttachments function 15
CVBufferRemoveAttachment function 16
CVBufferRetain function 16
CVBufferSetAttachment function 17
CVBufferSetAttachments function 18
CVDisplayLinkCreateWithActiveCGDisplays function 18
CVDisplayLinkCreateWithCGDisplay function 19
CVDisplayLinkCreateWithCGDisplays function 20
CVDisplayLinkCreateWithOpenGLDisplayMask function 20
CVDisplayLinkGetActualOutputVideoRefreshPeriod function 21
CVDisplayLinkGetCurrentCGDisplay function 21
CVDisplayLinkGetCurrentTime function 22
CVDisplayLinkGetNominalOutputVideoRefreshPeriod function 22
CVDisplayLinkGetOutputVideoLatency function 23
CVDisplayLinkGetTypeID function 23
CVDisplayLinkIsRunning function 23
CVDisplayLinkOutputCallback callback 63
CVDisplayLinkRef data type 67
CVDisplayLinkRelease function 24
CVDisplayLinkRetain function 24
CVDisplayLinkSetCurrentCGDisplay function 25
CVDisplayLinkSetCurrentCGDisplayFromOpenGLContext function 25
CVDisplayLinkSetOutputCallback function 26
CVDisplayLinkStart function 27
CVDisplayLinkStop function 27
CVDisplayLinkTranslateTime function 28
CVFillExtendedPixelsCallBack callback 65
CVFillExtendedPixelsCallbackData structure 67
CVGetCurrentHostTime function 29
CVGetHostClockFrequency function 29
CVGetHostClockMinimumTimeDelta function 29
CVImageBufferGetCleanRect function 30
CVImageBufferGetColorSpace function 30
CVImageBufferGetDisplaySize function 31
CVImageBufferGetEncodedSize function 31
CVImageBufferRef data type 68
CVOpenGLBufferAttach function 32
CVOpenGLBufferCreate function 32
CVOpenGLBufferGetAttributes function 33
CVOpenGLBufferGetTypeID function 33
CVOpenGLBufferPoolCreate function 34
CVOpenGLBufferPoolCreateOpenGLBuffer function 34
CVOpenGLBufferPoolGetAttributes function 35
CVOpenGLBufferPoolGetOpenGLBufferAttributes function 35
CVOpenGLBufferPoolGetTypeID function 36
CVOpenGLBufferPoolRef data type 69
CVOpenGLBufferPoolRelease function 36
CVOpenGLBufferPoolRetain function 37
CVOpenGLBufferRef data type 69
CVOpenGLBufferRelease function 37
CVOpenGLBufferRetain function 37
CVOpenGLTextureCacheCreate function 38
CVOpenGLTextureCacheCreateTextureFromImage function 39
CVOpenGLTextureCacheFlush function 39
CVOpenGLTextureCacheGetTypeID function 40
CVOpenGLTextureCacheRef data type 69
CVOpenGLTextureCacheRelease function 40
CVOpenGLTextureCacheRetain function 41
CVOpenGLTextureGetCleanTexCoords function 41
CVOpenGLTextureGetName function 42
CVOpenGLTextureGetTarget function 43
CVOpenGLTextureGetTypeID function 43
CVOpenGLTextureIsFlipped function 43
CVOpenGLTextureRef data type 69
CVOpenGLTextureRelease function 44

