Core Media Framework Reference

Audio & Video

2010-03-23

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

Simultaneously published in the United States and Canada.

Contents

Core MediaFramework Reference

Framework:	CoreMedia.framework
Declared in	CMAttachment.h CMBase.h CMBlockBuffer.h CMBufferQueue.h CMFormatDescription.h CMSampleBuffer.h CMTime.h CMTimeRange.h

The Core Media framework provides a low-level C interface for managing and and playing audio-visual media in your iOS application.

Opaque Types

PART I

Opaque Types

CMBlockBuffer Reference

Derived From:	*CFType Reference*
Framework:	CoreMedia.framework
Declared in	CMBlockBuffer.h

Overview

This document describes the Core Foundation objects that you use to move blocks of memory through a processing system.

A CMBlockBuffer represents a contiguous range of data offsets (from zero to `CMBlockBufferGetDataLength` (page 16)) across a possibly noncontiguous memory region composed of memory blocks and buffer references which in turn could refer to additional regions.

Functions

CMBlockBufferAccessDataBytes

```
OSStatus CMBlockBufferAccessDataBytes (
   CMBlockBufferRef theBuffer,
   size_t offset,
   size_t length,
   void *temporaryBlock,
   char **returnedPointer
);
```

Parameters

theBuffer

offset

length

temporaryBlock

returnedPointer

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMBlockBuffer.h

CMBlockBufferAppendBufferReference

```
OSStatus CMBlockBufferAppendBufferReference (
   CMBlockBufferRef theBuffer,
   CMBlockBufferRef targetBBuf,
   size_t offsetToData,
   size_t dataLength,
   CMBlockBufferFlags flags
);
```

Parameters

theBuffer

targetBBuf

offsetToData

dataLength

flags

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMBlockBuffer.h

CMBlockBufferAppendMemoryBlock

```
OSStatus CMBlockBufferAppendMemoryBlock (
   CMBlockBufferRef theBuffer,
   void *memoryBlock,
   size_t blockLength,
   CFAllocatorRef blockAllocator,
   const CMBlockBufferCustomBlockSource *customBlockSource,
   size_t offsetToData,
   size_t dataLength,
   CMBlockBufferFlags flags
);
```

Parameters

theBuffer

memoryBlock

blockLength

blockAllocator

customBlockSource

offsetToData

dataLength

flags

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMBlockBuffer.h

CMBlockBufferAssureBlockMemory

```
OSStatus CMBlockBufferAssureBlockMemory (
   CMBlockBufferRef theBuffer
);
```

Parameters

theBuffer

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMBlockBuffer.h

CMBlockBufferCopyDataBytes

```
OSStatus CMBlockBufferCopyDataBytes (
   CMBlockBufferRef theSourceBuffer,
   size_t offsetToData,
   size_t dataLength,
   void *destination
);
```

Parameters

theSourceBuffer

offsetToData

dataLength

destination

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMBlockBuffer.h

CMBlockBufferCreateContiguous

```
OSStatus CMBlockBufferCreateContiguous (
   CFAllocatorRef structureAllocator,
   CMBlockBufferRef sourceBuffer,
   CFAllocatorRef blockAllocator,
   const CMBlockBufferCustomBlockSource *customBlockSource,
   size_t offsetToData,
   size_t dataLength,
   CMBlockBufferFlags flags,
   CMBlockBufferRef *newBBufOut
);
```

Parameters

structureAllocator

sourceBuffer

blockAllocator

customBlockSource

offsetToData

dataLength

flags

newBBufOut

Return Value

Discussion

Availability

Available in iOS 4.0 and later.

Declared In

CMBlockBuffer.h

CMBlockBufferCreateEmpty

```
OSStatus CMBlockBufferCreateEmpty (
   CFAllocatorRef structureAllocator,
   uint32_t subBlockCapacity,
   CMBlockBufferFlags flags,
   CMBlockBufferRef *newBBufOut
);
```

Parameters

structureAllocator

subBlockCapacity

flags

newBBufOut

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMBlockBuffer.h

CMBlockBufferCreateWithBufferReference

```
OSStatus CMBlockBufferCreateWithBufferReference (
   CFAllocatorRef structureAllocator,
   CMBlockBufferRef targetBuffer,
   size_t offsetToData,
   size_t dataLength,
   CMBlockBufferFlags flags,
   CMBlockBufferRef *newBBufOut
);
```

Parameters

structureAllocator

targetBuffer

offsetToData

dataLength

flags

newBBufOut

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMBlockBuffer.h

CMBlockBufferCreateWithMemoryBlock

```
OSStatus CMBlockBufferCreateWithMemoryBlock (
   CFAllocatorRef structureAllocator,
   void *memoryBlock,
   size_t blockLength,
   CFAllocatorRef blockAllocator,
   const CMBlockBufferCustomBlockSource *customBlockSource,
   size_t offsetToData,
   size_t dataLength,
   CMBlockBufferFlags flags,
   CMBlockBufferRef *newBBufOut
);
```

Parameters

structureAllocator

memoryBlock

blockLength

blockAllocator

customBlockSource

offsetToData

dataLength

flags

newBBufOut

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMBlockBuffer.h

CMBlockBufferFillDataBytes

```
OSStatus CMBlockBufferFillDataBytes (
   char fillByte,
   CMBlockBufferRef destinationBuffer,
   size_t offsetIntoDestination,
   size_t dataLength
);
```

Parameters

fillByte

destinationBuffer

offsetIntoDestination

dataLength

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMBlockBuffer.h

CMBlockBufferGetDataLength

```
size_t CMBlockBufferGetDataLength (
   CMBlockBufferRef theBuffer
);
```

Parameters

theBuffer

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMBlockBuffer.h

CMBlockBufferGetDataPointer

```
OSStatus CMBlockBufferGetDataPointer (
   CMBlockBufferRef theBuffer,
   size_t offset,
   size_t *lengthAtOffset,
   size_t *totalLength,
   char **dataPointer
);
```

Parameters

theBuffer

offset

lengthAtOffset

totalLength

dataPointer

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMBlockBuffer.h

CMBlockBufferGetTypeID

```
CFTypeID CMBlockBufferGetTypeID (
   void
);
```

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMBlockBuffer.h

CMBlockBufferIsEmpty

```
Boolean CMBlockBufferIsEmpty (
   CMBlockBufferRef theBuffer
);
```

Parameters

theBuffer

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMBlockBuffer.h

CMBlockBufferIsRangeContiguous

```
Boolean CMBlockBufferIsRangeContiguous (
   CMBlockBufferRef theBuffer,
   size_t offset,
   size_t length
);
```

Parameters

theBuffer

offset

length

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMBlockBuffer.h

CMBlockBufferReplaceDataBytes

```
OSStatus CMBlockBufferReplaceDataBytes (
   const void *sourceBytes,
   CMBlockBufferRef destinationBuffer,
   size_t offsetIntoDestination,
   size_t dataLength
);
```

Parameters

sourceBytes

destinationBuffer

offsetIntoDestination

dataLength

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMBlockBuffer.h

Data Types

CMBlockBufferCustomBlockSource

Structure to support custom memory allocation for the block used in a CMBlockBuffer.

```
typedef  struct {
    uint32_t    version;
    void        *(*AllocateBlock)(void *refCon, size_t sizeInBytes);
    void        (*FreeBlock)(void *refCon, void *doomedMemoryBlock, size_t
sizeInBytes);
    void        *refCon;
} CMBlockBufferCustomBlockSource;
```

Fields

`AllocateBlock`

The function to allocate memory.

This must be non-0 if the CMBlockBuffer code will need to call for allocation. (This is not required if a previously-obtained memory block is provided to the CMBlockBuffer API.)

`FreeBlock`

A function to call once when the CMBlockBuffer is disposed.

Pass `NULL` if you don't want a function to be called after disposal. The function will not be called if no memory block is ever allocated or supplied.

`refCon`

Contextual information passed to both the allocate and free function calls.

You responsible for its disposal (if any) during the FreeBlock callback.

Discussion
This structure allows a client to provide a custom facility for obtaining the memory block to be used in a CMBlockBuffer. You use this structure with functions that accept a memory block allocator.

Availability
Available in iOS 4.0 and later.

Declared In
`CMBlockBuffer.h`

CMBlockBufferFlags

Type used for parameters containing CMBlockBuffer feature and control flags.

```
typedef uint32_t CMBlockBufferFlags;
```

Discussion
For possible values, see "CMBlockBuffer Flags" (page 22).

Availability
Available in iOS 4.0 and later.

Declared In
`CMBlockBuffer.h`

CMBlockBufferRef

A reference to a CMBlockBuffer object.

```
typedef struct OpaqueCMBlockBuffer *CMBlockBufferRef;
```

Availability
Available in iOS 4.0 and later.

Declared In
`CMBlockBuffer.h`

Constants

Error Flags

Constants for errors returned from the CMBlockBuffer APIs.

```
enum {
    kCMBlockBufferNoErr                             = 0,
    kCMBlockBufferStructureAllocationFailedErr      = -12700,
    kCMBlockBufferBlockAllocationFailedErr          = -12701,
    kCMBlockBufferBadCustomBlockSourceErr           = -12702,
    kCMBlockBufferBadOffsetParameterErr             = -12703,
    kCMBlockBufferBadLengthParameterErr             = -12704,
    kCMBlockBufferBadPointerParameterErr            = -12705,
    kCMBlockBufferEmptyBBufErr                      = -12706,
    kCMBlockBufferUnallocatedBlockErr               = -12707,
};
```

Constants

`kCMBlockBufferNoErr`
Indicates the operation completed successfully.

Available in iOS 4.0 and later.

Declared in `CMBlockBuffer.h`.

`kCMBlockBufferStructureAllocationFailedErr`
Indicates that CMBlockBuffer-creating API received a failure from the CFAllocator provided for CMBlockBuffer construction.

Available in iOS 4.0 and later.

Declared in `CMBlockBuffer.h`.

`kCMBlockBufferBlockAllocationFailedErr`
Indicates that the allocator provided to allocate a memory block (as distinct from CMBlockBuffer structures) failed.

Available in iOS 4.0 and later.

Declared in `CMBlockBuffer.h`.

`kCMBlockBufferBadCustomBlockSourceErr`
Indicates that the custom block source's Allocate() routine was NULL when an allocation was attempted.

Available in iOS 4.0 and later.

Declared in `CMBlockBuffer.h`.

`kCMBlockBufferBadOffsetParameterErr`
Indicates that the offset provided to an API is out of the range of the relevant CMBlockBuffer.

Available in iOS 4.0 and later.

Declared in `CMBlockBuffer.h`.

`kCMBlockBufferBadLengthParameterErr`
Indicates that the length provided to an API is out of the range of the relevant CMBlockBuffer, or is not allowed to be zero.

Available in iOS 4.0 and later.

Declared in `CMBlockBuffer.h`.

`kCMBlockBufferBadPointerParameterErr`
Indicates that a pointer parameter (for example, a CMBlockBuffer reference, or destination memory) is NULL or otherwise invalid..

Available in iOS 4.0 and later.

Declared in `CMBlockBuffer.h`.

`kCMBlockBufferEmptyBBufErr`
Indicates that an empty CMBlockBuffer was received unexpectedly.

Available in iOS 4.0 and later.

Declared in `CMBlockBuffer.h`.

`kCMBlockBufferUnallocatedBlockErr`
Indicates that an unallocated memory block was encountered.

Available in iOS 4.0 and later.

Declared in `CMBlockBuffer.h`.

Custom Block Source Version

Provides the block source version.

```
enum {
    kCMBlockBufferCustomBlockSourceVersion = 0
};
```

Constants

`kCMBlockBufferCustomBlockSourceVersion`
The value is the block source version.

Available in iOS 4.0 and later.

Declared in `CMBlockBuffer.h`.

CMBlockBuffer Flags

Flags controlling behaviors and features of CMBlockBuffer APIs.

```
enum {
    kCMBlockBufferAssureMemoryNowFlag       = (1L<<0),
    kCMBlockBufferAlwaysCopyDataFlag        = (1L<<1),
    kCMBlockBufferDontOptimizeDepthFlag     = (1L<<2),
    kCMBlockBufferPermitEmptyReferenceFlag  = (1L<<3)
};
```

Constants

`kCMBlockBufferAssureMemoryNowFlag`
When passed to routines that accept block allocators, causes the memory block to be allocated immediately.

Available in iOS 4.0 and later.

Declared in `CMBlockBuffer.h`.

`kCMBlockBufferAlwaysCopyDataFlag`
Used with `CMBlockBufferCreateContiguous` (page 13) to cause it to always produce an allocated copy of the desired data.

Available in iOS 4.0 and later.

Declared in `CMBlockBuffer.h`.

`kCMBlockBufferDontOptimizeDepthFlag`
Passed to `CMBlockBufferAppendBufferReference` (page 10) and `CMBlockBufferCreateWithBufferReference` (page 14) to suppress reference depth optimization.

Available in iOS 4.0 and later.

Declared in `CMBlockBuffer.h`.

`kCMBlockBufferPermitEmptyReferenceFlag`
Passed to `CMBlockBufferAppendBufferReference` (page 10) and `CMBlockBufferCreateWithBufferReference` (page 14) to allow references into a CMBlockBuffer that may not yet be populated.

Available in iOS 4.0 and later.

Declared in `CMBlockBuffer.h`.

CMBufferQueue Reference

Derived From:	*CFType Reference*
Framework:	CoreMedia.framework
Declared in	CMBufferQueue.h

Overview

This document describes the API for creating and manipulating CMBufferQueue structs.

CMBufferQueues are Core Foundation objects that implement a queue of timed buffers. These buffers can be of any CF-based type (CFTypeRef), but must have a concept of duration. During CMBufferQueue creation, a set of callbacks is provided, one of which is a required callback that returns the duration of the CF-based buffer object. A standard callback struct for CMSampleBuffers is provided as a convenience. These callbacks are called synchronously from within various CMBufferQueue APIs, on the thread that called the API.

CMBufferQueues are designed to be read and written from different threads in a producer/consumer model. While this is generally two threads (one producer/enqueuer, one dequeuer/consumer), CMBufferQueues can service any number of threads enqueueing and/or dequeueing buffers. In the CMBufferQueue APIs, all operations (not just `CMBufferQueueEnqueue` (page 28) and `CMBufferQueueDequeueAndRetain` (page 28), but `CMBufferQueueGetDuration` (page 29), `CMBufferQueueInstallTrigger` (page 33) and so on) are made atomic by use of a single mutex (one mutex per created queue object).

By default, a CMBufferQueue is a FIFO queue, but if a comparison callback is provided, the resulting CMBufferQueue will be sorted based on that callback. For example, one might create a CMBufferQueue where the buffers are enqueued in decode order, and dequeued in presentation order, by providing a comparison callback that sorts by presentation timestamp.

CMBufferQueues retain the enqueued buffer during Enqueue, so the client can release the buffer if it has no further need of the reference. During `CMBufferQueueDequeueAndRetain` (page 28), the buffer is retained on behalf of the client, and released by the queue. The result is that the retain count remains the same, and the ownership of the buffer is transferred from the queue to the client.

If provided with a buffer-readiness callback, CMBufferQueues can check for buffer readiness during `CMBufferQueueDequeueIfDataReadyAndRetain` (page 28). If that callback is not provided, all buffers are assumed to be ready, and there is no difference between `CMBufferQueueDequeueAndRetain`, and `CMBufferQueueDequeueIfDataReadyAndRetain`.

CMBufferQueues also implement `CMBufferQueueGetMinDecodeTimeStamp` (page 32) and `CMBufferQueueGetMinPresentationTimeStamp` (page 32), with the help of optional callbacks that get decode and presentation timestamps from a buffer. If either or both of these callbacks is not provided, `kCMTimeInvalid` will be returned for the missing timestamp(s).

CMBufferQueues can be marked with an end-of-data (`CMBufferQueueMarkEndOfData` (page 35)). Once so marked, further enqueues will fail, and once all the buffers have been dequeued, the queue is permanently empty ("at end of data") until Reset is called. Reset empties the queue and undoes the end-of-data marking.

The current status of a CMBufferQueue can be interrogated. You can test for emptiness (`CMBufferQueueIsEmpty` (page 35)), current queue duration (`CMBufferQueueGetDuration` (page 29)), and end-of-data status (`CMBufferQueueContainsEndOfData` (page 27) and `CMBufferQueueIsAtEndOfData` (page 34)).

You can install trigger callbacks (using `CMBufferQueueInstallTrigger` (page 33)) to get notifications of various queue state transitions, such as "duration becomes less than 1 second". The queue cannot be modified during a trigger callback, but it can be interrogated. Trigger conditions can be tested explicitly as well (`CMBufferQueueTestTrigger` (page 37)). Triggers with `NULL` callbacks can be added to a queue for this type of use, but triggers with callbacks can also have their conditions explicitly tested.

Trigger callbacks may be called from any CMBufferQueue API that modifies the total duration of the queue (such as Enqueue/Dequeue/Reset). Trigger callbacks are called synchronously, on the thread that called the API.

Modifying the state of the queue in any way from within a trigger callback is forbidden, and will fail, returning `kCMBufferQueueError_CannotModifyQueueFromTriggerCallback`.

