Core Audio Framework Reference

Audio & Video: Audio

2008-07-08

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

Simultaneously published in the United States and Canada.

Contents

Core Audio Framework Reference

Framework:	/System/Library/Frameworks/CoreAudio.framework
Declared in	CoreAudioTypes.h

The Core Audio framework (which is not an umbrella framework for the other services in Core Audio, but rather a peer) declares data types and constants used by other Core Audio interfaces. This framework also includes a handful of convenience functions.

Data Types

PART I

Data Types

Core Audio Data Types Reference

Framework: CoreAudio/CoreAudio.h

Declared in

Overview

This document describes data types and constants used throughout Core Audio, as well as some convenience functions for working with these types and constants.

If you are unfamiliar with the specialized terminology used when manipulating audio data, refer to *Core Audio Glossary*.

Functions by Task

Testing for Native Endian Linear PCM Data

`TestAudioFormatNativeEndian` (page 15)
A macro for checking if an `AudioFormatBasicDescription` structure indicates native endian linear PCM data.

`IsAudioFormatNativeEndian` (page 15)
A C++ inline function for checking if an `AudioFormatBasicDescription` structure indicates native-endian linear PCM data.

Getting the Number of Channels From a Layout Tag

`AudioChannelLayoutTag_GetNumberOfChannels` (page 10)
A macro to get the number of channels from an audio channel layout tag (`AudioChannelLayoutTag` data type).

Helper Functions for Filling out Core Audio Data Structures

`CalculateLPCMFlags` (page 10)
A C++ inline function for calculating the value for the audio stream basic description `mFormatFlags` field for linear PCM data.

`FillOutASBDForLPCM` (page 11)
A C++ inline function for filling out an `AudioStreamBasicDescription` to describe linear PCM data.

`FillOutAudioTimeStampWithHostTime` (page 13)
A C++ inline function for filling out an `AudioTimeStamp` structure with a host time.

`FillOutAudioTimeStampWithSampleTime` (page 14)
A C++ inline function for filling out an `AudioTimeStamp` structure with a sample time.

`FillOutAudioTimeStampWithSampleAndHostTime` (page 13)
A C++ inline function for filling out an `AudioTimeStamp` structure with a sample time and a host time.

Functions

AudioChannelLayoutTag_GetNumberOfChannels

A macro to get the number of channels from an audio channel layout tag (`AudioChannelLayoutTag` data type).

```
#define AudioChannelLayoutTag_GetNumberOfChannels(layoutTag)
                ((UInt32)((layoutTag) & 0x0000FFFF))
```

Parameters

`layoutTag`
The audio channel layout tag to examine.

Return Value

The number of channels the tag indicates.

Discussion

The low 16 bits of an audio channel layout tag gives the number of channels, unless the layout tag is `kAudioChannelLayoutTag_UseChannelDescriptions` or `kAudioChannelLayoutTag_UseChannelBitmap`, which specify other ways of defining the layout.

Availability

Available in iOS 2.0 and later.

Declared In

`CoreAudioTypes.h`

CalculateLPCMFlags

A C++ inline function for calculating the value for the audio stream basic description `mFormatFlags` field for linear PCM data.

```
#if defined(__cplusplus)
inline UInt32 CalculateLPCMFlags (
    UInt32 inValidBitsPerChannel,
    UInt32 inTotalBitsPerChannel,
    bool inIsFloat,
    bool inIsBigEndian,
    bool inIsNonInterleaved = false
) {
    return
        (inIsFloat ? kAudioFormatFlagIsFloat : kAudioFormatFlagIsSignedInteger) |
        (inIsBigEndian ? ((UInt32)kAudioFormatFlagIsBigEndian) : 0)             |
        ((!inIsFloat && (inValidBitsPerChannel == inTotalBitsPerChannel)) ?
            kAudioFormatFlagIsPacked : kAudioFormatFlagIsAlignedHigh)           |
        (inIsNonInterleaved ? ((UInt32)kAudioFormatFlagIsNonInterleaved) : 0);
}
#endif
```

Parameters

inValidBitsPerChannel

The number of valid bits in each sample.

inTotalBitsPerChannel

The total number of bits in each sample.

inIsFloat

Use `true` if the samples are represented with floating point numbers.

inIsBigEndian

Use `true` if the samples are big endian.

inIsNonInterleaved

Use `true` if the samples are noninterleaved.