CVOpenGLTextureRetain function 44
CVOptionFlags data type 68
CVPixelBufferCreate function 45
CVPixelBufferCreateResolvedAttributesDictionary function 46
CVPixelBufferCreateWithBytes function 46
CVPixelBufferCreateWithPlanarBytes function 48
CVPixelBufferFillExtendedPixels function 49
CVPixelBufferGetBaseAddress function 49
CVPixelBufferGetBaseAddressOfPlane function 50
CVPixelBufferGetBytesPerRow function 51
CVPixelBufferGetBytesPerRowOfPlane function 51
CVPixelBufferGetDataSize function 52
CVPixelBufferGetExtendedPixels function 52
CVPixelBufferGetHeight function 53
CVPixelBufferGetHeightOfPlane function 53
CVPixelBufferGetPixelFormatType function 54
CVPixelBufferGetPlaneCount function 54
CVPixelBufferGetTypeID function 54
CVPixelBufferGetWidth function 55
CVPixelBufferGetWidthOfPlane function 55
CVPixelBufferIsPlanar function 56
CVPixelBufferLockBaseAddress function 56
CVPixelBufferPoolCreate function 57
CVPixelBufferPoolCreatePixelBuffer function 57
CVPixelBufferPoolGetAttributes function 58
CVPixelBufferPoolGetPixelBufferAttributes function 59
CVPixelBufferPoolGetTypeID function 59
CVPixelBufferPoolRef data type 70
CVPixelBufferPoolRelease function 59
CVPixelBufferPoolRetain function 60
CVPixelBufferRef data type 70
CVPixelBufferRelease function 60
CVPixelBufferReleaseBytesCallback callback 65
CVPixelBufferReleasePlanarBytesCallback callback 66
CVPixelBufferRetain function 61
CVPixelBufferUnlockBaseAddress function 61
CVPixelFormatDescriptionArrayCreateWithAllPixelFormatTypes function 62
CVPixelFormatDescriptionCreateWithPixelFormatType function 62
CVPixelFormatDescriptionRegisterDescriptionWithPixelFormatType function 63
CVReturn data type 70
CVSMPTETime structure 71
CVTime Constants 75
CVTime structure 71
CVTime Values 75
CVTimeStamp Flags 75
CVTimeStamp structure 72

I

Image Buffer Attachment Keys 77

K

kCVAttachmentMode_ShouldNotPropagate constant 74
kCVAttachmentMode_ShouldPropagate constant 74
kCVBufferMovieTimeKey constant 73
kCVBufferNonPropagatedAttachmentsKey constant 74
kCVBufferPropagatedAttachmentsKey constant 74
kCVBufferTimeScaleKey constant 74
kCVBufferTimeValueKey constant 73
kCVImageBufferCGColorSpaceKey constant 77
kCVImageBufferCleanApertureHeightKey constant 78
kCVImageBufferCleanApertureHorizontalOffsetKey constant 78
kCVImageBufferCleanApertureKey constant 78
kCVImageBufferCleanApertureVerticalOffsetKey constant 78
kCVImageBufferCleanApertureWidthKey constant 78
kCVImageBufferDisplayDimensionsKey constant 79
kCVImageBufferDisplayHeightKey constant 79
kCVImageBufferDisplayWidthKey constant 79
kCVImageBufferFieldCountKey constant 78
kCVImageBufferFieldDetailKey constant 78
kCVImageBufferFieldDetailSpatialFirstLineEarly constant 79
kCVImageBufferFieldDetailSpatialFirstLineLate constant 79
kCVImageBufferFieldDetailTemporalBottomFirst constant 78
kCVImageBufferFieldDetailTemporalTopFirst constant 78
kCVImageBufferGammaLevelKey constant 77
kCVImageBufferPixelAspectRatioHorizontalSpacingKey constant 79
kCVImageBufferPixelAspectRatioKey constant 79
kCVImageBufferPixelAspectRatioVerticalSpacingKey constant 79
kCVImageBufferPreferredCleanApertureKey constant 78
kCVImageBufferYCbCrMatrixKey constant 79
kCVImageBufferYCbCrMatrix_ITU_R_601_4 constant 80
kCVImageBufferYCbCrMatrix_ITU_R_709_2 constant 80