An attempt to Enqueue onto a full queue or to Dequeue from an empty queue will not block, but will return immediately with an error (or with a NULL buffer). Triggers should be installed by the client to manage the client's knowledge of queue fullness. The use of repeated retries (polling) is discouraged as an inefficient use of resources.

Functions

CMBufferQueueCallForEachBuffer

```
OSStatus CMBufferQueueCallForEachBuffer (
   CMBufferQueueRef queue,
   OSStatus (*callback)(CMBufferRef buffer, void *refcon),
   void *refcon
);
```

Parameters

queue

refcon

refcon

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
`CMBufferQueue.h`

CMBufferQueueContainsEndOfData

```
Boolean CMBufferQueueContainsEndOfData (
   CMBufferQueueRef queue
);
```

Parameters

queue

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMBufferQueue.h

CMBufferQueueCreate

```
OSStatus CMBufferQueueCreate (
   CFAllocatorRef allocator,
   CMItemCount capacity,
   const CMBufferCallbacks *callbacks,
   CMBufferQueueRef *queueOut
);
```

Parameters

allocator

capacity

callbacks

queueOut

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMBufferQueue.h

CMBufferQueueDequeueAndRetain

```
CMBufferRef CMBufferQueueDequeueAndRetain (
   CMBufferQueueRef queue
);
```

Parameters

queue

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMBufferQueue.h

CMBufferQueueDequeueIfDataReadyAndRetain

```
CMBufferRef CMBufferQueueDequeueIfDataReadyAndRetain (
   CMBufferQueueRef queue
);
```

Parameters

queue

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMBufferQueue.h

CMBufferQueueEnqueue

```
OSStatus CMBufferQueueEnqueue (
   CMBufferQueueRef queue,
   CMBufferRef buf
);
```

Parameters

queue

buf

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMBufferQueue.h

CMBufferQueueGetBufferCount

```
CMItemCount CMBufferQueueGetBufferCount (
   CMBufferQueueRef queue
);
```

Parameters

queue

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMBufferQueue.h

CMBufferQueueGetCallbacksForUnsortedSampleBuffers

```
const CMBufferCallbacks * CMBufferQueueGetCallbacksForUnsortedSampleBuffers (
   void
);
```

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMBufferQueue.h

CMBufferQueueGetDuration

```
CMTime CMBufferQueueGetDuration (
   CMBufferQueueRef queue
);
```

Parameters

queue

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMBufferQueue.h

CMBufferQueueGetEndPresentationTimeStamp

```
CMTime CMBufferQueueGetEndPresentationTimeStamp (
   CMBufferQueueRef queue
);
```

Parameters

queue

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMBufferQueue.h

CMBufferQueueGetFirstDecodeTimeStamp

```
CMTime CMBufferQueueGetFirstDecodeTimeStamp (
   CMBufferQueueRef queue
);
```

Parameters

queue

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMBufferQueue.h

CMBufferQueueGetFirstPresentationTimeStamp

```
CMTime CMBufferQueueGetFirstPresentationTimeStamp (
   CMBufferQueueRef queue
);
```

Parameters

queue

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMBufferQueue.h

CMBufferQueueGetHead

```
CMBufferRef CMBufferQueueGetHead (
   CMBufferQueueRef queue
);
```

Parameters

queue

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMBufferQueue.h

CMBufferQueueGetMaxPresentationTimeStamp

```
CMTime CMBufferQueueGetMaxPresentationTimeStamp (
   CMBufferQueueRef queue
);
```

Parameters

queue

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMBufferQueue.h

CMBufferQueueGetMinDecodeTimeStamp

```
CMTime CMBufferQueueGetMinDecodeTimeStamp (
   CMBufferQueueRef queue
);
```

Parameters

queue

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMBufferQueue.h

CMBufferQueueGetMinPresentationTimeStamp

```
CMTime CMBufferQueueGetMinPresentationTimeStamp (
   CMBufferQueueRef queue
);
```

Parameters

queue

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMBufferQueue.h

CMBufferQueueGetTypeID

```
CFTypeID CMBufferQueueGetTypeID (
   void
);
```

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMBufferQueue.h

CMBufferQueueInstallTrigger

```
OSStatus CMBufferQueueInstallTrigger (
   CMBufferQueueRef queue,
   CMBufferQueueTriggerCallback triggerCallback,
   void *triggerRefcon,
   CMBufferQueueTriggerCondition triggerCondition,
   CMTime triggerTime,
   CMBufferQueueTriggerToken *triggerTokenOut
);
```

Parameters

queue

triggerCallback

triggerRefcon

triggerCondition

triggerTime

triggerTokenOut

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMBufferQueue.h

CMBufferQueueInstallTriggerWithIntegerThreshold

```
OSStatus CMBufferQueueInstallTriggerWithIntegerThreshold (
   CMBufferQueueRef queue,
   CMBufferQueueTriggerCallback triggerCallback,
   void *triggerRefcon,
   CMBufferQueueTriggerCondition triggerCondition,
   CMItemCount triggerThreshold,
   CMBufferQueueTriggerToken *triggerTokenOut
);
```

Parameters

queue

triggerCallback

triggerRefcon

triggerCondition

triggerThreshold

triggerTokenOut

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMBufferQueue.h

CMBufferQueueIsAtEndOfData

```
Boolean CMBufferQueueIsAtEndOfData (
   CMBufferQueueRef queue
);
```

Parameters

queue

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMBufferQueue.h

CMBufferQueueIsEmpty

```
Boolean CMBufferQueueIsEmpty (
   CMBufferQueueRef queue
);
```

Parameters

queue

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMBufferQueue.h

CMBufferQueueMarkEndOfData

```
OSStatus CMBufferQueueMarkEndOfData (
   CMBufferQueueRef queue
);
```

Parameters

queue

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMBufferQueue.h

CMBufferQueueRemoveTrigger

```
OSStatus CMBufferQueueRemoveTrigger (
   CMBufferQueueRef queue,
   CMBufferQueueTriggerToken triggerToken
);
```

Parameters

queue

triggerToken

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMBufferQueue.h

CMBufferQueueReset

```
OSStatus CMBufferQueueReset (
   CMBufferQueueRef queue
);
```

Parameters

queue

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMBufferQueue.h

CMBufferQueueResetWithCallback

```
OSStatus CMBufferQueueResetWithCallback (
   CMBufferQueueRef queue,
   void (*callback)(CMBufferRef buffer, void *refcon),
   void *refcon
);
```

Parameters

queue

refcon

refcon

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMBufferQueue.h

CMBufferQueueSetValidationCallback

```
OSStatus CMBufferQueueSetValidationCallback (
   CMBufferQueueRef queue,
   CMBufferValidationCallback validationCallback,
   void *validationRefCon
);
```

Parameters

queue

validationCallback

validationRefCon

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMBufferQueue.h

CMBufferQueueTestTrigger

```
Boolean CMBufferQueueTestTrigger (
   CMBufferQueueRef queue,
   CMBufferQueueTriggerToken triggerToken
);
```

Parameters

queue

triggerToken

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMBufferQueue.h

Data Types

Opaque Types

CMBufferRef

A reference to a CMBuffer object.

```
typedef CFTypeRef CMBufferRef;
```

Discussion
A CMBuffer can be an instance of any Core Foundation type, as long as a getDuration callback can be provided. Commonly-used types are CMSampleBuffer and CVPixelBuffer.

Availability
Available in iOS 4.0 and later.

Declared In
`CMBufferQueue.h`

CMBufferQueueRef

A reference to a CMBufferQueueRef object.

```
typedef struct opaqueCMBufferQueue *CMBufferQueueRef;
```

Discussion
A CMBufferQueue is a Core Foundation object that implements a queue of timed buffers.

Availability
Available in iOS 4.0 and later.

Declared In
`CMBufferQueue.h`

Callbacks

CMBufferCallbacks

Callbacks provided to `CMBufferQueueCreate` (page 27), for use by the queue in interrogating the buffers that it will see.

```
typedef struct {
    uint32_t                    version;
    void                        *refcon;
    CMBufferGetTimeCallback     getDecodeTimeStamp;
    CMBufferGetTimeCallback     getPresentationTimeStamp;
    CMBufferGetTimeCallback     getDuration;
```

```
    CMBufferGetBooleanCallback   isDataReady;
    CMBufferCompareCallback      compare;
    CFStringRef                  dataBecameReadyNotification;
} CMBufferCallbacks;
```

Fields

`version`

The callback version.

This must be `0`.

`refcon`

Contextual data to be passed to all callbacks

This can be `NULL`, if the callbacks don't require it.

`getDecodeTimeStamp`

This callback is called from `CMBufferQueueGetFirstDecodeTimeStamp` (page 30) (once), and from `CMBufferQueueGetMinDecodeTimeStamp` (page 32) (multiple times).

It should return the decode timestamp of the buffer. If there are multiple samples in the buffer, this callback should return the minimum decode timestamp in the buffer.

This can be `NULL` (`CMBufferQueueGetFirstDecodeTimeStamp` and `CMBufferQueueGetMinDecodeTimeStamp` will return `kCMTimeInvalid`).

`getPresentationTimeStamp`

This callback is called from `CMBufferQueueGetFirstPresentationTimeStamp` (page 31) (once), and from `CMBufferQueueGetMinPresentationTimeStamp` (page 32) (multiple times).

It should return the presentation timestamp of the buffer. If there are multiple samples in the buffer, this callback should return the minimum presentation timestamp in the buffer.

This can be `NULL` (`CMBufferQueueGetFirstPresentationTimeStamp` and `CMBufferQueueGetMinPresentationTimeStamp` will return `kCMTimeInvalid`).

`getDuration`

This callback is called (once) during enqueue and dequeue operations to update the total duration of the queue.

This must not be `NULL`.

`isDataReady`

This callback is called from `CMBufferQueueDequeueIfDataReadyAndRetain` (page 28), to ask if the buffer that is about to be dequeued is ready.

This may be `NULL` (data will be assumed to be ready).

`compare`

This callback is called (multiple times) from `CMBufferQueueEnqueue` (page 28), to perform an insertion sort.

This may be `NULL` (queue will be FIFO).

`dataBecameReadyNotification`

If triggers of type `kCMBufferQueueTrigger_WhenDataBecomesReady` are installed, the queue will listen for this notification on the head buffer.

This may be `NULL` (then the queue won't listen for it).

Discussion

With the exception of isDataReady, all these callbacks must always return the same result for the same arguments.

A buffer's duration, timestamps, or position relative to other buffers must not appear to change while it is in the queue. Once isDataReady has returned true for a given CMBuffer, it must always return true for that CMBuffer.

Durations must always be positive.

Availability
Available in iOS 4.0 and later.

Declared In
`CMBufferQueue.h`

CMBufferValidationCallback

Tests whether a buffer is in a valid state to add to a queue.

```
typedef OSStatus (*CMBufferValidationCallback)( CMBufferQueueRef queue, CMBufferRef
 buf, void *validationRefCon );
```

Fields

`queue`
The queue requesting validation.

`buf`
The buffer about to be added.

`validationRefCon`
Contextual data.

Discussion
`CMBufferQueueEnqueue` (page 28) will call this function to validate buffers.

Return `noErr` if the buffer is in a valid state to add.

Return a nonzero error code if the buffer should be rejected; `CMBufferQueueEnqueue` will return this error to the caller. If you do not have a more descriptive error code, use `kCMBufferQueueError_InvalidBuffer`.

Triggers

A trigger is a callback function that a queue calls every time the triggering condition becomes true. Trigger conditions include things like queue duration, queue buffer count, and so on. Trigger callbacks are called from within CMBufferQueue routines that modify the trigger condition (for example, Enqueue/Dequeue/Reset).

Trigger callbacks cannot modify the queue that called them; they can, however, interrogate it. Trigger callbacks should perform as little processing as possible, preferably arranging for processing to occur by, for example, signaling a semaphore, or rescheduling a runloop timer.

You can install as many triggers as you like. The order in which they are called is non-deterministic.

Triggers with a `NULL` callback are valid, since even though no trigger callback will be called, the trigger condition can still be explicitly tested.

CMBufferQueueTriggerToken

A reference to a CMBufferQueueTriggerToken object.

```
typedef struct opaqueCMBufferQueueTriggerToken *CMBufferQueueTriggerToken;
```

Discussion

The CMBufferQueueTriggerToken is returned from `CMBufferQueueInstallTrigger` (page 33), so you can remove it later if necessary. Triggers will automatically be removed when the queue is finalized. Note that if more than one module has access to a queue, it may be hard for an individual module to know when the queue is finalized since other modules may retain it. To address this concern, modules should remove their triggers before they themselves are finalized.

Special Considerations

A CMBufferQueueTrigger is not a Core Foundation object; you must not `CFRetain` or `CFRelease` it.

Availability

Available in iOS 4.0 and later.

Declared In

`CMBufferQueue.h`

CMBufferQueueTriggerCallback

A callback to be called when a CMBufferQueue trigger condition becomes true.

```
typedef void (*CMBufferQueueTriggerCallback) (
    void *triggerRefcon,
    CMBufferQueueTriggerToken triggerToken
);
```

Fields

`triggerRefcon`
The contextual data.

`triggerToken`
The trigger whose condition became true.

Discussion

CMBufferQueueTriggerCondition

A type to specify conditions to be associated with a CMBufferQueueTrigger.

```
typedef int32_t CMBufferQueueTriggerCondition;
```

Discussion

For possible values, see "Trigger Conditions" (page 43).

Availability

Available in iOS 4.0 and later.

Declared In

`CMBufferQueue.h`

Constants

Error Codes

Error codes returned by CMBufferQueue APIs.

```
enum {
    kCMBufferQueueError_AllocationFailed                        = -12760,
    kCMBufferQueueError_RequiredParameterMissing                = -12761,
    kCMBufferQueueError_InvalidCMBufferCallbacksStruct          = -12762,
    kCMBufferQueueError_EnqueueAfterEndOfData                   = -12763,
    kCMBufferQueueError_QueueIsFull                             = -12764,
    kCMBufferQueueError_BadTriggerDuration                      = -12765,
    kCMBufferQueueError_CannotModifyQueueFromTriggerCallback    = -12766,
    kCMBufferQueueError_InvalidTriggerCondition                 = -12767,
    kCMBufferQueueError_InvalidTriggerToken                     = -12768,
    kCMBufferQueueError_InvalidBuffer                           = -12769,
};
```

Constants

`kCMBufferQueueError_AllocationFailed`

Indicates that allocation failed.

Available in iOS 4.0 and later.

Declared in `CMBufferQueue.h`.

`kCMBufferQueueError_RequiredParameterMissing`

Indicates NULL or 0 was passed for a required parameter.

Available in iOS 4.0 and later.

Declared in `CMBufferQueue.h`.

`kCMBufferQueueError_InvalidCMBufferCallbacksStruct`

Indicates that Version was not 0, or getDuration was NULL.

Available in iOS 4.0 and later.

Declared in `CMBufferQueue.h`.

`kCMBufferQueueError_EnqueueAfterEndOfData`

Indicates that CMBufferQueueEnqueue was attempted after CMBufferQueueMarkEndOfData was called (without a call to CMBufferQueueReset in between).

Available in iOS 4.0 and later.

Declared in `CMBufferQueue.h`.

`kCMBufferQueueError_QueueIsFull`

Indicates that CMBufferQueueEnqueue was attempted on a full queue.

Available in iOS 4.0 and later.

Declared in `CMBufferQueue.h`.

`kCMBufferQueueError_BadTriggerDuration`
Indicates that the trigger duration was invalid.

Trigger duration must be numeric, and epoch must be zero (non-zero epoch is only for timestamps, not durations).

Available in iOS 4.0 and later.

Declared in `CMBufferQueue.h`.

`kCMBufferQueueError_CannotModifyQueueFromTriggerCallback`
Indicates that an attempt was made to modify the queue from a trigger callback.

Available in iOS 4.0 and later.

Declared in `CMBufferQueue.h`.

`kCMBufferQueueError_InvalidTriggerCondition`
Indicates that a trigger condition is not a value from the CMBufferQueueTriggerCondition enum, or the trigger condition is not supported by a buffer queue.

Available in iOS 4.0 and later.

Declared in `CMBufferQueue.h`.

`kCMBufferQueueError_InvalidTriggerToken`
Indicates that the trigger token is not a trigger that is currently associated with this queue.

Available in iOS 4.0 and later.

Declared in `CMBufferQueue.h`.

`kCMBufferQueueError_InvalidBuffer`
Indicates that a buffer was rejected by the CMBufferValidationCallback.

Available in iOS 4.0 and later.

Declared in `CMBufferQueue.h`.