Return Value

A `UInt32` value containing the calculated format flags.

Discussion

This function does not support specifying sample formats that are either unsigned integer or low-aligned.

Availability

Available in iOS 2.0 and later.

Declared In

`CoreAudioTypes.h`

FillOutASBDForLPCM

A C++ inline function for filling out an `AudioStreamBasicDescription` to describe linear PCM data.

```
#if defined(__cplusplus)
inline void FillOutASBDForLPCM (
    AudioStreamBasicDescription &outASBD,
    Float64 inSampleRate,
    UInt32 inChannelsPerFrame,
    UInt32 inValidBitsPerChannel,
    UInt32 inTotalBitsPerChannel,
    bool inIsFloat,
    bool inIsBigEndian,
    bool inIsNonInterleaved = false
) {
    outASBD.mSampleRate = inSampleRate;
    outASBD.mFormatID = kAudioFormatLinearPCM;
    outASBD.mFormatFlags =    CalculateLPCMFlags (
                                inValidBitsPerChannel,
                                inTotalBitsPerChannel,
                                inIsFloat,
                                inIsBigEndian,
                                inIsNonInterleaved
                            );
    outASBD.mBytesPerPacket =
       (inIsNonInterleaved ? 1 : inChannelsPerFrame) * (inTotalBitsPerChannel/8);
    outASBD.mFramesPerPacket = 1;
    outASBD.mBytesPerFrame =
       (inIsNonInterleaved ? 1 : inChannelsPerFrame) * (inTotalBitsPerChannel/8);
    outASBD.mChannelsPerFrame = inChannelsPerFrame;
    outASBD.mBitsPerChannel = inValidBitsPerChannel;
}
#endif
```

Parameters

outASBD

On output, a filled-out `AudioStreamBasicDescription` structure.

inSampleRate

The number of sample frames per second of the data in the stream.

inChannelsPerFrame

The number of channels in each frame of data.

inValidBitsPerChannel

The number of valid bits in each sample.

inTotalBitsPerChannel

The total number of bits in each sample.

inIsFloat

Use `true` if the samples are represented as floating-point numbers.

inIsBigEndian

Use `true` if the samples are big endian.

inIsNonInterleaved

Use `true` if the samples are noninterleaved.

Discussion

This function does not support specifying sample formats that are either unsigned integer or low-aligned.

Availability

Available in iOS 2.0 and later.

Declared In
`CoreAudioTypes.h`

FillOutAudioTimeStampWithHostTime

A C++ inline function for filling out an `AudioTimeStamp` structure with a host time.

```
#if defined(__cplusplus)
inline void FillOutAudioTimeStampWithHostTime (
    AudioTimeStamp &outATS,
    UInt64 inHostTime
) {
    outATS.mSampleTime = 0;
    outATS.mHostTime = inHostTime;
    outATS.mRateScalar = 0;
    outATS.mWordClockTime = 0;
    memset (&outATS.mSMPTETime, 0, sizeof (SMPTETime));
    outATS.mFlags = kAudioTimeStampHostTimeValid;
}
#endif
```

Parameters

outATS
: On output, a filled-out `AudioTimeStamp` structure.

inHostTime
: The host time to assign to the audio timestamp.

Availability
Available in iOS 2.0 and later.

Declared In
`CoreAudioTypes.h`

FillOutAudioTimeStampWithSampleAndHostTime

A C++ inline function for filling out an `AudioTimeStamp` structure with a sample time and a host time.

```
#if defined(__cplusplus)
inline void FillOutAudioTimeStampWithSampleAndHostTime (
    AudioTimeStamp &outATS,
    Float64 inSampleTime,
    UInt64 inHostTime
) {
    outATS.mSampleTime = inSampleTime;
    outATS.mHostTime = inHostTime;
    outATS.mRateScalar = 0;
    outATS.mWordClockTime = 0;
    memset (&outATS.mSMPTETime, 0, sizeof (SMPTETime));
    outATS.mFlags = kAudioTimeStampSampleTimeValid |
                        kAudioTimeStampHostTimeValid;
}
#endif
```

Parameters

outATS

On output, a filled-out `AudioTimeStamp` structure.

inSampleTime

The sample time to assign to the audio timestamp.

inHostTime

The host time to assign to the audio timestamp.