kCVImageBufferYCbCrMatrix_SMPTE_240M_1995 constant 80
kCVIndefiniteTime constant 75
kCVOpenGLBufferHeight constant 80
kCVOpenGLBufferInternalFormat constant 80
kCVOpenGLBufferMaximumMipmapLevel constant 81
kCVOpenGLBufferPoolMaximumBufferAgeKey constant 81
kCVOpenGLBufferPoolMinimumBufferCountKey constant 81
kCVOpenGLBufferTarget constant 80
kCVOpenGLBufferWidth constant 80
kCVPixelBufferBytesPerRowAlignmentKey constant 82
kCVPixelBufferCGBitmapContextCompatibilityKey constant 82
kCVPixelBufferCGImageCompatibilityKey constant 82
kCVPixelBufferExtendedPixelsBottomKey constant 82
kCVPixelBufferExtendedPixelsLeftKey constant 82
kCVPixelBufferExtendedPixelsRightKey constant 82
kCVPixelBufferExtendedPixelsTopKey constant 82
kCVPixelBufferHeightKey constant 82
kCVPixelBufferMemoryAllocatorKey constant 82
kCVPixelBufferOpenGLCompatibilityKey constant 83
kCVPixelBufferPixelFormatTypeKey constant 81
kCVPixelBufferPoolMaximumBufferAgeKey constant 83
kCVPixelBufferPoolMinimumBufferCountKey constant 83
kCVPixelBufferWidthKey constant 82
kCVPixelFormatBitsPerBlock constant 85
kCVPixelFormatBlockHeight constant 85
kCVPixelFormatBlockHorizontalAlignment constant 85
kCVPixelFormatBlockVerticalAlignment constant 85
kCVPixelFormatBlockWidth constant 84
kCVPixelFormatCGBitmapContextCompatibility constant 86
kCVPixelFormatCGBitmapInfo constant 86
kCVPixelFormatCGImageCompatibility constant 86
kCVPixelFormatCodecType constant 84
kCVPixelFormatConstant constant 84
kCVPixelFormatFillExtendedPixelsCallback constant 86
kCVPixelFormatFourCC constant 84
kCVPixelFormatHorizontalSubsampling constant 85
kCVPixelFormatName constant 84
kCVPixelFormatOpenGLCompatibility constant 86
kCVPixelFormatOpenGLFormat constant 85
kCVPixelFormatOpenGLInternalFormat constant 86
kCVPixelFormatOpenGLType constant 85
kCVPixelFormatPlanes constant 84
kCVPixelFormatQDCompatibility constant 86
kCVPixelFormatVerticalSubsampling constant 85
kCVReturnAllocationFailed constant 89
kCVReturnDisplayLinkAlreadyRunning constant 89
kCVReturnDisplayLinkCallbacksNotSet constant 89
kCVReturnDisplayLinkNotRunning constant 89
kCVReturnError constant 88
kCVReturnFirst constant 88
kCVReturnInvalidArgument constant 88
kCVReturnInvalidDisplay constant 89
kCVReturnInvalidPixelBufferAttributes constant 89
kCVReturnInvalidPixelFormat constant 89
kCVReturnInvalidPoolAttributes constant 89
kCVReturnInvalidSize constant 89
kCVReturnLast constant 90
kCVReturnPixelBufferNotOpenGLCompatible constant 89
kCVReturnPoolAllocationFailed constant 89
kCVReturnSuccess constant 88
kCVSMPTETimeRunning constant 87
kCVSMPTETimeType24 constant 87
kCVSMPTETimeType25 constant 87
kCVSMPTETimeType2997 constant 88
kCVSMPTETimeType2997Drop constant 88
kCVSMPTETimeType30 constant 88
kCVSMPTETimeType30Drop constant 87
kCVSMPTETimeType5994 constant 88
kCVSMPTETimeType60 constant 88
kCVSMPTETimeValid constant 87
kCVTimeIsIndefinite constant 75
kCVTimeStampBottomField constant 76
kCVTimeStampHostTimeValid constant 76
kCVTimeStampIsInterlaced constant 77
kCVTimeStampRateScalarValid constant 76
kCVTimeStampSMPTETimeValid constant 76
kCVTimeStampTopField constant 76
kCVTimeStampVideoHostTimeValid constant 77
kCVTimeStampVideoRefreshPeriodValid constant 76
kCVTimeStampVideoTimeValid constant 76
kCVZeroTime constant 75

O

OpenGL Buffer Attribute Keys 80
OpenGL Buffer Pool Attribute Keys 81

P

Pixel Buffer Attribute Keys 81
Pixel Buffer Pool Attribute Keys 83
Pixel Format Description Keys 83

S

SMPTE State Flags 86
SMPTE Time Types 87