Trigger Conditions

Conditions to be associated with a CMBufferQueueTrigger.

```
enum {
    kCMBufferQueueTrigger_WhenDurationBecomesLessThan = 1,
    kCMBufferQueueTrigger_WhenDurationBecomesLessThanOrEqualTo = 2,
    kCMBufferQueueTrigger_WhenDurationBecomesGreaterThan = 3,
    kCMBufferQueueTrigger_WhenDurationBecomesGreaterThanOrEqualTo = 4,
    kCMBufferQueueTrigger_WhenMinPresentationTimeStampChanges = 5,
    kCMBufferQueueTrigger_WhenMaxPresentationTimeStampChanges = 6,
    kCMBufferQueueTrigger_WhenDataBecomesReady = 7,
    kCMBufferQueueTrigger_WhenEndOfDataReached = 8,
    kCMBufferQueueTrigger_WhenReset = 9,
    kCMBufferQueueTrigger_WhenBufferCountBecomesLessThan = 10,
    kCMBufferQueueTrigger_WhenBufferCountBecomesGreaterThan = 11,
};
```

Constants

`kCMBufferQueueTrigger_WhenDurationBecomesLessThan`
Trigger fires when queue duration becomes less than the specified duration.

Available in iOS 4.0 and later.

Declared in `CMBufferQueue.h`.

`kCMBufferQueueTrigger_WhenDurationBecomesLessThanOrEqualTo`
Trigger fires when queue duration becomes less than or equal to the specified duration.

Available in iOS 4.0 and later.

Declared in `CMBufferQueue.h`.

`kCMBufferQueueTrigger_WhenDurationBecomesGreaterThan`
Trigger fires when queue duration becomes greater than the specified duration.

Available in iOS 4.0 and later.

Declared in `CMBufferQueue.h`.

`kCMBufferQueueTrigger_WhenDurationBecomesGreaterThanOrEqualTo`
Trigger fires when queue duration becomes greater than or equal to the specified duration.

Available in iOS 4.0 and later.

Declared in `CMBufferQueue.h`.

`kCMBufferQueueTrigger_WhenMinPresentationTimeStampChanges`
Trigger fires when the minimum presentation timestamp changes (triggerDuration is ignored).

Available in iOS 4.0 and later.

Declared in `CMBufferQueue.h`.

`kCMBufferQueueTrigger_WhenMaxPresentationTimeStampChanges`
Trigger fires when the maximum presentation timestamp changes (triggerDuration is ignored).

Available in iOS 4.0 and later.

Declared in `CMBufferQueue.h`.

`kCMBufferQueueTrigger_WhenDataBecomesReady`
Trigger fires when next dequeueable buffer becomes ready (that is, `CMBufferQueueDequeueIfDataReadyAndRetain` (page 28) will now succeed). (triggerDuration is ignored.)

Available in iOS 4.0 and later.

Declared in `CMBufferQueue.h`.

`kCMBufferQueueTrigger_WhenEndOfDataReached`
Trigger fires when CMBufferQueueIsAtEndOfData's condition becomes true. (triggerDuration is ignored.)

Available in iOS 4.0 and later.

Declared in `CMBufferQueue.h`.

`kCMBufferQueueTrigger_WhenReset`
Trigger fires when CMBufferQueueReset called. (triggerDuration is ignored.)

Available in iOS 4.0 and later.

Declared in `CMBufferQueue.h`.

`kCMBufferQueueTrigger_WhenBufferCountBecomesLessThan`
Trigger fires when buffer count becomes less than the specified threshold number.

Available in iOS 4.0 and later.

Declared in `CMBufferQueue.h`.

`kCMBufferQueueTrigger_WhenBufferCountBecomesGreaterThan`
Trigger fires when buffer count becomes > the specified threshold number.

Available in iOS 4.0 and later.

Declared in `CMBufferQueue.h`.

CMFormatDescription Reference

Derived From:	*CFType Reference*
Framework:	CoreMedia.framework
Declared in	CMFormatDescription.h

Overview

This document describes the API for creating and manipulating CMFormatDescription structs.

CMFormatDescriptions are immutable Core Foundation objects that describe media data of various types, including audio, video, and muxed media data. There are two types of API: media-type-agnostic APIs (supported by all CMFormatDescriptions) and media-type-specific APIs. The media-type-agnostic APIs are prefixed with CMFormatDescription, and the media-type-specific APIs are prefixed with CMAudioFormatDescription CMVideoFormatDescription, and so on.

Functions by Task

Media-type-Agnostic Functions

Audio-Specific Functions

Video-Specific Functions

Muxed-Specific Function

Metadata-Specific Functions

Text-Specific Functions

Time-code-Specific Functions

Functions

CMAudioFormatDescriptionCreate

```
OSStatus CMAudioFormatDescriptionCreate (
   CFAllocatorRef allocator,
   const AudioStreamBasicDescription *asbd,
   size_t layoutSize,
   const AudioChannelLayout *layout,
   size_t magicCookieSize,
   const void *magicCookie,
   CFDictionaryRef extensions,
   CMAudioFormatDescriptionRef *outDesc
);
```

Parameters

allocator

asbd

layoutSize

layout

magicCookieSize

magicCookie

extensions

outDesc

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMFormatDescription.h

CMAudioFormatDescriptionEqual

```
Boolean CMAudioFormatDescriptionEqual (
   CMAudioFormatDescriptionRef desc1,
   CMAudioFormatDescriptionRef desc2,
   CMAudioFormatDescriptionMask equalityMask,
   CMAudioFormatDescriptionMask *equalityMaskOut
);
```

Parameters

desc1

desc2

equalityMask

equalityMaskOut

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMFormatDescription.h

CMAudioFormatDescriptionGetChannelLayout

```
const AudioChannelLayout * CMAudioFormatDescriptionGetChannelLayout (
   CMAudioFormatDescriptionRef desc,
   size_t *layoutSize
);
```

Parameters

desc

layoutSize

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMFormatDescription.h

CMAudioFormatDescriptionGetFormatList

```
const AudioFormatListItem * CMAudioFormatDescriptionGetFormatList (
   CMAudioFormatDescriptionRef desc,
   size_t *formatListSize
);
```

Parameters

desc

formatListSize

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMFormatDescription.h

CMAudioFormatDescriptionGetMagicCookie

```
const void * CMAudioFormatDescriptionGetMagicCookie (
   CMAudioFormatDescriptionRef desc,
   size_t *cookieSizeOut
);
```

Parameters

desc

cookieSizeOut

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMFormatDescription.h

CMAudioFormatDescriptionGetMostCompatibleFormat

```
const AudioFormatListItem * CMAudioFormatDescriptionGetMostCompatibleFormat (
   CMAudioFormatDescriptionRef desc
);
```

Parameters

desc

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMFormatDescription.h

CMAudioFormatDescriptionGetRichestDecodableFormat

```
const AudioFormatListItem * CMAudioFormatDescriptionGetRichestDecodableFormat (
   CMAudioFormatDescriptionRef desc
);
```

Parameters

desc

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMFormatDescription.h

CMAudioFormatDescriptionGetStreamBasicDescription

```
const AudioStreamBasicDescription * CMAudioFormatDescriptionGetStreamBasicDescription
 (
   CMAudioFormatDescriptionRef desc
);
```

Parameters

desc

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMFormatDescription.h

CMFormatDescriptionCreate

```
OSStatus CMFormatDescriptionCreate (
   CFAllocatorRef allocator,
   CMMediaType mediaType,
   FourCharCode mediaSubtype,
   CFDictionaryRef extensions,
   CMFormatDescriptionRef *descOut
);
```

Parameters

allocator

mediaType

mediaSubtype

extensions

descOut

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMFormatDescription.h

CMFormatDescriptionEqual

```
Boolean CMFormatDescriptionEqual (
   CMFormatDescriptionRef ffd1,
   CMFormatDescriptionRef ffd2
);
```

Parameters

ffd1

ffd2

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMFormatDescription.h

CMFormatDescriptionGetExtension

```
CFPropertyListRef CMFormatDescriptionGetExtension (
   CMFormatDescriptionRef desc,
   CFStringRef extensionKey
);
```

Parameters

desc

extensionKey

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMFormatDescription.h

CMFormatDescriptionGetExtensions

```
CFDictionaryRef CMFormatDescriptionGetExtensions (
   CMFormatDescriptionRef desc
);
```

Parameters

desc

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMFormatDescription.h

CMFormatDescriptionGetMediaSubType

```
FourCharCode CMFormatDescriptionGetMediaSubType (
   CMFormatDescriptionRef desc
);
```

Parameters

desc

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMFormatDescription.h

CMFormatDescriptionGetMediaType

```
CMMediaType CMFormatDescriptionGetMediaType (
   CMFormatDescriptionRef desc
);
```

Parameters
desc

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMFormatDescription.h

CMFormatDescriptionGetTypeID

```
CFTypeID CMFormatDescriptionGetTypeID (
   void
);
```

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMFormatDescription.h

CMMetadataFormatDescriptionCreateWithKeys

```
OSStatus CMMetadataFormatDescriptionCreateWithKeys (
   CFAllocatorRef allocator,
   CMMetadataFormatType metadataType,
   CFArrayRef keys,
   CMMetadataFormatDescriptionRef *outDesc
);
```

Parameters

allocator

metadataType

keys

outDesc

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMFormatDescription.h

CMMetadataFormatDescriptionGetKeyWithLocalID

```
CFDictionaryRef CMMetadataFormatDescriptionGetKeyWithLocalID (
   CMMetadataFormatDescriptionRef desc,
   OSType localKeyID
);
```

Parameters

desc

localKeyID

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMFormatDescription.h

CMMuxedFormatDescriptionCreate

```
OSStatus CMMuxedFormatDescriptionCreate (
   CFAllocatorRef allocator,
   CMMuxedStreamType muxType,
   CFDictionaryRef extensions,
   CMMuxedFormatDescriptionRef *outDesc
);
```

Parameters

allocator

muxType

extensions

outDesc

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMFormatDescription.h

CMTextFormatDescriptionGetDefaultStyle

```
OSStatus CMTextFormatDescriptionGetDefaultStyle (
   CMFormatDescriptionRef desc,
   uint16_t *outLocalFontID,
   Boolean *outBold,
   Boolean *outItalic,
   Boolean *outUnderline,
   CGFloat *outFontSize,
   CGFloat outColorComponents[4]
);
```

Parameters

desc

outLocalFontID

outBold

outItalic

outUnderline

outFontSize

CGFloat outColorComponents[4]

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMFormatDescription.h

CMTextFormatDescriptionGetDefaultTextBox

```
OSStatus CMTextFormatDescriptionGetDefaultTextBox (
   CMFormatDescriptionRef desc,
   Boolean originIsAtTopLeft,
   CGFloat heightOfTextTrack,
   CGRect *outDefaultTextBox
);
```

Parameters

desc

originIsAtTopLeft

heightOfTextTrack

outDefaultTextBox

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMFormatDescription.h

CMTextFormatDescriptionGetDisplayFlags

```
OSStatus CMTextFormatDescriptionGetDisplayFlags (
   CMFormatDescriptionRef desc,
   CMTextDisplayFlags *outDisplayFlags
);
```

Parameters

desc

outDisplayFlags

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMFormatDescription.h

CMTextFormatDescriptionGetFontName

```
OSStatus CMTextFormatDescriptionGetFontName (
   CMFormatDescriptionRef desc,
   uint16_t localFontID,
   CFStringRef *outFontName
);
```

Parameters

desc

localFontID

outFontName

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMFormatDescription.h

CMTextFormatDescriptionGetJustification

```
OSStatus CMTextFormatDescriptionGetJustification (
   CMFormatDescriptionRef desc,
   CMTextJustificationValue *outHorizontalJust,
   CMTextJustificationValue *outVerticalJust
);
```

Parameters

desc

outHorizontalJust

outVerticalJust

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMFormatDescription.h

CMTimeCodeFormatDescriptionCreate

```
OSStatus CMTimeCodeFormatDescriptionCreate (
   CFAllocatorRef allocator,
   CMTimeCodeFormatType timeCodeFormatType,
   CMTime frameDuration,
   uint32_t frameQuanta,
   uint32_t tcFlags,
   CFDictionaryRef extensions,
   CMTimeCodeFormatDescriptionRef *descOut
);
```

Parameters

allocator

timeCodeFormatType

frameDuration

frameQuanta

tcFlags

For possible values, see "CMTimeCodeFormatType" (page 68).

extensions

descOut

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMFormatDescription.h

CMTimeCodeFormatDescriptionGetFrameDuration

```
CMTime CMTimeCodeFormatDescriptionGetFrameDuration (
   CMTimeCodeFormatDescriptionRef timeCodeFormatDescription
);
```

Parameters

timeCodeFormatDescription

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMFormatDescription.h

CMTimeCodeFormatDescriptionGetFrameQuanta

```
uint32_t CMTimeCodeFormatDescriptionGetFrameQuanta (
   CMTimeCodeFormatDescriptionRef timeCodeFormatDescription
);
```

Parameters

timeCodeFormatDescription

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMFormatDescription.h

CMTimeCodeFormatDescriptionGetTimeCodeFlags

```
uint32_t CMTimeCodeFormatDescriptionGetTimeCodeFlags (
   CMTimeCodeFormatDescriptionRef desc
);
```

Parameters

desc

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMFormatDescription.h

CMVideoFormatDescriptionCreate

```
OSStatus CMVideoFormatDescriptionCreate (
   CFAllocatorRef allocator,
   CMVideoCodecType codecType,
   int32_t width,
   int32_t height,
   CFDictionaryRef extensions,
   CMVideoFormatDescriptionRef *outDesc
);
```

Parameters

allocator

codecType

width

height

extensions

outDesc

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMFormatDescription.h

CMVideoFormatDescriptionCreateForImageBuffer

```
OSStatus CMVideoFormatDescriptionCreateForImageBuffer (
   CFAllocatorRef allocator,
   CVImageBufferRef imageBuffer,
   CMVideoFormatDescriptionRef *outDesc
);
```

Parameters

allocator

imageBuffer

outDesc

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMFormatDescription.h

CMVideoFormatDescriptionGetCleanAperture

```
CGRect CMVideoFormatDescriptionGetCleanAperture (
   CMVideoFormatDescriptionRef videoDesc,
   Boolean originIsAtTopLeft
);
```

Parameters

videoDesc

originIsAtTopLeft

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMFormatDescription.h

CMVideoFormatDescriptionGetDimensions

```
CMVideoDimensions CMVideoFormatDescriptionGetDimensions (
   CMVideoFormatDescriptionRef videoDesc
);
```

Parameters

videoDesc

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMFormatDescription.h

CMVideoFormatDescriptionGetExtensionKeysCommonWithImageBuffers

```
CFArrayRef CMVideoFormatDescriptionGetExtensionKeysCommonWithImageBuffers (
   void
);
```

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMFormatDescription.h

CMVideoFormatDescriptionGetPresentationDimensions

```
CGSize CMVideoFormatDescriptionGetPresentationDimensions (
   CMVideoFormatDescriptionRef videoDesc,
   Boolean usePixelAspectRatio,
   Boolean useCleanAperture
);
```

Parameters

videoDesc

usePixelAspectRatio

useCleanAperture

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMFormatDescription.h

CMVideoFormatDescriptionMatchesImageBuffer

```
Boolean CMVideoFormatDescriptionMatchesImageBuffer (
   CMVideoFormatDescriptionRef desc,
   CVImageBufferRef imageBuffer
);
```

Parameters

desc

imageBuffer

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMFormatDescription.h

Data Types

Format Description Types

CMFormatDescriptionRef

A reference to a CMFormatDescription object.

```
typedef struct opaqueCMFormatDescription *CMFormatDescriptionRef;
```

Discussion
A CMFormatDescription object is a Core Foundation object describing media of a particular type (audio, video, muxed, and so on).

Availability
Available in iOS 4.0 and later.

Declared In
`CMFormatDescription.h`

CMAudioFormatDescriptionRef

A synonym type used for manipulating audio CMFormatDescriptions.

```
typedef CMFormatDescriptionRef CMAudioFormatDescriptionRef;
```

Availability
Available in iOS 4.0 and later.

Declared In
`CMFormatDescription.h`

CMVideoFormatDescriptionRef

A synonym type used for manipulating video CMFormatDescriptions.

```
typedef CMFormatDescriptionRef CMVideoFormatDescriptionRef;
```

Availability
Available in iOS 4.0 and later.

Declared In
`CMFormatDescription.h`

CMMuxedFormatDescriptionRef

A synonym type used for manipulating muxed media CMFormatDescriptions.

```
typedef CMFormatDescriptionRef CMMuxedFormatDescriptionRef;
```

Availability
Available in iOS 4.0 and later.

Declared In
`CMFormatDescription.h`

CMClosedCaptionFormatDescriptionRef

A synonym type used for manipulating closed-caption media CMFormatDescriptions.

```
typedef CMFormatDescriptionRef CMClosedCaptionFormatDescriptionRef;
```

Availability
Available in iOS 4.0 and later.

Declared In
`CMFormatDescription.h`

CMTimeCodeFormatDescriptionRef

A synonym type used for manipulating TimeCode CMFormatDescriptions.

```
typedef CMFormatDescriptionRef CMTimeCodeFormatDescriptionRef;
```

Availability
Available in iOS 4.0 and later.

Declared In
`CMFormatDescription.h`

CMMetadataFormatDescriptionRef

A synonym type used for manipulating metadata CMFormatDescriptions.

```
typedef CMFormatDescriptionRef CMMetadataFormatDescriptionRef;
```

Availability
Available in iOS 4.0 and later.

Declared In
`CMFormatDescription.h`

Media Types

CMAudioFormatDescriptionMask

A type for mask bits passed to (and returned from) `CMAudioFormatDescriptionEqual` (page 49), representing various parts of an audio format description.

```
typedef uint32_t CMAudioFormatDescriptionMask;
```

Availability
Available in iOS 4.0 and later.