Availability
Available in iOS 2.0 and later.

Declared In
`CoreAudioTypes.h`

FillOutAudioTimeStampWithSampleTime

A C++ inline function for filling out an `AudioTimeStamp` structure with a sample time.

```
#if defined(__cplusplus)
inline void FillOutAudioTimeStampWithSampleTime (
    AudioTimeStamp &outATS,
    Float64 inSampleTime
) {
    outATS.mSampleTime = inSampleTime;
    outATS.mHostTime = 0;
    outATS.mRateScalar = 0;
    outATS.mWordClockTime = 0;
    memset (&outATS.mSMPTETime, 0, sizeof (SMPTETime));
    outATS.mFlags = kAudioTimeStampSampleTimeValid;
}
#endif
```

Parameters

outATS

On output, a filled-out `AudioTimeStamp` structure.

inSampleTime

The sample time to assign to the audio timestamp.

Availability
Available in iOS 2.0 and later.

Declared In
`CoreAudioTypes.h`

IsAudioFormatNativeEndian

A C++ inline function for checking if an `AudioFormatBasicDescription` structure indicates native-endian linear PCM data.

```
#if defined(__cplusplus)
inline bool IsAudioFormatNativeEndian (
    const AudioStreamBasicDescription &f
) {
    return  (f.mFormatID == kAudioFormatLinearPCM) &&
            ((f.mFormatFlags & kAudioFormatFlagIsBigEndian) ==
                                    kAudioFormatFlagsNativeEndian);
}
#endif
```

Parameters

f

The `AudioFormatBasicDescription` structure you want to examine.

Return Value
A Boolean value indicating whether the `AudioFormatBasicDescription` structure specifies native endian linear PCM data, `true` if the data is linear PCM and is native endian.

Availability
Available in iOS 2.0 and later.

Declared In
`CoreAudioTypes.h`

TestAudioFormatNativeEndian

A macro for checking if an `AudioFormatBasicDescription` structure indicates native endian linear PCM data.

```
#define TestAudioFormatNativeEndian (f)  (
        (f.mFormatID == kAudioFormatLinearPCM) &&
        ((f.mFormatFlags & kAudioFormatFlagIsBigEndian) ==
                                kAudioFormatFlagsNativeEndian)
)
```

Parameters

f

The `AudioFormatBasicDescription` structure you want to examine.

Return Value
`True` if the data is linear PCM and is native endian.

Availability
Available in iOS 2.0 and later.

Declared In
`CoreAudioTypes.h`

Data Types

AudioBuffer

Holds and describes a buffer of audio data.

```
struct AudioBuffer {
    UInt32 mNumberChannels;
    UInt32 mDataByteSize;
    void   *mData;
};
typedef struct AudioBuffer AudioBuffer;
```

Fields

`mNumberChannels`
The number of interleaved channels in the buffer. If the number is `1`, the buffer is noninterleaved.

`mDataByteSize`
The number of bytes in the buffer pointed at by the `mData` field.

`mData`
A pointer to a buffer of audio data.

Discussion
An `AudioBuffer` structure holds a single buffer of audio data in its `mData` field. The buffer can represent two different sorts of audio:

- A single, monophonic, noninterleaved channel of audio
- Interleaved audio with any number of channels—as designated by the `mNumberChannels` field

Noninterleaved formats are used primarily by audio units and audio converters.

Availability
Available in iOS 2.0 and later.

Declared In
`CoreAudioTypes.h`

AudioBufferList

Holds a variable-length array of `AudioBuffer` structures.

```
struct AudioBufferList {
    UInt32      mNumberBuffers;
    AudioBuffer mBuffers[1];
};
typedef struct AudioBufferList AudioBufferList;
```

Fields

`mNumberBuffers`

The number of `AudioBuffer` structures in the `mBuffers` array.