Declared In
`CMFormatDescription.h`

CMVideoDimensions

Type used for video dimensions.

```
typedef struct {
    int32_t width;
    int32_t height;
} CMVideoDimensions;
```

Fields

`width`
The width of the video.

`height`
The height of the video.

Discussion
The units are pixels.

Availability
Available in iOS 4.0 and later.

Declared In
`CMFormatDescription.h`

Constants

CMMediaType

The type of media described by a CMFormatDescription.

```
enum {
    kCMMediaType_Video              = 'vide',
    kCMMediaType_Audio              = 'soun',
    kCMMediaType_Muxed              = 'muxx',
    kCMMediaType_Text               = 'text',
    kCMMediaType_ClosedCaption      = 'clcp',
    kCMMediaType_Subtitle           = 'sbtl',
    kCMMediaType_TimeCode           = 'tmcd',
    kCMMediaType_TimedMetadata      = 'tmet'
};
typedef FourCharCode CMMediaType;
```

Constants

`kCMMediaType_Video`
Video media.
Available in iOS 4.0 and later.
Declared in `CMFormatDescription.h`.

`kCMMediaType_Audio`
Audio media.
Available in iOS 4.0 and later.
Declared in `CMFormatDescription.h`.

`kCMMediaType_Muxed`
Muxed media.
Available in iOS 4.0 and later.
Declared in `CMFormatDescription.h`.

`kCMMediaType_Text`
Text media.
Available in iOS 4.0 and later.
Declared in `CMFormatDescription.h`.

`kCMMediaType_ClosedCaption`
Closed-caption media.
Available in iOS 4.0 and later.
Declared in `CMFormatDescription.h`.

`kCMMediaType_Subtitle`
Subtitle media.
Available in iOS 4.0 and later.
Declared in `CMFormatDescription.h`.

`kCMMediaType_TimeCode`
Time code media.
Available in iOS 4.0 and later.
Declared in `CMFormatDescription.h`.

`kCMMediaType_TimedMetadata`
Timed meta data.
Available in iOS 4.0 and later.
Declared in `CMFormatDescription.h`.

Error Codes

OSStatus errors returned by CMFormatDescription APIs.

```
enum {
    kCMFormatDescriptionError_InvalidParameter        = -12710,
    kCMFormatDescriptionError_AllocationFailed        = -12711,
};
```

Constants

`kCMFormatDescriptionError_InvalidParameter`

Indicates that a parameter is valid.

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

`kCMFormatDescriptionError_AllocationFailed`

Returned when an allocation fails.

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

CMTimeCodeFormatType

The types of Time Code.

```
enum {
    kCMTimeCodeFormatType_TimeCode32    = 'tmcd',
    kCMTimeCodeFormatType_TimeCode64    = 'tc64',
    kCMTimeCodeFormatType_Counter32     = 'cn32',
    kCMTimeCodeFormatType_Counter64     = 'cn64'
};
typedef FourCharCode CMTimeCodeFormatType;
```

Constants

`kCMTimeCodeFormatType_TimeCode32`

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

`kCMTimeCodeFormatType_TimeCode64`

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

`kCMTimeCodeFormatType_Counter32`

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

`kCMTimeCodeFormatType_Counter64`

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

Time Code Flags

Flags passed to `CMFormatDescriptionCreate` (page 52).

```
enum {
    kCMTimeCodeFlag_DropFrame    = 1 << 0,
    kCMTimeCodeFlag_24HourMax    = 1 << 1,
    kCMTimeCodeFlag_NegTimesOK    = 1 << 2
};
```

Constants

`kCMTimeCodeFlag_DropFrame`

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

`kCMTimeCodeFlag_24HourMax`

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

`kCMTimeCodeFlag_NegTimesOK`

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

CMTextDisplayFlags

Display mode flags for text media.

```
enum {
    kCMTextDisplayFlag_scrollIn                    = 0x00000020,
    kCMTextDisplayFlag_scrollOut                   = 0x00000040,
    kCMTextDisplayFlag_scrollDirectionMask         = 0x00000180,
        kCMTextDisplayFlag_scrollDirection_bottomToTop = 0x00000000,
        kCMTextDisplayFlag_scrollDirection_rightToLeft = 0x00000080,
        kCMTextDisplayFlag_scrollDirection_topToBottom = 0x00000100,
        kCMTextDisplayFlag_scrollDirection_leftToRight = 0x00000180,
    kCMTextDisplayFlag_continuousKaraoke           = 0x00000800,
    kCMTextDisplayFlag_writeTextVertically         = 0x00020000,
    kCMTextDisplayFlag_fillTextRegion              = 0x00040000,
    kCMTextDisplayFlag_forcedSubtitlesPresent      = 0x40000000,
    kCMTextDisplayFlag_allSubtitlesForced          = 0x80000000,
};
typedef uint32_t CMTextDisplayFlags;
```

Constants

`kCMTextDisplayFlag_scrollIn`

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

`kCMTextDisplayFlag_scrollOut`

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

`kCMTextDisplayFlag_scrollDirectionMask`

The scrolling direction is set by a two-bit field, obtained from displayFlags using kCMTextDisplayFlag_scrollDirectionMask.

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

`kCMTextDisplayFlag_scrollDirection_bottomToTop`
Text is vertically scrolled up ("credits style"), entering from the bottom and leaving towards the top.

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

`kCMTextDisplayFlag_scrollDirection_rightToLeft`
Text is horizontally scrolled ("marquee style"), entering from the right and leaving towards the left.

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

`kCMTextDisplayFlag_scrollDirection_topToBottom`
Text is vertically scrolled down, entering from the top and leaving towards the bottom.

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

`kCMTextDisplayFlag_scrollDirection_leftToRight`
Text is horizontally scrolled, entering from the left and leaving towards the right.

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

`kCMTextDisplayFlag_continuousKaraoke`

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

`kCMTextDisplayFlag_writeTextVertically`

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

`kCMTextDisplayFlag_fillTextRegion`

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

`kCMTextDisplayFlag_forcedSubtitlesPresent`

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

`kCMTextDisplayFlag_allSubtitlesForced`

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

CMTextFormatType

Text media format/subtype.

```
enum {
    kCMTextFormatType_QTText           = 'text',
    kCMTextFormatType_3GText           = 'tx3g'
};
typedef FourCharCode CMTextFormatType;
```

Constants

`kCMTextFormatType_QTText`

QuickTime Text media.

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

`kCMTextFormatType_3GText`

3GPP Text media.

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

CMTextJustificationValue

Justification modes for text media.

```
enum {
    kCMTextJustification_left_top        =  0,
    kCMTextJustification_centered        =  1,
    kCMTextJustification_bottom_right    = -1
};
typedef int8_t CMTextJustificationValue;
```

Constants

`kCMTextJustification_left_top`

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

`kCMTextJustification_centered`

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

`kCMTextJustification_bottom_right`

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

CMAudioCodecType

Codes to identify audio codecs.

```
enum {
    kCMAudioCodecType_AAC_LCProtected      = 'paac',
    kCMAudioCodecType_AAC_AudibleProtected = 'aaac'
};
typedef FourCharCode CMAudioCodecType;
```

Constants

`kCMAudioCodecType_AAC_LCProtected`
iTMS protected low-complexity AAC.
Available in iOS 4.0 and later.
Declared in `CMFormatDescription.h`.

`kCMAudioCodecType_AAC_AudibleProtected`
Audible's protected AAC.
Available in iOS 4.0 and later.
Declared in `CMFormatDescription.h`.

Discussion
Certain codec types are also audio formats.

Audio Format Description Masks

Mask bits representing various parts of an audio format description.

```
enum
{
    kCMAudioFormatDescriptionMask_StreamBasicDescription = (1<<0),
    kCMAudioFormatDescriptionMask_MagicCookie            = (1<<1),
    kCMAudioFormatDescriptionMask_ChannelLayout          = (1<<2),
    kCMAudioFormatDescriptionMask_Extensions             = (1<<3),
    kCMAudioFormatDescriptionMask_All                    =
kCMAudioFormatDescriptionMask_StreamBasicDescription
                                                         |
kCMAudioFormatDescriptionMask_MagicCookie
                                                         |
kCMAudioFormatDescriptionMask_ChannelLayout
                                                         |
kCMAudioFormatDescriptionMask_Extensions
};
```

Constants

`CMAudioFormatDescriptionMask_StreamBasicDescription`
Represents the AudioStreamBasicDescription.

`CMAudioFormatDescriptionMask_MagicCookie`
Represents the magic cookie.

`CMAudioFormatDescriptionMask_ChannelLayout`
Represents the AudioChannelLayout.

`CMAudioFormatDescriptionMask_Extensions`
Represents the format description extensions.

`CMAudioFormatDescriptionMask_All`
Represents all the parts of an audio format description.

Discussion
These components form the bitmask passed to (and returned from) `CMAudioFormatDescriptionEqual` (page 49).

CMMuxedStreamType

Muxed media format/subtype.

```
enum {
    kCMMuxedStreamType_MPEG1System       = 'mp1s',
    kCMMuxedStreamType_MPEG2Transport    = 'mp2t',
    kCMMuxedStreamType_MPEG2Program      = 'mp2p',
    kCMMuxedStreamType_DV                = 'dv  '
};
typedef FourCharCode CMMuxedStreamType;
```

Constants

`kCMMuxedStreamType_MPEG1System`
MPEG-1 System stream.
Available in iOS 4.0 and later.
Declared in `CMFormatDescription.h`.

`kCMMuxedStreamType_MPEG2Transport`
MPEG-2 Transport stream.
Available in iOS 4.0 and later.
Declared in `CMFormatDescription.h`.

`kCMMuxedStreamType_MPEG2Program`
MPEG-2 Program stream.
Available in iOS 4.0 and later.
Declared in `CMFormatDescription.h`.

`kCMMuxedStreamType_DV`
DV stream.
Available in iOS 4.0 and later.
Declared in `CMFormatDescription.h`.

CMClosedCaptionFormatType

Four-character codes identifying closed-caption media format types.

```
enum {
    kCMClosedCaptionFormatType_CEA608    = 'c608',
    kCMClosedCaptionFormatType_CEA708    = 'c708'
};
typedef FourCharCode CMClosedCaptionFormatType;
```

Constants

`kCMClosedCaptionFormatType_CEA608`

CEA 608-compliant samples.

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

`kCMClosedCaptionFormatType_CEA708`

CEA 708-compliant samples.

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

CMPixelFormatType

Four-character codes identifying pixel formats.

```
enum {
    kCMPixelFormat_32ARGB          = 32,
    kCMPixelFormat_32BGRA          = 'BGRA',
    kCMPixelFormat_24RGB           = 24,
    kCMPixelFormat_16BE555         = 16,
    kCMPixelFormat_16BE565         = 'B565',
    kCMPixelFormat_16LE555         = 'L555',
    kCMPixelFormat_16LE565         = 'L565',
    kCMPixelFormat_16LE5551        = '5551',
    kCMPixelFormat_422YpCbCr8      = '2vuy',
    kCMPixelFormat_422YpCbCr8_yuvs = 'yuvs',
    kCMPixelFormat_444YpCbCr8      = 'v308',
    kCMPixelFormat_4444YpCbCrA8    = 'v408',
    kCMPixelFormat_422YpCbCr16     = 'v216',
    kCMPixelFormat_422YpCbCr10     = 'v210',
    kCMPixelFormat_444YpCbCr10     = 'v410',
    kCMPixelFormat_8IndexedGray_WhiteIsZero = 0x00000028,
};
typedef FourCharCode CMPixelFormatType;
```

Constants

`kCMPixelFormat_32ARGB`

32-bit ARGB.

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

`kCMPixelFormat_32BGRA`

32-bit BGRA.

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

`kCMPixelFormat_24RGB`
24-bit RGB.

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

`kCMPixelFormat_16BE555`
16-bit big-endian 5-5-5.

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

`kCMPixelFormat_16BE565`
16-bit big-endian 5-6-5.

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

`kCMPixelFormat_16LE555`
16-bit little-endian 5-5-5.

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

`kCMPixelFormat_16LE565`
16-bit little-endian 5-6-5.

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

`kCMPixelFormat_16LE5551`
16-bit little-endian 5-5-5-1.

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

`kCMPixelFormat_422YpCbCr8`
Component Y'CbCr 8-bit 4:2:2 ordered Cb Y'0 Cr Y'1.

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

`kCMPixelFormat_422YpCbCr8_yuvs`
Component Y'CbCr 8-bit 4:2:2 ordered Y'0 Cb Y'1 Cr.

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

`kCMPixelFormat_444YpCbCr8`
Component Y'CbCr 8-bit 4:4:4.

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

`kCMPixelFormat_4444YpCbCrA8`
Component Y'CbCrA 8-bit 4:4:4:4.

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

`kCMPixelFormat_422YpCbCr16`
Component Y'CbCr 10,12,14,16-bit 4:2:2.

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

`kCMPixelFormat_422YpCbCr10`
Component Y'CbCr 10-bit 4:2:2.

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

`kCMPixelFormat_444YpCbCr10`
Component Y'CbCr 10-bit 4:4:4

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

`kCMPixelFormat_8IndexedGray_WhiteIsZero`
8 bit indexed gray, white is zero.

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

Discussion
Only some codec types are pixel formats.

CMMetadataFormatType

The types of TimedMetadata.

```
enum {
    kCMTimedMetadataFormatType_ICY      = 'icy ',
    kCMTimedMetadataFormatType_ID3      = 'id3 ',
    kCMTimedMetadataFormatType_Boxed    = 'mebx',
};
typedef FourCharCode CMMetadataFormatType;
```

Constants
`kCMTimedMetadataFormatType_ICY`
SHOUTCast format.

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

`kCMTimedMetadataFormatType_ID3`
ID3 format.

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

`kCMTimedMetadataFormatType_Boxed`
Boxed format.

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

MPEG-2-conformant Formats

Constants to access MPEG-2 attributes.

```
CFStringRef kCMFormatDescriptionConformsToMPEG2VideoProfile;*
CFStringRef kCMFormatDescriptionExtension_TemporalQuality;
CFStringRef kCMFormatDescriptionExtension_SpatialQuality;
CFStringRef kCMFormatDescriptionExtension_Version;
CFStringRef kCMFormatDescriptionExtension_RevisionLevel;
CFStringRef kCMFormatDescriptionExtension_Vendor;
CFStringRef kCMFormatDescriptionVendor_Apple;
```

Constants

`kCMFormatDescriptionConformsToMPEG2VideoProfile`
The value is a CFNumber specifying a kCMMPEG2VideoProfile.

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

`kCMFormatDescriptionExtension_TemporalQuality`
The value is a CFNumber.

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

`kCMFormatDescriptionExtension_SpatialQuality`
The value is a CFNumber.

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

`kCMFormatDescriptionExtension_Version`
The value is a CFNumber.

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

`kCMFormatDescriptionExtension_RevisionLevel`
The value is a CFNumber.

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

`kCMFormatDescriptionExtension_Vendor`
The value is a CFString of four character codes.

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

`kCMFormatDescriptionVendor_Apple`
A CFString specifying Apple as the vendor.

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

CMTextFormatDescription Constants

These keys and values are used by text-based format descriptions.

```
// Extension keys and values common to kCMTextFormatType_QTText and
kCMTextFormatType_3GText format descriptions
CFStringRef kCMTextFormatDescriptionExtension_DisplayFlags;
CFStringRef kCMTextFormatDescriptionExtension_BackgroundColor;
CFStringRef kCMTextFormatDescriptionColor_Red;
CFStringRef kCMTextFormatDescriptionColor_Green;
CFStringRef kCMTextFormatDescriptionColor_Blue;
CFStringRef kCMTextFormatDescriptionColor_Alpha;
CFStringRef kCMTextFormatDescriptionExtension_DefaultTextBox;
CFStringRef kCMTextFormatDescriptionRect_Top;
CFStringRef kCMTextFormatDescriptionRect_Left;
CFStringRef kCMTextFormatDescriptionRect_Bottom;
CFStringRef kCMTextFormatDescriptionRect_Right;
CFStringRef kCMTextFormatDescriptionExtension_DefaultStyle;
CFStringRef kCMTextFormatDescriptionStyle_StartChar;
CFStringRef kCMTextFormatDescriptionStyle_Font;
CFStringRef kCMTextFormatDescriptionStyle_FontFace;
CFStringRef kCMTextFormatDescriptionStyle_ForegroundColor;
CFStringRef kCMTextFormatDescriptionStyle_FontSize;

// Extension keys and values specific to kCMTextFormatType_3GText
CFStringRef kCMTextFormatDescriptionExtension_HorizontalJustification;
CFStringRef kCMTextFormatDescriptionExtension_VerticalJustification;

// Extension keys and values specific to the kCMTextFormatType_3GText
kCMTextFormatDescriptionExtension_DefaultStyle dictionary
CFStringRef kCMTextFormatDescriptionStyle_EndChar;
CFStringRef kCMTextFormatDescriptionExtension_FontTable;

// Extension keys and values specific to kCMTextFormatType_QTText
CFStringRef kCMTextFormatDescriptionExtension_TextJustification;

// Extension keys and values specific to the kCMTextFormatType_QTText
kCMTextFormatDescriptionExtension_DefaultStyle dictionary
CFStringRef kCMTextFormatDescriptionStyle_Height;
CFStringRef kCMTextFormatDescriptionStyle_Ascent;
CFStringRef kCMTextFormatDescriptionExtension_DefaultFontName;
```

Constants

`kCMTextFormatDescriptionExtension_DisplayFlags`

The value is a CFNumber (an SInt32 holding CMTextDisplayFlags).

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

`kCMTextFormatDescriptionExtension_BackgroundColor`

The value is a CFDictionary.

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

`kCMTextFormatDescriptionColor_Red`

The value is a CFNumber (SInt8 for 3G), (SInt16 for QT).

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

`kCMTextFormatDescriptionColor_Green`
The value is a CFNumber (SInt8 for 3G), (SInt16 for QT).

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

`kCMTextFormatDescriptionColor_Blue`
The value is a CFNumber (SInt8 for 3G), (SInt16 for QT).

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

`kCMTextFormatDescriptionColor_Alpha`
The value is a CFNumber (SInt8 for 3G), not applicable for QT text.

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

`kCMTextFormatDescriptionExtension_DefaultTextBox`
The value is a CFDictionary.

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

`kCMTextFormatDescriptionRect_Top`
The value is a CFNumber (SInt16).

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

`kCMTextFormatDescriptionRect_Left`
The value is a CFNumber (SInt16).

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

`kCMTextFormatDescriptionRect_Bottom`
The value is a CFNumber (SInt16).

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

`kCMTextFormatDescriptionRect_Right`
The value is a CFNumber (SInt16).

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

`kCMTextFormatDescriptionExtension_DefaultStyle`
The value is a CFDictionary.

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

`kCMTextFormatDescriptionStyle_StartChar`
The value is a CFNumber (SInt16 for 3G), (SInt32 for QT).

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

`kCMTextFormatDescriptionStyle_Font`
The value is a CFNumber (SInt16).

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

`kCMTextFormatDescriptionStyle_FontFace`
The value is a CFNumber (SInt8).

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

`kCMTextFormatDescriptionStyle_ForegroundColor`
The value is a CFDictionary.

The dictionary contains values for `kCMTextFormatDescriptionColor_Red`, `kCMTextFormatDescriptionColor_Green`, and so on.

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

`kCMTextFormatDescriptionExtension_HorizontalJustification`
The value is a CFNumber (SInt8) containing a CMTextJustificationValue.

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

`kCMTextFormatDescriptionExtension_VerticalJustification`
The value is a CFNumber (SInt8) containing a CMTextJustificationValue.

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

`kCMTextFormatDescriptionStyle_EndChar`
The value is a CFNumber (SInt16).

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

`kCMTextFormatDescriptionExtension_FontTable`
The value is a CFDictionary.

Keys are FontIDs as CFStrings, values are font names as CFStrings.

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

`kCMTextFormatDescriptionExtension_TextJustification`
The value is a CFNumber (SInt8) containing a CMTextJustificationValue.

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

`kCMTextFormatDescriptionStyle_Height`
The value is a CFNumber (SInt16).

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

`kCMTextFormatDescriptionStyle_Ascent`
The value is a CFNumber (SInt16).

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

`kCMTextFormatDescriptionExtension_DefaultFontName`
The value is a CFString.

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

CMVideoCodecType

Four-character codes identifying the video codec.

```
enum {
    kCMVideoCodecType_422YpCbCr8       = kCMPixelFormat_422YpCbCr8,
    kCMVideoCodecType_Animation        = 'rle ',
    kCMVideoCodecType_Cinepak          = 'cvid',
    kCMVideoCodecType_JPEG             = 'jpeg',
    kCMVideoCodecType_JPEG_OpenDML     = 'dmb1',
    kCMVideoCodecType_SorensonVideo    = 'SVQ1',
    kCMVideoCodecType_SorensonVideo3   = 'SVQ3',
    kCMVideoCodecType_H263             = 'h263',
    kCMVideoCodecType_H264             = 'avc1',
    kCMVideoCodecType_MPEG4Video       = 'mp4v',
    kCMVideoCodecType_MPEG2Video       = 'mp2v',
    kCMVideoCodecType_MPEG1Video       = 'mp1v',

    kCMVideoCodecType_DVCNTSC          = 'dvc ',
    kCMVideoCodecType_DVCPAL           = 'dvcp',
    kCMVideoCodecType_DVCProPAL        = 'dvpp',
    kCMVideoCodecType_DVCPro50NTSC     = 'dv5n',
    kCMVideoCodecType_DVCPro50PAL      = 'dv5p',
    kCMVideoCodecType_DVCPROHD720p60   = 'dvhp',
    kCMVideoCodecType_DVCPROHD720p50   = 'dvhq',
    kCMVideoCodecType_DVCPROHD1080i60  = 'dvh6',
    kCMVideoCodecType_DVCPROHD1080i50  = 'dvh5',
    kCMVideoCodecType_DVCPROHD1080p30  = 'dvh3',
    kCMVideoCodecType_DVCPROHD1080p25  = 'dvh2',
};
typedef FourCharCode CMVideoCodecType;
```

Constants

`kCMVideoCodecType_422YpCbCr8`
Component Y'CbCr 8-bit 4:2:2 ordered Cb Y'0 Cr Y'1.

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

`kCMVideoCodecType_Animation`
Apple Animation format.

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

`kCMVideoCodecType_Cinepak`
Cinepak format.

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

`kCMVideoCodecType_JPEG`
: Joint Photographic Experts Group (JPEG) format.

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

`kCMVideoCodecType_JPEG_OpenDML`
: JPEG format with Open-DML extensions.

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

`kCMVideoCodecType_SorensonVideo`
: Sorenson video format.

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

`kCMVideoCodecType_SorensonVideo3`
: Sorenson 3 video format.

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

`kCMVideoCodecType_H263`
: ITU-T H.263 format.

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

`kCMVideoCodecType_H264`
: ITU-T H.264 format (also known as ISO/IEC 14496-10 - MPEG-4 Part 10, Advanced Video Coding format).

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

`kCMVideoCodecType_MPEG4Video`
: ISO/IEC Moving Picture Experts Group (MPEG) MPEG-4 Part 2 video format.

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

`kCMVideoCodecType_MPEG2Video`
: MPEG-2 video format.

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

`kCMVideoCodecType_MPEG1Video`
: MPEG-1 video format.

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

`kCMVideoCodecType_DVCNTSC`
: DV NTSC format.

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

`kCMVideoCodecType_DVCPAL`
DV PAL format.

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

`kCMVideoCodecType_DVCProPAL`
Panasonic DVCPro PAL format.

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

`kCMVideoCodecType_DVCPro50NTSC`
Panasonic DVCPro-50 NTSC format.

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

`kCMVideoCodecType_DVCPro50PAL`
Panasonic DVCPro-50 PAL format.

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

`kCMVideoCodecType_DVCPROHD720p60`
Panasonic DVCPro-HD 720p60 format.

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

`kCMVideoCodecType_DVCPROHD720p50`
Panasonic DVCPro-HD 720p50 format.

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

`kCMVideoCodecType_DVCPROHD1080i60`
Panasonic DVCPro-HD 1080i60 format.

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

`kCMVideoCodecType_DVCPROHD1080i50`
Panasonic DVCPro-HD 1080i50 format.

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

`kCMVideoCodecType_DVCPROHD1080p30`
Panasonic DVCPro-HD 1080p30 format.

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

`kCMVideoCodecType_DVCPROHD1080p25`
Panasonic DVCPro-HD 1080p25 format.

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

Discussion
Certain codec types are also pixel formats.

There is no “kCMVideoCodecType_Raw”; you should use the appropriate pixel format type as the codec type.

Video Profiles

Constants for video profiles.

```
enum
{
    kCMMPEG2VideoProfile_HDV_720p30 =               'hdv1',
    kCMMPEG2VideoProfile_HDV_1080i60 =              'hdv2',
    kCMMPEG2VideoProfile_HDV_1080i50 =              'hdv3',
    kCMMPEG2VideoProfile_HDV_720p24 =               'hdv4',
    kCMMPEG2VideoProfile_HDV_720p25 =               'hdv5',
    kCMMPEG2VideoProfile_HDV_1080p24 =              'hdv6',
    kCMMPEG2VideoProfile_HDV_1080p25 =              'hdv7',
    kCMMPEG2VideoProfile_HDV_1080p30 =              'hdv8',
    kCMMPEG2VideoProfile_HDV_720p60 =               'hdv9',
    kCMMPEG2VideoProfile_HDV_720p50 =               'hdva',
    kCMMPEG2VideoProfile_XDCAM_HD_1080i60_VBR35 =   'xdv2',
    kCMMPEG2VideoProfile_XDCAM_HD_1080i50_VBR35 =   'xdv3',
    kCMMPEG2VideoProfile_XDCAM_HD_1080p24_VBR35 =   'xdv6',
    kCMMPEG2VideoProfile_XDCAM_HD_1080p25_VBR35 =   'xdv7',
    kCMMPEG2VideoProfile_XDCAM_HD_1080p30_VBR35 =   'xdv8',
    kCMMPEG2VideoProfile_XDCAM_EX_720p24_VBR35 =    'xdv4',
    kCMMPEG2VideoProfile_XDCAM_EX_720p25_VBR35 =    'xdv5',
    kCMMPEG2VideoProfile_XDCAM_EX_720p30_VBR35 =    'xdv1',
    kCMMPEG2VideoProfile_XDCAM_EX_720p50_VBR35 =    'xdva',
    kCMMPEG2VideoProfile_XDCAM_EX_720p60_VBR35 =    'xdv9',
    kCMMPEG2VideoProfile_XDCAM_EX_1080i60_VBR35 =   'xdvb',
    kCMMPEG2VideoProfile_XDCAM_EX_1080i50_VBR35 =   'xdvc',
    kCMMPEG2VideoProfile_XDCAM_EX_1080p24_VBR35 =   'xdvd',
    kCMMPEG2VideoProfile_XDCAM_EX_1080p25_VBR35 =   'xdve',
    kCMMPEG2VideoProfile_XDCAM_EX_1080p30_VBR35 =   'xdvf',
    kCMMPEG2VideoProfile_XDCAM_HD422_720p50_CBR50 =  'xd5a',
    kCMMPEG2VideoProfile_XDCAM_HD422_720p60_CBR50 =  'xd59',
    kCMMPEG2VideoProfile_XDCAM_HD422_1080i60_CBR50 = 'xd5b',
    kCMMPEG2VideoProfile_XDCAM_HD422_1080i50_CBR50 = 'xd5c',
    kCMMPEG2VideoProfile_XDCAM_HD422_1080p24_CBR50 = 'xd5d',
    kCMMPEG2VideoProfile_XDCAM_HD422_1080p25_CBR50 = 'xd5e',
    kCMMPEG2VideoProfile_XDCAM_HD422_1080p30_CBR50 = 'xd5f',
    kCMMPEG2VideoProfile_XDCAM_HD_540p =            'xdhd',
    kCMMPEG2VideoProfile_XDCAM_HD422_540p =         'xdh2',
};
```

Constants

`kCMMPEG2VideoProfile_HDV_720p30`

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

`kCMMPEG2VideoProfile_HDV_1080i60`

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

`kCMMPEG2VideoProfile_HDV_1080i50`

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

`kCMMPEG2VideoProfile_HDV_720p24`

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

`kCMMPEG2VideoProfile_HDV_720p25`

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

`kCMMPEG2VideoProfile_HDV_1080p24`

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

`kCMMPEG2VideoProfile_HDV_1080p25`

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

`kCMMPEG2VideoProfile_HDV_1080p30`

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

`kCMMPEG2VideoProfile_HDV_720p60`

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

`kCMMPEG2VideoProfile_HDV_720p50`

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

`kCMMPEG2VideoProfile_XDCAM_HD_1080i60_VBR35`

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

`kCMMPEG2VideoProfile_XDCAM_HD_1080i50_VBR35`

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

`kCMMPEG2VideoProfile_XDCAM_HD_1080p24_VBR35`

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

`kCMMPEG2VideoProfile_XDCAM_HD_1080p25_VBR35`

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

`kCMMPEG2VideoProfile_XDCAM_HD_1080p30_VBR35`

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

`kCMMPEG2VideoProfile_XDCAM_EX_720p24_VBR35`

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

`kCMMPEG2VideoProfile_XDCAM_EX_720p25_VBR35`

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

`kCMMPEG2VideoProfile_XDCAM_EX_720p30_VBR35`

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

`kCMMPEG2VideoProfile_XDCAM_EX_720p50_VBR35`

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

`kCMMPEG2VideoProfile_XDCAM_EX_720p60_VBR35`

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

`kCMMPEG2VideoProfile_XDCAM_EX_1080i60_VBR35`

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

`kCMMPEG2VideoProfile_XDCAM_EX_1080i50_VBR35`

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

`kCMMPEG2VideoProfile_XDCAM_EX_1080p24_VBR35`

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

`kCMMPEG2VideoProfile_XDCAM_EX_1080p25_VBR35`

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

`kCMMPEG2VideoProfile_XDCAM_EX_1080p30_VBR35`

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

`kCMMPEG2VideoProfile_XDCAM_HD422_720p50_CBR50`

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

`kCMMPEG2VideoProfile_XDCAM_HD422_720p60_CBR50`

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

`kCMMPEG2VideoProfile_XDCAM_HD422_1080i60_CBR50`

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

`kCMMPEG2VideoProfile_XDCAM_HD422_1080i50_CBR50`

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

`kCMMPEG2VideoProfile_XDCAM_HD422_1080p24_CBR50`

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

`kCMMPEG2VideoProfile_XDCAM_HD422_1080p25_CBR50`

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

`kCMMPEG2VideoProfile_XDCAM_HD422_1080p30_CBR50`

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

`kCMMPEG2VideoProfile_XDCAM_HD_540p`

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

`kCMMPEG2VideoProfile_XDCAM_HD422_540p`

Available in iOS 4.0 and later.

Declared in `CMFormatDescription.h`.

CMSampleBuffer Reference

Derived From:	*CFType Reference*
Framework:	CoreMedia.framework
Declared in	CMSampleBuffer.h

Overview

This document describes the API you use to create and manipulate CMSampleBuffer structs.

CMSampleBuffers are Core Foundation objects containing zero or more compressed (or uncompressed) samples of a particular media type (audio, video, muxed, etc), that are used to move media sample data through the media system. A CMSampleBuffer can contain:

- A CMBlockBuffer of one or more media samples, or
- A CVImageBuffer, a reference to the format description for the stream of CMSampleBuffers, size and timing information for each of the contained media samples, and both buffer-level and sample-level attachments.

The buffer-level attachments of a CMSampleBuffer are distinct from the attachments of its contained CMBlockBuffer. An example of a sample-level attachment is an annotation about video frame dependencies (such as “droppable,” “other frames depend on me,” or “I depend on other frames”). Each sample (video frame) in the CMSampleBuffer would need its own attachment in this case. Another sample-level attachment example is SMPTE timecode acquired during capture. To get and set a CMSampleBuffer's buffer-level attachments, use the APIs in `CMAttachmentBearerRef` (page 160).

It is possible for a CMSampleBuffer to describe samples it does not yet contain. For example, some media services may have access to sample size, timing and format information before the data is read. Such services may create CMSampleBuffers with that information and insert them into queues early, and attach (or fill) the CMBlockBuffers of media data later, when the data becomes ready. To this end, CMSampleBuffers have the concept of data-readiness, which can be tested, set, forced to become ready “now" and so on. It is also possible for a CMSampleBuffer to contain nothing but a special buffer-level attachment that describes a media stream event (for example, "discontinuity: drain and reset decoder before processing the next CMSampleBuffer”). Such a special attachment can also be attached to regular CMSampleBuffers (i.e. that contain media sample data), and if so, the event it describes is defined to occur after the samples in that CMSampleBuffer.

Functions

CMAudioSampleBufferCreateWithPacketDescriptions

```
OSStatus CMAudioSampleBufferCreateWithPacketDescriptions (
   CFAllocatorRef allocator,
   CMBlockBufferRef dataBuffer,
   Boolean dataReady,
   CMSampleBufferMakeDataReadyCallback makeDataReadyCallback,
   void *makeDataReadyRefcon,
   CMFormatDescriptionRef formatDescription,
   CMItemCount numSamples,
   CMTime sbufPTS,
   const AudioStreamPacketDescription *packetDescriptions,
   CMSampleBufferRef *sBufOut
);
```

Parameters

allocator

dataBuffer

dataReady

makeDataReadyCallback

makeDataReadyRefcon

formatDescription

numSamples

sbufPTS

packetDescriptions

sBufOut

Return Value

Discussion

Availability

Available in iOS 4.0 and later.