`mBuffers`

A variable length array of `AudioBuffer` structures.

Discussion

The `AudioBufferList` structure provides a mechanism for encapsulating one or more buffers of audio data. It is used by functions in various Core Audio APIs, as described in *Audio Converter Services Reference*, *Audio Unit Component Services Reference*, and *Extended Audio File Services Reference*.

Availability

Available in iOS 2.0 and later.

Declared In

`CoreAudioTypes.h`

AudioChannelDescription

Describes an audio data channel.

```
struct AudioChannelDescription {
    AudioChannelLabel   mChannelLabel;
    UInt32              mChannelFlags;
    Float32             mCoordinates[3];
};
typedef struct AudioChannelDescription AudioChannelDescription;
```

Fields

`mChannelLabel`

The `AudioChannelLabel` structure that describes the channel.

`mChannelFlags`

Flags that control the interpretation of `mCoordinates`. See "`Channel Coordinate Flags`" (page 51) for possible values.

`mCoordinates`

An ordered triple that specifies a precise speaker location. See "`Channel Coordinate Index Constants`" (page 51) for the interpretation of the items in the array.

Availability

Available in iOS 2.0 and later.

Declared In

`CoreAudioTypes.h`

AudioChannelLabel

Identifies how an audio data channel is to be used.

```
typedef UInt32 AudioChannelLabel;
```

Discussion
This data type is used for the `mChannelLabel` field of the `AudioChannelDescription` (page 17) structure. See "Audio Channel Label Constants" (page 40) for possible values.

Availability
Available in iOS 2.0 and later.

Declared In
`CoreAudioTypes.h`

AudioChannelLayout

Specifies a channel layout in a file or in hardware.

```
struct AudioChannelLayout {
    AudioChannelLayoutTag     mChannelLayoutTag;
    UInt32                    mChannelBitmap;
    UInt32                    mNumberChannelDescriptions;
    AudioChannelDescription   mChannelDescriptions[1];
};
typedef struct AudioChannelLayout AudioChannelLayout;
```

Fields

`mChannelLayoutTag`
: The `AudioChannelLayoutTag` value that indicates the layout. See "Audio Channel Layout Tags" (page 52) for possible values.

`mChannelBitmap`
: If `mChannelLayoutTag` is set to `kAudioChannelLayoutTag_UseChannelBitmap`, this field is the channel-use bitmap.

`mNumberChannelDescriptions`
: The number of items in the `mChannelDescriptions` array.

`mChannelDescriptions`
: A variable length array of `mNumberChannelDescription` elements that describes a layout. If the `mChannelLayoutTag` field is set to `kAudioChannelLayoutTag_UseChannelDescriptions`, use this field to describe the layout.

Availability
Available in iOS 2.0 and later.

Declared In
`CoreAudioTypes.h`

AudioChannelLayoutTag

Identifies a previously-defined channel layout.

```
typedef UInt32 AudioChannelLayoutTag;
```

Discussion
This data type is used for the `mChannelLayoutTag` field of the `AudioChannelLayout` (page 18) structure. See "Audio Channel Layout Tags" (page 52) for possible values.

Availability
Available in iOS 2.0 and later.

Declared In
`CoreAudioTypes.h`

AudioClassDescription

Describes an installed codec.

```
struct AudioClassDescription {
    OSType  mType;
    OSType  mSubType;
    OSType  mManufacturer;
};
typedef struct AudioClassDescription    AudioClassDescription;
```

Fields

`mType`
The four character code for the codec type. Defined by the codec manufacturer.

`mSubType`
The four character code for the codec subtype. Defined by the codec manufacturer.

`mManufacturer`
The four character code for the codec manufacturer. This must be a unique code registered with Apple.

Availability
Available in iOS 2.0 and later.

Declared In
`CoreAudioTypes.h`

AudioSampleType

The canonical audio data sample type for input and output.

```
typedef SInt16 AudioSampleType;
```

Discussion
The canonical audio sample type for input and output in iOS is linear PCM with 16-bit integer samples.

Availability
Available in iOS 2.0 and later.