Declared In

CMSampleBuffer.h

CMSampleBufferCallForEachSample

```
OSStatus CMSampleBufferCallForEachSample (
   CMSampleBufferRef sbuf,
   OSStatus (*callback)(CMSampleBufferRef sampleBuffer, CMItemCount index, void
*refcon),
   void *refcon
);
```

Parameters

sbuf

refcon

refcon

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMSampleBuffer.h

CMSampleBufferCopySampleBufferForRange

```
OSStatus CMSampleBufferCopySampleBufferForRange (
   CFAllocatorRef allocator,
   CMSampleBufferRef sbuf,
   CFRange sampleRange,
   CMSampleBufferRef *sBufOut
);
```

Parameters

allocator

sbuf

sampleRange

sBufOut

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMSampleBuffer.h

CMSampleBufferCreate

```
OSStatus CMSampleBufferCreate (
   CFAllocatorRef allocator,
   CMBlockBufferRef dataBuffer,
   Boolean dataReady,
   CMSampleBufferMakeDataReadyCallback makeDataReadyCallback,
   void *makeDataReadyRefcon,
   CMFormatDescriptionRef formatDescription,
   CMItemCount numSamples,
   CMItemCount numSampleTimingEntries,
   const CMSampleTimingInfo *sampleTimingArray,
   CMItemCount numSampleSizeEntries,
   const size_t *sampleSizeArray,
   CMSampleBufferRef *sBufOut
);
```

Parameters

allocator

dataBuffer

dataReady

makeDataReadyCallback

makeDataReadyRefcon

formatDescription

numSamples

numSampleTimingEntries

sampleTimingArray

numSampleSizeEntries

sampleSizeArray

sBufOut

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMSampleBuffer.h

CMSampleBufferCreateCopy

```
OSStatus CMSampleBufferCreateCopy (
   CFAllocatorRef allocator,
   CMSampleBufferRef sbuf,
   CMSampleBufferRef *sbufCopyOut
);
```

Parameters

allocator

sbuf

sbufCopyOut

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMSampleBuffer.h

CMSampleBufferCreateCopyWithNewTiming

```
OSStatus CMSampleBufferCreateCopyWithNewTiming (
   CFAllocatorRef allocator,
   CMSampleBufferRef originalSBuf,
   CMItemCount numSampleTimingEntries,
   const CMSampleTimingInfo *sampleTimingArray,
   CMSampleBufferRef *sBufCopyOut
);
```

Parameters

allocator

originalSBuf

numSampleTimingEntries

sampleTimingArray

sBufCopyOut

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMSampleBuffer.h

CMSampleBufferCreateForImageBuffer

```
OSStatus CMSampleBufferCreateForImageBuffer (
   CFAllocatorRef allocator,
   CVImageBufferRef imageBuffer,
   Boolean dataReady,
   CMSampleBufferMakeDataReadyCallback makeDataReadyCallback,
   void *makeDataReadyRefcon,
   CMVideoFormatDescriptionRef formatDescription,
   const CMSampleTimingInfo *sampleTiming,
   CMSampleBufferRef *sBufOut
);
```

Parameters

allocator

imageBuffer

dataReady

makeDataReadyCallback

makeDataReadyRefcon

formatDescription

sampleTiming

sBufOut

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMSampleBuffer.h

CMSampleBufferDataIsReady

```
Boolean CMSampleBufferDataIsReady (
   CMSampleBufferRef sbuf
);
```

Parameters

sbuf

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMSampleBuffer.h

CMSampleBufferGetAudioBufferListWithRetainedBlockBuffer

```
OSStatus CMSampleBufferGetAudioBufferListWithRetainedBlockBuffer (
   CMSampleBufferRef sbuf,
   size_t *bufferListSizeNeededOut,
   AudioBufferList *bufferListOut,
   size_t bufferListSize,
   CFAllocatorRef bbufStructAllocator,
   CFAllocatorRef bbufMemoryAllocator,
   uint32_t flags,
   CMBlockBufferRef *blockBufferOut
);
```

Parameters

sbuf

bufferListSizeNeededOut

bufferListOut

bufferListSize

bbufStructAllocator

bbufMemoryAllocator

flags

blockBufferOut

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMSampleBuffer.h

CMSampleBufferGetAudioStreamPacketDescriptions

```
OSStatus CMSampleBufferGetAudioStreamPacketDescriptions (
   CMSampleBufferRef sbuf,
   size_t packetDescriptionsSize,
   AudioStreamPacketDescription *packetDescriptionsOut,
   size_t *packetDescriptionsSizeNeededOut
);
```

Parameters

sbuf

packetDescriptionsSize

packetDescriptionsOut

packetDescriptionsSizeNeededOut

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMSampleBuffer.h

CMSampleBufferGetAudioStreamPacketDescriptionsPtr

```
OSStatus CMSampleBufferGetAudioStreamPacketDescriptionsPtr (
   CMSampleBufferRef sbuf,
   const AudioStreamPacketDescription **packetDescriptionsPtrOut,
   size_t *packetDescriptionsSizeOut
);
```

Parameters

sbuf

packetDescriptionsPtrOut

packetDescriptionsSizeOut

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMSampleBuffer.h

CMSampleBufferGetDataBuffer

```
CMBlockBufferRef CMSampleBufferGetDataBuffer (
   CMSampleBufferRef sbuf
);
```

Parameters

sbuf

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMSampleBuffer.h

CMSampleBufferGetDecodeTimeStamp

```
CMTime CMSampleBufferGetDecodeTimeStamp (
   CMSampleBufferRef sbuf
);
```

Parameters

sbuf

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMSampleBuffer.h

CMSampleBufferGetDuration

```
CMTime CMSampleBufferGetDuration (
   CMSampleBufferRef sbuf
);
```

Parameters

sbuf

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMSampleBuffer.h

CMSampleBufferGetFormatDescription

```
CMFormatDescriptionRef CMSampleBufferGetFormatDescription (
   CMSampleBufferRef sbuf
);
```

Parameters
sbuf

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMSampleBuffer.h

CMSampleBufferGetImageBuffer

```
CVImageBufferRef CMSampleBufferGetImageBuffer (
   CMSampleBufferRef sbuf
);
```

Parameters
sbuf

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMSampleBuffer.h

CMSampleBufferGetNumSamples

```
CMItemCount CMSampleBufferGetNumSamples (
   CMSampleBufferRef sbuf
);
```

Parameters

sbuf

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMSampleBuffer.h

CMSampleBufferGetOutputDecodeTimeStamp

```
CMTime CMSampleBufferGetOutputDecodeTimeStamp (
   CMSampleBufferRef sbuf
);
```

Parameters

sbuf

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMSampleBuffer.h

CMSampleBufferGetOutputDuration

```
CMTime CMSampleBufferGetOutputDuration (
   CMSampleBufferRef sbuf
);
```

Parameters

sbuf

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMSampleBuffer.h

CMSampleBufferGetOutputPresentationTimeStamp

```
CMTime CMSampleBufferGetOutputPresentationTimeStamp (
   CMSampleBufferRef sbuf
);
```

Parameters

sbuf

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMSampleBuffer.h

CMSampleBufferGetOutputSampleTimingInfoArray

```
OSStatus CMSampleBufferGetOutputSampleTimingInfoArray (
   CMSampleBufferRef sbuf,
   CMItemCount timingArrayEntries,
   CMSampleTimingInfo *timingArrayOut,
   CMItemCount *timingArrayEntriesNeededOut
);
```

Parameters

sbuf

timingArrayEntries

timingArrayOut

timingArrayEntriesNeededOut

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMSampleBuffer.h

CMSampleBufferGetPresentationTimeStamp

```
CMTime CMSampleBufferGetPresentationTimeStamp (
   CMSampleBufferRef sbuf
);
```

Parameters

sbuf

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMSampleBuffer.h

CMSampleBufferGetSampleAttachmentsArray

```
CFArrayRef CMSampleBufferGetSampleAttachmentsArray (
   CMSampleBufferRef sbuf,
   Boolean createIfNecessary
);
```

Parameters

sbuf

createIfNecessary

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMSampleBuffer.h

CMSampleBufferGetSampleSize

```
size_t CMSampleBufferGetSampleSize (
   CMSampleBufferRef sbuf,
   CMItemIndex sampleIndex
);
```

Parameters

sbuf

sampleIndex

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMSampleBuffer.h

CMSampleBufferGetSampleSizeArray

```
OSStatus CMSampleBufferGetSampleSizeArray (
   CMSampleBufferRef sbuf,
   CMItemCount sizeArrayEntries,
   size_t *sizeArrayOut,
   CMItemCount *sizeArrayEntriesNeededOut
);
```

Parameters

sbuf

sizeArrayEntries

sizeArrayOut

sizeArrayEntriesNeededOut

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMSampleBuffer.h

CMSampleBufferGetSampleTimingInfo

```
OSStatus CMSampleBufferGetSampleTimingInfo (
   CMSampleBufferRef sbuf,
   CMItemIndex sampleIndex,
   CMSampleTimingInfo *timingInfoOut
);
```

Parameters

sbuf

sampleIndex

timingInfoOut

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMSampleBuffer.h

CMSampleBufferGetSampleTimingInfoArray

```
OSStatus CMSampleBufferGetSampleTimingInfoArray (
   CMSampleBufferRef sbuf,
   CMItemCount timingArrayEntries,
   CMSampleTimingInfo *timingArrayOut,
   CMItemCount *timingArrayEntriesNeededOut
);
```

Parameters

sbuf

timingArrayEntries

timingArrayOut

timingArrayEntriesNeededOut

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMSampleBuffer.h

CMSampleBufferGetTotalSampleSize

```
size_t CMSampleBufferGetTotalSampleSize (
   CMSampleBufferRef sbuf
);
```

Parameters

sbuf

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMSampleBuffer.h

CMSampleBufferGetTypeID

```
CFTypeID CMSampleBufferGetTypeID (
   void
);
```

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMSampleBuffer.h

CMSampleBufferInvalidate

```
OSStatus CMSampleBufferInvalidate (
   CMSampleBufferRef sbuf
);
```

Parameters

sbuf

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMSampleBuffer.h

CMSampleBufferIsValid

```
Boolean CMSampleBufferIsValid (
   CMSampleBufferRef sbuf
);
```

Parameters

sbuf

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMSampleBuffer.h

CMSampleBufferMakeDataReady

```
OSStatus CMSampleBufferMakeDataReady (
   CMSampleBufferRef sbuf
);
```

Parameters

sbuf

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMSampleBuffer.h

CMSampleBufferSetDataBuffer

```
OSStatus CMSampleBufferSetDataBuffer (
   CMSampleBufferRef sbuf,
   CMBlockBufferRef dataBuffer
);
```

Parameters

sbuf

dataBuffer

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMSampleBuffer.h

CMSampleBufferSetDataBufferFromAudioBufferList

```
OSStatus CMSampleBufferSetDataBufferFromAudioBufferList (
   CMSampleBufferRef sbuf,
   CFAllocatorRef bbufStructAllocator,
   CFAllocatorRef bbufMemoryAllocator,
   uint32_t flags,
   const AudioBufferList *bufferList
);
```

Parameters

sbuf

bbufStructAllocator

bbufMemoryAllocator

flags

bufferList

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMSampleBuffer.h

CMSampleBufferSetDataReady

```
OSStatus CMSampleBufferSetDataReady (
   CMSampleBufferRef sbuf
);
```

Parameters

sbuf

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMSampleBuffer.h

CMSampleBufferSetInvalidateCallback

```
OSStatus CMSampleBufferSetInvalidateCallback (
   CMSampleBufferRef sbuf,
   CMSampleBufferInvalidateCallback invalidateCallback,
   uint64_t invalidateRefCon
);
```

Parameters

sbuf

invalidateCallback

invalidateRefCon

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMSampleBuffer.h

CMSampleBufferSetOutputPresentationTimeStamp

```
OSStatus CMSampleBufferSetOutputPresentationTimeStamp (
   CMSampleBufferRef sbuf,
   CMTime outputPresentationTimeStamp
);
```

Parameters

sbuf

outputPresentationTimeStamp

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMSampleBuffer.h

CMSampleBufferTrackDataReadiness

```
OSStatus CMSampleBufferTrackDataReadiness (
   CMSampleBufferRef sbuf,
   CMSampleBufferRef sbufToTrack
);
```

Parameters

sbuf

sbufToTrack

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMSampleBuffer.h

Data Types

CMSampleBufferRef

A reference to an immutable CMSampleBufferRef object.

```
typedef struct opaqueCMSampleBuffer *CMSampleBufferRef;
```

Discussion
A CMSampleBuffer is a Core Foundation object containing zero or more compressed (or uncompressed) samples of a particular media type (audio, video, muxed, and so on).

Availability
Available in iOS 4.0 and later.

Declared In
CMSampleBuffer.h

CMSampleTimingInfo

A collection of timing information for a sample in a CMSampleBuffer.

```
typedef struct
{
    CMTime duration;
    CMTime presentationTimeStamp;
    CMTime decodeTimeStamp;
} CMSampleTimingInfo;
```

Fields

`duration`

The duration of the sample.

If a single struct applies to each of the samples, they all have this duration.

`presentationTimeStamp`

The time at which the sample will be presented.

If a single struct applies to each of the samples, they all have this duration.

`decodeTimeStamp`

The time at which the sample will be decoded.

If the samples are in presentation order, this must be set to `kCMInvalidTime`.

Discussion

A single CMSampleTimingInfo struct can describe every individual sample in a CMSampleBuffer, if the samples all have the same duration and are in presentation order with no gaps.

Availability

Available in iOS 4.0 and later.

Declared In

`CMSampleBuffer.h`

Constants

Error Codes

Error codes returned from the CMSampleBuffer functions.

```
enum {
    kCMSampleBufferError_AllocationFailed               = -12730,
    kCMSampleBufferError_RequiredParameterMissing       = -12731,
    kCMSampleBufferError_AlreadyHasDataBuffer           = -12732,
    kCMSampleBufferError_BufferNotReady                 = -12733,
    kCMSampleBufferError_SampleIndexOutOfRange          = -12734,
    kCMSampleBufferError_BufferHasNoSampleSizes         = -12735,
    kCMSampleBufferError_BufferHasNoSampleTimingInfo    = -12736,
    kCMSampleBufferError_ArrayTooSmall                  = -12737,
    kCMSampleBufferError_InvalidEntryCount              = -12738,
    kCMSampleBufferError_CannotSubdivide                = -12739,
    kCMSampleBufferError_SampleTimingInfoInvalid        = -12740,
    kCMSampleBufferError_InvalidMediaTypeForOperation   = -12741,
    kCMSampleBufferError_InvalidSampleData              = -12742,
    kCMSampleBufferError_InvalidMediaFormat             = -12743,
    kCMSampleBufferError_Invalidated                    = -12744,
};
```

Constants

`kCMSampleBufferError_AllocationFailed`

Indicates that an allocation failed.

Available in iOS 4.0 and later.

Declared in `CMSampleBuffer.h`.

`kCMSampleBufferError_RequiredParameterMissing`

Indicates that `NULL` or `0` was passed for a required parameter.

Available in iOS 4.0 and later.

Declared in `CMSampleBuffer.h`.

`kCMSampleBufferError_AlreadyHasDataBuffer`

Indicates that an attempt was made to set a data buffer on a CMSampleBuffer that already has one.

Available in iOS 4.0 and later.

Declared in `CMSampleBuffer.h`.

`kCMSampleBufferError_BufferNotReady`

Indicates that the buffer could not be made ready.

Available in iOS 4.0 and later.

Declared in `CMSampleBuffer.h`.

`kCMSampleBufferError_SampleIndexOutOfRange`

Indicates that the sample index was not between `0` and `numSamples-1`, inclusive.

Available in iOS 4.0 and later.

Declared in `CMSampleBuffer.h`.

`kCMSampleBufferError_BufferHasNoSampleSizes`

Indicates that there was an attempt to get sample size information when there was none.

Available in iOS 4.0 and later.

Declared in `CMSampleBuffer.h`.

`kCMSampleBufferError_BufferHasNoSampleTimingInfo`

Indicates that there was an attempt to get sample timing information when there was none.

Available in iOS 4.0 and later.

Declared in `CMSampleBuffer.h`.

`kCMSampleBufferError_ArrayTooSmall`
Indicates that the output array was not large enough for the array being requested.

Available in iOS 4.0 and later.

Declared in `CMSampleBuffer.h`.

`kCMSampleBufferError_InvalidEntryCount`
Indicates that the timing info or size array entry count was not `0`, `1`, or `numSamples`.

Available in iOS 4.0 and later.

Declared in `CMSampleBuffer.h`.

`kCMSampleBufferError_CannotSubdivide`
Indicates that the sample buffer does not contain sample sizes.

This can happen when the samples in the buffer are non-contiguous (for example, in non-interleaved audio, where the channel values for a single sample are scattered through the buffer).

Available in iOS 4.0 and later.

Declared in `CMSampleBuffer.h`.

`kCMSampleBufferError_SampleTimingInfoInvalid`
Indicates that the buffer unexpectedly contains a non-numeric sample timing info.

Available in iOS 4.0 and later.

Declared in `CMSampleBuffer.h`.

`kCMSampleBufferError_InvalidMediaTypeForOperation`
Indicates that the media type specified by a format description is not valid for the given operation.

For example, a CMSampleBuffer with a non-audio format description was passed to CMSampleBufferGetAudioStreamPacketDescriptionsPtr.

Available in iOS 4.0 and later.

Declared in `CMSampleBuffer.h`.

`kCMSampleBufferError_InvalidSampleData`
Indicates that Buffer contains bad data.

This value is only returned by CMSampleBuffer functions that inspect its sample data.

Available in iOS 4.0 and later.

Declared in `CMSampleBuffer.h`.

`kCMSampleBufferError_InvalidMediaFormat`
Indicates that the format of the given media does not match the given format description.

For example, a format description paired with a CVImageBuffer that fails CMVideoFormatDescriptionMatchesImageBuffer.

Available in iOS 4.0 and later.

Declared in `CMSampleBuffer.h`.

`kCMSampleBufferError_Invalidated`
Indicates that the sample buffer was invalidated..

Available in iOS 4.0 and later.

Declared in `CMSampleBuffer.h`.

CMTime Reference

Derived From:	*CFType Reference*
Framework:	CoreMedia
Declared in	CMTime.h

Overview

This document describes the API for creating and manipulating CMTime structs.

CMTime structs are non-opaque mutable structs representing times (either timestamps or durations).

A CMTime is represented as a rational number, with a numerator (an `int64_t` value), and a denominator (an `int32_t` timescale). A flags field allows various non-numeric values to be stored (+infinity, -infinity, indefinite, invalid). There is also a flag to mark whether or not the time is completely precise, or had to be rounded at some point in its past.

CMTimes contain an epoch number, which is usually set to 0, but can be used to distinguish unrelated timelines: for example, it could be incremented each time through a presentation loop, to differentiate between time N in loop 0 from time N in loop 1.

You can convert CMTimes to and from immutable CFDictionaries (see `CFDictionaryRef`) using `CMTimeCopyAsDictionary` (page 117) and `CMTimeMakeFromDictionary` (page 118), for use in annotations and various Core Foundation containers.

Additional functions for managing dates and times are described in *Time Utilities Reference*; see also *AV Foundation Constants Reference*.

Functions by Task

CMTime Miscellaneous Functions

`CMTimeAbsoluteValue` (page 115)

`CMTimeAdd` (page 115)

`CMTimeCompare` (page 116)

CMTimeConvertScale (page 116)

CMTimeCopyAsDictionary (page 117)

CMTimeCopyDescription (page 117)

CMTimeGetSeconds (page 118)

CMTimeMake (page 118)

CMTimeMakeFromDictionary (page 118)

CMTimeMakeWithEpoch (page 119)

CMTimeMakeWithSeconds (page 119)

CMTimeMaximum (page 120)

CMTimeMinimum (page 120)

CMTimeMultiply (page 121)

CMTimeMultiplyByFloat64 (page 121)

CMTimeShow (page 122)

CMTimeSubtract (page 122)

CMTime Macros

CMTIME_COMPARE_INLINE (page 122)
: Returns a Boolean value that indicates whether the specified comparison of two CMTimes is true.

CMTIME_IS_VALID (page 125)
: Returns a Boolean value that indicates whether a given time is valid.

CMTIME_IS_INVALID (page 123)
: Returns a Boolean value that indicates whether a given time is invalid.

CMTIME_IS_POSITIVE_INFINITY (page 124)
: Returns a Boolean value that indicates whether a given time is positive infinity.

CMTIME_IS_NEGATIVE_INFINITY (page 124)
: Returns a Boolean value that indicates whether a given time is negative infinity.

CMTIME_IS_INDEFINITE (page 123)
: Returns a Boolean value that indicates whether a given time is indefinite.

`CMTIME_IS_NUMERIC` (page 124)
Returns a Boolean value that indicates whether a given time is numeric.

`CMTIME_HAS_BEEN_ROUNDED` (page 123)
Returns a Boolean value that indicates whether a given time has been rounded.

Functions

CMTimeAbsoluteValue

```
CMTime CMTimeAbsoluteValue (
   CMTime time
);
```

Parameters

time

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMTime.h

CMTimeAdd

```
CMTime CMTimeAdd (
   CMTime addend1,
   CMTime addend2
);
```

Parameters

addend1

addend2

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMTime.h

CMTimeCompare

```
int32_t CMTimeCompare (
   CMTime time1,
   CMTime time2
);
```

Parameters

time1

time2

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMTime.h

CMTimeConvertScale

```
CMTime CMTimeConvertScale (
   CMTime time,
   int32_t newTimescale,
   CMTimeRoundingMethod method
);
```

Parameters

time

newTimescale

method

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMTime.h

CMTimeCopyAsDictionary

```
CFDictionaryRef CMTimeCopyAsDictionary (
   CMTime time,
   CFAllocatorRef allocator
);
```

Parameters

time

allocator

Return Value

Discussion
For keys in the resulting dictionary, see "Dictionary Keys" (page 131).

Availability
Available in iOS 4.0 and later.

Declared In
CMTime.h

CMTimeCopyDescription

```
CFStringRef CMTimeCopyDescription (
   CFAllocatorRef allocator,
   CMTime time
);
```

Parameters

allocator

time

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMTime.h

CMTimeGetSeconds

```
Float64 CMTimeGetSeconds (
   CMTime time
);
```

Parameters

time

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMTime.h

CMTimeMake

```
CMTime CMTimeMake (
   int64_t value,
   int32_t timescale
);
```

Parameters

value

timescale

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMTime.h

CMTimeMakeFromDictionary

```
CMTime CMTimeMakeFromDictionary (
   CFDictionaryRef dict
);
```

Parameters

dict

Return Value

Discussion
For keys in the dictionary, see "Dictionary Keys" (page 131).

Availability
Available in iOS 4.0 and later.

Declared In
`CMTime.h`

CMTimeMakeWithEpoch

```
CMTime CMTimeMakeWithEpoch (
   int64_t value,
   int32_t timescale,
   int64_t epoch
);
```

Parameters

value

timescale

epoch

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
`CMTime.h`

CMTimeMakeWithSeconds

```
CMTime CMTimeMakeWithSeconds (
   Float64 seconds,
   int32_t preferredTimeScale
);
```

Parameters

seconds

preferredTimeScale

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
`CMTime.h`

CMTimeMaximum

```
CMTime CMTimeMaximum (
   CMTime time1,
   CMTime time2
);
```

Parameters

time1

time2

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMTime.h

CMTimeMinimum

```
CMTime CMTimeMinimum (
   CMTime time1,
   CMTime time2
);
```

Parameters

time1

time2

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMTime.h

CMTimeMultiply

```
CMTime CMTimeMultiply (
   CMTime time,
   int32_t multiplier
);
```

Parameters

time

multiplier

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMTime.h

CMTimeMultiplyByFloat64

```
CMTime CMTimeMultiplyByFloat64 (
   CMTime time,
   Float64 multiplier
);
```

Parameters

time

multiplier

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMTime.h

CMTimeShow

```
void CMTimeShow (
   CMTime time
);
```

Parameters

time

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMTime.h

CMTimeSubtract

```
CMTime CMTimeSubtract (
   CMTime minuend,
   CMTime subtrahend
);
```

Parameters

minuend

subtrahend

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMTime.h

CMTIME_COMPARE_INLINE

Returns a Boolean value that indicates whether the specified comparison of two CMTimes is true.

```
#define CMTIME_COMPARE_INLINE(time1, comparator, time2)
((Boolean)(CMTimeCompare(time1, time2) comparator 0))
```

Discussion
For example:

```
CMTIME_COMPARE_INLINE(time1, <=, time2)
```

will return true if time1 <= time2.

Availability
Available in iOS 4.0 and later.

Declared In
CMTime.h

CMTIME_HAS_BEEN_ROUNDED

Returns a Boolean value that indicates whether a given time has been rounded.

```
#define CMTIME_HAS_BEEN_ROUNDED(time) ((Boolean)(CMTIME_IS_NUMERIC(time) &&
(((time).flags & kCMTimeFlags_HasBeenRounded) != 0)))
```

Return Value
`true` if the CMTime has been rounded, otherwise `false` (the time is completely accurate).

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMTime.h

CMTIME_IS_INDEFINITE

Returns a Boolean value that indicates whether a given time is indefinite.

```
#define CMTIME_IS_INDEFINITE(time) ((Boolean)(CMTIME_IS_VALID(time) && (((time).flags
 & kCMTimeFlags_Indefinite) != 0)))
```

Return Value
`true` if the CMTime is indefinite, otherwise `false`.

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMTime.h

CMTIME_IS_INVALID

Returns a Boolean value that indicates whether a given time is invalid.

```
#define CMTIME_IS_INVALID(time) (! CMTIME_IS_VALID(time))
```

Return Value
`true` if the CMTime is invalid, otherwise `false`.

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
`CMTime.h`

CMTIME_IS_NEGATIVE_INFINITY

Returns a Boolean value that indicates whether a given time is negative infinity.

```
#define CMTIME_IS_NEGATIVE_INFINITY(time) ((Boolean)(CMTIME_IS_VALID(time) &&
(((time).flags & kCMTimeFlags_NegativeInfinity) != 0)))
```

Return Value
`true` if the CMTime is negative infinity, otherwise `false`.

Discussion
Use this instead of `(myTime == kCMTimeNegativeInfinity)`, since there are many CMTime structs that represent positive infinity. This is because the non-flags fields are ignored, so they can contain anything.

Availability
Available in iOS 4.0 and later.

Declared In
`CMTime.h`

CMTIME_IS_NUMERIC

Returns a Boolean value that indicates whether a given time is numeric.

```
#define CMTIME_IS_NUMERIC(time) ((Boolean)(((time).flags & (kCMTimeFlags_Valid |
kCMTimeFlags_ImpliedValueFlagsMask)) == kCMTimeFlags_Valid))
```

Return Value
`true` if the CMTime is numeric, otherwise `false`. Returns `false` if the CMTime is invalid, indefinite, or +/- infinity.

Discussion
A numeric time contains a usable value/timescale/epoch.

Availability
Available in iOS 4.0 and later.

Declared In
`CMTime.h`

CMTIME_IS_POSITIVE_INFINITY

Returns a Boolean value that indicates whether a given time is positive infinity.

```
#define CMTIME_IS_POSITIVE_INFINITY(time) ((Boolean)(CMTIME_IS_VALID(time) &&
(((time).flags & kCMTimeFlags_PositiveInfinity) != 0)))
```

Return Value
`true` if the CMTime is positive infinity, otherwise `false`.

Discussion
Use this instead of `(myTime == kCMTimePositiveInfinity)`, since there are many CMTime structs that represent positive infinity. This is because the non-flags fields are ignored, so they can contain anything.

Availability
Available in iOS 4.0 and later.

Declared In
`CMTime.h`

CMTIME_IS_VALID

Returns a Boolean value that indicates whether a given time is valid.

```
#define CMTIME_IS_VALID(time) ((Boolean)(((time).flags & kCMTimeFlags_Valid) != 0))
```

Return Value
`true` if the CMTime is valid, otherwise `false`.

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
`CMTime.h`

Data Types

CMTime

Defines a structure that represents a rational time value `int64`/`int32`.

```
typedef struct
{
    CMTimeValue    value;
    CMTimeScale    timescale;
    CMTimeFlags    flags;
    CMTimeEpoch    epoch;
} CMTime;
```

Fields

`value`
The value of the CMTime.
value/timescale = seconds.

`timescale`
The timescale of the CMTime.
value/timescale = seconds.

`flags`

A bitfield representing the flags set for the CMTime.

For example, `kCMTimeFlags_Valid`. See "CMTime Flags" (page 128) for possible values.

`epoch`

The epoch of the CMTime.

You use the epoch to differentiate between equal timestamps that are actually different because of looping, multi-item sequencing, and so on.

The epoch is used during comparison: greater epochs happen after lesser ones. Addition or subtraction is only possible within a single epoch, however, since the epoch length may be unknown or variable.

Availability
Available in iOS 4.0 and later.

Declared In
`CMTime.h`

CMTimeEpoch

The epoch to which a CMTime refers.

```
typedef int64_t CMTimeEpoch;
```

Discussion
The epoch is typically `0`, but you might use a different value, for example, in a loop.

Availability
Available in iOS 4.0 and later.

Declared In
`CMTime.h`

CMTimeFlags

A type to specify the flag bits for a CMTime.

```
typedef uint32_t CMTimeFlags;
```

Discussion
For possible values, see "CMTime Flags" (page 128).

Availability
Available in iOS 4.0 and later.

Declared In
`CMTime.h`

CMTimeRoundingMethod

Type for constants used to specify the rounding method to use when computing `time.value` during timescale conversions.

```
typedef uint32_t CMTimeRoundingMethod;
```

Discussion
For possible values, see "Rounding Methods" (page 130).

Availability
Available in iOS 4.0 and later.

Declared In
`CMTime.h`

CMTimeScale

Denominator of rational CMTime.

```
typedef int32_t CMTimeScale;
```

Discussion
Timescales must be positive.

Availability
Available in iOS 4.0 and later.

Declared In
`CMTime.h`

CMTimeValue

Numerator of rational CMTime.

```
typedef int64_t CMTimeValue;
```

Availability
Available in iOS 4.0 and later.

Declared In
`CMTime.h`

Constants

Time Constants

Constants to initialize CMTime structures.

```
const CMTime kCMTimeInvalid;
const CMTime kCMTimeIndefinite;
const CMTime kCMTimePositiveInfinity;
const CMTime kCMTimeNegativeInfinity;
const CMTime kCMTimeZero;
```

Constants

`kCMTimeInvalid`

Use this constant to initialize an invalid CMTime.

All fields are 0, so you can `calloc` or fill with `0`'s to make lots of them. Do not test against this using `(time == kCMTimeInvalid)`, there are many CMTimes other than this that are also invalid. Use `CMTIME_IS_INVALID(time)` instead.

Available in iOS 4.0 and later.

Declared in `CMTime.h`.

`kCMTimeIndefinite`

Use this constant to initialize an indefinite CMTime (for example, the duration of a live broadcast).

Do not test against this using `(time == kCMTimeIndefinite)`, there are many CMTimes other than this that are also indefinite. Use `CMTIME_IS_INDEFINITE(time)` instead.

Available in iOS 4.0 and later.

Declared in `CMTime.h`.

`kCMTimePositiveInfinity`

Use this constant to initialize a CMTime to +infinity.

Do not test against this using `(time == kCMTimePositiveInfinity)`, there are many CMTimes other than this that are also +infinity. Use `CMTIME_IS_POSITIVE_INFINITY(time)` instead.

Available in iOS 4.0 and later.

Declared in `CMTime.h`.

`kCMTimeNegativeInfinity`

Use this constant to initialize a CMTime to -infinity.

Do not test against this using `(time == kCMTimeNegativeInfinity)`, there are many CMTimes other than this that are also -infinity. Use `CMTIME_IS_NEGATIVE_INFINITY(time)` instead.

Available in iOS 4.0 and later.

Declared in `CMTime.h`.

`kCMTimeZero`

Use this constant to initialize a CMTime to 0.

Do not test against this using `(time == kCMTimeZero)`, there are many CMTimes other than this that are also 0. Use `CMTimeCompare(time, kCMTimeZero)` instead.

Available in iOS 4.0 and later.

Declared in `CMTime.h`.

CMTime Flags

Constants to specify flags for CMTime.

```
enum {
    kCMTimeFlags_Valid = 1UL<<0,
    kCMTimeFlags_HasBeenRounded = 1UL<<1,
    kCMTimeFlags_PositiveInfinity = 1UL<<2,
    kCMTimeFlags_NegativeInfinity = 1UL<<3,
    kCMTimeFlags_Indefinite = 1UL<<4,
    kCMTimeFlags_ImpliedValueFlagsMask = kCMTimeFlags_PositiveInfinity |
kCMTimeFlags_NegativeInfinity | kCMTimeFlags_Indefinite
};
```

Constants

`kCMTimeFlags_Valid`
Indicates that the time is valid.
Available in iOS 4.0 and later.
Declared in `CMTime.h`.

`kCMTimeFlags_HasBeenRounded`
Indicates that the time has been rounded.
Available in iOS 4.0 and later.
Declared in `CMTime.h`.

`kCMTimeFlags_PositiveInfinity`
Indicates that the time is +infinity.
Available in iOS 4.0 and later.
Declared in `CMTime.h`.

`kCMTimeFlags_NegativeInfinity`
Indicates that the time is -infinity.
Available in iOS 4.0 and later.
Declared in `CMTime.h`.

`kCMTimeFlags_Indefinite`
Indicates that the time is indefinite.
Available in iOS 4.0 and later.
Declared in `CMTime.h`.

`kCMTimeFlags_ImpliedValueFlagsMask`
Indicates that the time is +infinity, -infinity, or indefinite.
Available in iOS 4.0 and later.
Declared in `CMTime.h`.

Maximum Timescale

A constant to define the maximum timescale.

```
#define kCMTimeMaxTimescale 0x7fffffffL
```

Constants

`kCMTimeMaxTimescale`
The maximum timescale.
Available in iOS 4.0 and later.
Declared in `CMTime.h`.

Rounding Methods

Constants used to specify the rounding method to use when computing `time.value` during timescale conversions.

```
enum {
    kCMTimeRoundingMethod_RoundHalfAwayFromZero = 1,
    kCMTimeRoundingMethod_RoundTowardZero = 2,
    kCMTimeRoundingMethod_RoundAwayFromZero = 3,
    kCMTimeRoundingMethod_QuickTime = 4,
    kCMTimeRoundingMethod_RoundTowardPositiveInfinity = 5,
    kCMTimeRoundingMethod_RoundTowardNegativeInfinity = 6,

    kCMTimeRoundingMethod_Default = kCMTimeRoundingMethod_RoundHalfAwayFromZero
};
```

Constants

`kCMTimeRoundingMethod_RoundHalfAwayFromZero`
Round towards zero if abs(fraction) is < 0.5, away from 0 if abs(fraction) is >= 0.5.

Available in iOS 4.0 and later.

Declared in `CMTime.h`.

`kCMTimeRoundingMethod_RoundTowardZero`
Round towards zero if fraction is != 0.

Available in iOS 4.0 and later.

Declared in `CMTime.h`.

`kCMTimeRoundingMethod_RoundAwayFromZero`
Round away from zero if abs(fraction) is > 0.

Available in iOS 4.0 and later.

Declared in `CMTime.h`.

`kCMTimeRoundingMethod_QuickTime`
Use `kCMTimeRoundingMethod_RoundTowardZero` if converting from larger to smaller scale (that is, from more precision to less precision), but use `kCMTimeRoundingMethod_RoundAwayFromZero` if converting from smaller to larger scale (ie. from less precision to more precision).

Also, never round a negative number down to 0; always return the smallest magnitude negative CMTime in this case (-1/newTimescale).

Available in iOS 4.0 and later.

Declared in `CMTime.h`.

`kCMTimeRoundingMethod_RoundTowardPositiveInfinity`
Round towards +infinity if fraction is != 0.

Available in iOS 4.0 and later.

Declared in `CMTime.h`.

`kCMTimeRoundingMethod_RoundTowardNegativeInfinity`
Round towards -infinity if fraction is != 0.

Available in iOS 4.0 and later.

Declared in `CMTime.h`.

`kCMTimeRoundingMethod_Default`
Synonym for `kCMTimeRoundingMethod_RoundHalfAwayFromZero`.

Available in iOS 4.0 and later.

Declared in `CMTime.h`.

Dictionary Keys

Keys to access values of a CMTime represented by a CFDictionary.

```
const CFStringRef kCMTimeValueKey;
const CFStringRef kCMTimeScaleKey;
const CFStringRef kCMTimeEpochKey;
const CFStringRef kCMTimeFlagsKey;
```

Constants

`kCMTimeValueKey`
CFDictionary key for the value field of CMTime (a CFNumber containing an int64_t value).

Available in iOS 4.0 and later.

Declared in `CMTime.h`.

`kCMTimeScaleKey`
CFDictionary key for the timescale field of CMTime (a CFNumber containing an int32_t value).

Available in iOS 4.0 and later.

Declared in `CMTime.h`.

`kCMTimeEpochKey`
CFDictionary key for the epoch field of CMTime (a CFNumber containing an int64_t value).

Available in iOS 4.0 and later.

Declared in `CMTime.h`.

`kCMTimeFlagsKey`
CFDictionary key for the flags field of CMTime (a CFNumber containing an int32_t value).

Available in iOS 4.0 and later.

Declared in `CMTime.h`.

Discussion

See also `CMTimeCopyAsDictionary` (page 117) and `CMTimeMakeFromDictionary` (page 118).

CMTimeRange Reference

Derived From:	*CFType Reference*
Framework:	CoreMedia.framework
Declared in	CMTimeRange.h

Overview

This document describes the API for creating and manipulating CMTimeRange structures.

CMTimeRange structs are non-opaque mutable structures that represent time ranges. A CMTimeRange is represented as two CMTime structs, one that specifies the start time of the range and another that specifies the duration of the range. A time range does not include the time that is the start time plus the duration. In other words, the following expression

```
CMTimeRangeContainsTime(range, CMTimeRangeGetEnd(range))
```

always evaluates to false.

You can convert CMTimeRanges to CFDictionaries (see `CFDictionaryRef`) using `CMTimeRangeCopyAsDictionary` (page 137) and `CMTimeRangeMakeFromDictionary` (page 141), for use in annotations and various Core Foundation containers.

The epoch in a CMTime that represents a duration should always be 0, and the value must be non-negative. The epoch in a CMTime that represents a timestamp may be non-zero, but range operations (such as `CMTimeRangeGetUnion` (page 140)) can only be performed on ranges whose start fields have the same epoch. CMTimeRanges cannot span different epochs.

Additional functions for managing dates and times are described in *Time Utilities Reference*; see also *AV Foundation Constants Reference*.

Functions by Task

Miscellaneous Functions

`CMTimeClampToRange` (page 135)

`CMTimeMapDurationFromRangeToRange` (page 135)

Macros

Functions

CMTimeClampToRange

```
CMTime CMTimeClampToRange (
   CMTime time,
   CMTimeRange range
);
```

Parameters

time

range

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMTimeRange.h

CMTimeMapDurationFromRangeToRange

```
CMTime CMTimeMapDurationFromRangeToRange (
   CMTime dur,
   CMTimeRange fromRange,
   CMTimeRange toRange
);
```

Parameters

dur

fromRange

toRange

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMTimeRange.h

CMTimeMapTimeFromRangeToRange

```
CMTime CMTimeMapTimeFromRangeToRange (
   CMTime t,
   CMTimeRange fromRange,
   CMTimeRange toRange
);
```

Parameters

t

fromRange

toRange

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMTimeRange.h

CMTimeRangeContainsTime

```
Boolean CMTimeRangeContainsTime (
   CMTimeRange range,
   CMTime time
);
```

Parameters

range

time

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMTimeRange.h

CMTimeRangeContainsTimeRange

```
Boolean CMTimeRangeContainsTimeRange (
   CMTimeRange range1,
   CMTimeRange range2
);
```

Parameters

range1

range2

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMTimeRange.h

CMTimeRangeCopyAsDictionary

```
CFDictionaryRef CMTimeRangeCopyAsDictionary (
   CMTimeRange range,
   CFAllocatorRef allocator
);
```

Parameters

range

allocator

Return Value

Discussion
For keys, see "CFDictionary Keys" (page 144).

Availability
Available in iOS 4.0 and later.

Declared In
CMTimeRange.h

CMTimeRangeCopyDescription

```
CFStringRef CMTimeRangeCopyDescription (
   CFAllocatorRef allocator,
   CMTimeRange range
);
```

Parameters

allocator

range

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMTimeRange.h

CMTimeRangeEqual

```
Boolean CMTimeRangeEqual (
   CMTimeRange range1,
   CMTimeRange range2
);
```

Parameters

range1

range2

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMTimeRange.h

CMTimeRangeFromTimeToTime

```
CMTimeRange CMTimeRangeFromTimeToTime (
   CMTime start,
   CMTime end
);
```

Parameters

start

end

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMTimeRange.h

CMTimeRangeGetEnd

```
CMTime CMTimeRangeGetEnd (
   CMTimeRange range
);
```

Parameters

range

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMTimeRange.h

CMTimeRangeGetIntersection

```
CMTimeRange CMTimeRangeGetIntersection (
   CMTimeRange range1,
   CMTimeRange range2
);
```

Parameters

range1

range2

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMTimeRange.h

CMTimeRangeGetUnion

```
CMTimeRange CMTimeRangeGetUnion (
   CMTimeRange range1,
   CMTimeRange range2
);
```

Parameters

range1

range2

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMTimeRange.h

CMTimeRangeMake

```
CMTimeRange CMTimeRangeMake (
   CMTime start,
   CMTime duration
);
```

Parameters

start

duration

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMTimeRange.h

CMTimeRangeMakeFromDictionary

```
CMTimeRange CMTimeRangeMakeFromDictionary (
   CFDictionaryRef dict
);
```

Parameters

dict

Return Value

Discussion
For keys, see "CFDictionary Keys" (page 144).

Availability
Available in iOS 4.0 and later.

Declared In
CMTimeRange.h

CMTimeRangeShow

```
void CMTimeRangeShow (
   CMTimeRange range
);
```

Parameters

range

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
`CMTimeRange.h`

CMTIMERANGE_IS_EMPTY

Returns a Boolean value that indicates whether a given CMTimeRange has a duration of 0.

```
#define CMTIMERANGE_IS_EMPTY(range) ((Boolean)(CMTIMERANGE_IS_VALID(range) &&
(CMTIME_COMPARE_INLINE(range.duration, ==, kCMTimeZero))))
```

Parameters

range

A time range.

Return Value

`true` if *range* has a duration of `0`; otherwise, `false`.

Availability

Available in iOS 4.0 and later.

Declared In

`CMTimeRange.h`

CMTIMERANGE_IS_INDEFINITE

Returns a Boolean value that indicates whether a given CMTimeRange is indefinite.

```
#define CMTIMERANGE_IS_INDEFINITE(range) ((Boolean)(CMTIMERANGE_IS_VALID(range) &&
 (CMTIME_IS_INDEFINITE(range.start) || CMTIME_IS_INDEFINITE(range.duration))))
```

Parameters

range

A time range.

Return Value

`true` if *range* is indefinite; otherwise, `false`.

Availability

Available in iOS 4.0 and later.

Declared In

`CMTimeRange.h`

CMTIMERANGE_IS_INVALID

Returns a Boolean value that indicates whether a given CMTimeRange is invalid.

```
#define CMTIMERANGE_IS_INVALID(range) (! CMTIMERANGE_IS_VALID(range))
```

Parameters

range

A time range.

Return Value
`true` if *`range`* is invalid; otherwise, `false`.

Availability
Available in iOS 4.0 and later.

Declared In
`CMTimeRange.h`

CMTIMERANGE_IS_VALID

Returns a Boolean value that indicates whether a given CMTimeRange is valid.

```
#define CMTIMERANGE_IS_VALID(range) ((Boolean)(CMTIME_IS_VALID(range.start) &&
CMTIME_IS_VALID(range.duration) && (range.duration.epoch == 0) &&
(range.duration.value >= 0)))
```

Parameters

`range`

A time range.

Return Value
`true` if *`range`* is valid; otherwise, `false`.

Availability
Available in iOS 4.0 and later.

Declared In
`CMTimeRange.h`

Data Types

CMTimeMapping

A structure to specify the mapping of a segment of one time line into another.

```
typedef struct
{
    CMTimeRange source;
    CMTimeRange target;
} CMTimeMapping;
```

Fields

`source`

The time range on the source time line.

For an empty edit, `source.start` is an invalid CMTime, in which case source.duration is ignored. Otherwise, `source.start` is the starting time within the source, and `source.duration` is the duration of the source timeline to be mapped to the target time range.

`target`
The time range on the target time line.

If `target.duration` and `source.duration` are different, then the source segment should be played at the rate `source.duration /target.duration` to fit.

Discussion
A CMTimeMapping specifies the mapping of a segment of one time line (called the source) into another time line (called the target). When used for movie edit lists, the source time line is the media and the target time line is the track or movie.

Availability
Available in iOS 4.0 and later.

Declared In
`CMTimeRange.h`

CMTimeRange

A time range represented as two CMTime structures.

```
typedef struct
{
    CMTime    start;
    CMTime    duration;
} CMTimeRange;
```

Fields
`start`
The start time of the time range.

`duration`
The duration of the time range.

Availability
Available in iOS 4.0 and later.

Declared In
`CMTimeRange.h`

Constants

CFDictionary Keys

Keys for components in a CFDictionary representation of a CMTimeRange.

```
const CFStringRef kCMTimeRangeStartKey;
const CFStringRef kCMTimeRangeDurationKey;
```

Constants

`kCMTimeRangeStartKey`

The key for start field of a CMTimeRange (CMTime).

Available in iOS 4.0 and later.

Declared in `CMTimeRange.h`.

`kCMTimeRangeDurationKey`

The key for timescale of a CMTimeRange (CMTime).

Available in iOS 4.0 and later.

Declared in `CMTimeRange.h`.

Discussion

To convert a CMTimeRange to and from a CFDictionary, see `CMTimeRangeCopyAsDictionary` (page 137) and `CMTimeRangeMakeFromDictionary` (page 141).

Pre-Specified Time Ranges

These constants specify zero and invalid time ranges.

```
const CMTimeRange kCMTimeRangeZero;
const CMTimeRange kCMTimeRangeInvalid;
```

Constants

`kCMTimeRangeZero`

Use this constant to generate an empty CMTimeRange at `0`.

Available in iOS 4.0 and later.

Declared in `CMTimeRange.h`.

`kCMTimeRangeInvalid`

Use this constant to generate an invalid CMTimeRange.

Available in iOS 4.0 and later.

Declared in `CMTimeRange.h`.

Data Types

Core Media Framework Data Types Reference

Framework: CoreMedia.framework

Declared in CMBase.h

Overview

This document describes data types defined in the Core Media framework not described in individual references.

Data Types

CMItemCount

Data type for the item count.

```
typedef signed long    CMItemCount;
```

Availability
Available in iOS 4.0 and later.

Declared In
`CMBase.h`

CMItemCount

Data type for the item index.

```
typedef signed long    CMItemIndex;
```

Availability
Available in iOS 4.0 and later.

Declared In
`CMBase.h`

CMItemCount

Data type for the persistent track ID.

```
typedef int32_t CMPersistentTrackID;
```

Availability
Available in iOS 4.0 and later.

Declared In
`CMBase.h`

Constants

Core Media Constants Reference

Framework: CoreMedia.framework
Declared in CMBase.h

Overview

This document describes constants defined in the Core Media framework not described in individual references.

Constants

Invalid Track ID Specifier

Constant to indicate an invalid track ID.

```
enum {
    kCMPersistentTrackID_Invalid = 0
};
```

Constants

`kCMPersistentTrackID_Invalid`
Indicates an invalid track ID.

Available in iOS 4.0 and later.

Declared in `CMBase.h`.

Other References

PART IV

Other References

CMAttachment Reference

Framework: CoreMedia.framework

Declared in CMAttachment.h

Overview

This document describes the Core Media attachment protocol.

Functions

CMCopyDictionaryOfAttachments

```
CFDictionaryRef CMCopyDictionaryOfAttachments (
   CFAllocatorRef allocator,
   CMAttachmentBearerRef target,
   CMAttachmentMode attachmentMode
);
```

Parameters

allocator

target

attachmentMode

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMAttachment.h

CMGetAttachment

```
CFTypeRef CMGetAttachment (
   CMAttachmentBearerRef target,
   CFStringRef key,
   CMAttachmentMode *attachmentModeOut
);
```

Parameters

target

key

attachmentModeOut

Return Value

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMAttachment.h

CMPropagateAttachments

```
void CMPropagateAttachments (
   CMAttachmentBearerRef source,
   CMAttachmentBearerRef destination
);
```

Parameters

source

destination

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMAttachment.h

CMRemoveAllAttachments

```
void CMRemoveAllAttachments (
   CMAttachmentBearerRef target
);
```

Parameters

target

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMAttachment.h

CMRemoveAttachment

```
void CMRemoveAttachment (
   CMAttachmentBearerRef target,
   CFStringRef key
);
```

Parameters

target

key

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMAttachment.h

CMSetAttachment

```
void CMSetAttachment (
   CMAttachmentBearerRef target,
   CFStringRef key,
   CFTypeRef value,
   CMAttachmentMode attachmentMode
);
```

Parameters

target

key

value

attachmentMode

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMAttachment.h

CMSetAttachments

```
void CMSetAttachments (
   CMAttachmentBearerRef target,
   CFDictionaryRef theAttachments,
   CMAttachmentMode attachmentMode
);
```

Parameters

target

theAttachments

attachmentMode

Discussion

Availability
Available in iOS 4.0 and later.

Declared In
CMAttachment.h

Data Types

CMAttachmentBearerRef

Type for objects that can bear attachments.

```
typedef CFTypeRef CMAttachmentBearerRef;
```

Discussion
A CMAttachmentBearer is a Core Foundation-based object that supports the suite of key/value/mode attachment APIs. Since "plain" C has no type subclassing, CFType is used as the basis for the CMAttachmentBearer type. (Not all CFTypes support CMAttachmentBearer methods; if a CMAttachmentBearer method is called on a CF object that does not support it, it will fail.)

Availability
Available in iOS 4.0 and later.

Declared In
`CMAttachment.h`

CMAttachmentMode

Type to specify attachment modes.

```
typedef uint32_t CMAttachmentMode;
```

Discussion
For possible values, see "Attachment Propagation" (page 161).

Availability
Available in iOS 4.0 and later.

Declared In
`CMAttachment.h`

Constants

Attachment Propagation

Constants to specify whether attachments should propagate.

```
enum {
    kCMAttachmentMode_ShouldNotPropagate    = 0,
    kCMAttachmentMode_ShouldPropagate       = 1
};
```

Constants

`kCMAttachmentMode_ShouldNotPropagate`
Indicates that attachments should not propagate.
Available in iOS 4.0 and later.
Declared in `CMAttachment.h`.

`kCMAttachmentMode_ShouldPropagate`
Indicates that attachments should propagate.
Available in iOS 4.0 and later.
Declared in `CMAttachment.h`.

Document Revision History

This table describes the changes to *Core Media Framework Reference*.

Date	Notes
2010-03-23	TBD

Document Revision History