Declared In
`CoreAudioTypes.h`

AudioUnitSampleType

The canonical audio data sample type for audio processing.

```
typedef SInt32 AudioUnitSampleType;
#define kAudioUnitSampleFractionBits 24
```

Discussion
The canonical audio sample type for audio units and other audio processing in iOS is noninterleaved linear PCM with 8.24-bit fixed-point samples.

Availability
Available in iOS 2.0 and later.

Declared In
`CoreAudioTypes.h`

AudioStreamBasicDescription

An audio data format specification for a stream of audio.

```
struct AudioStreamBasicDescription {
    Float64 mSampleRate;
    UInt32  mFormatID;
    UInt32  mFormatFlags;
    UInt32  mBytesPerPacket;
    UInt32  mFramesPerPacket;
    UInt32  mBytesPerFrame;
    UInt32  mChannelsPerFrame;
    UInt32  mBitsPerChannel;
    UInt32  mReserved;
};
typedef struct AudioStreamBasicDescription  AudioStreamBasicDescription;
```

Fields

`mSampleRate`

The number of frames per second of the data in the stream, when the stream is played at normal speed. For compressed formats, this field indicates the number of frames per second of equivalent decompressed data.

The `mSampleRate` field must be nonzero, except when this structure is used in a listing of supported formats (see "kAudioStreamAnyRate" (page 26)).

`mFormatID`

An identifier specifying the general audio data format in the stream. See "Audio Data Format Identifiers" (page 26). This value must be nonzero.

`mFormatFlags`

Format-specific flags to specify details of the format. May be set to `0` to indicate no format flags. See "Audio Data Format Identifiers" (page 26) for the flags that apply to each format.

`mBytesPerPacket`

The number of bytes in a packet of audio data. To indicate variable packet size, set this field to `0`. For a format that uses variable packet size, specify the size of each packet using an `AudioStreamPacketDescription` (page 22) structure.

`mFramesPerPacket`

The number of frames in a packet of audio data. For uncompressed audio, the value is 1. For variable bit-rate formats, the value is a larger fixed number, such as 1024 for AAC. For formats with a variable number of frames per packet, such as Ogg Vorbis, set this field to `0`.

`mBytesPerFrame`

The number of bytes from the start of one frame to the start of the next frame in an audio buffer. Set this field to `0` for compressed formats.

For an audio buffer containing **interleaved** data for *n* channels, with each sample of type `AudioSampleType`, calculate the value for this field as follows:

```
mBytesPerFrame = n * sizeof (AudioSampleType);
```

For an audio buffer containing **noninterleaved** (monophonic) data, also using `AudioSampleType` samples, calculate the value for this field as follows:

```
mBytesPerFrame = sizeof (AudioSampleType);
```

`mChannelsPerFrame`

The number of channels in each frame of audio data. This value must be nonzero.

`mBitsPerChannel`

The number of bits for one audio sample. For example, for linear PCM audio using the `kAudioFormatFlagsCanonical` (page 35) format flags, calculate the value for this field as follows:

```
mBitsPerChannel = 8 * sizeof (AudioSampleType);
```

Set this field to `0` for compressed formats.

`mReserved`

Pads the structure out to force an even 8-byte alignment. Must be set to `0`.

Discussion

You can configure an audio stream basic description (ASBD) to specify a linear PCM format or a constant bit rate (CBR) format that has channels of equal size. For variable bit rate (VBR) audio, and for CBR audio where the channels have unequal sizes, each packet must additionally be described by an `AudioStreamPacketDescription` (page 22) structure.

A field value of `0` indicates that the value is either unknown or not applicable to the format.

Always initialize the fields of a new audio stream basic description structure to zero, as shown here:

```
AudioStreamBasicDescription myAudioDataFormat = {0};
```

To determine the duration represented by one packet, use the `mSampleRate` field with the `mFramesPerPacket` field, as follows:

```
duration = (1 / mSampleRate) * mFramesPerPacket
```

In Core Audio, the following definitions apply:

- An **audio stream** is a continuous series of data that represents a sound, such as a song.
- A **channel** is a discrete track of monophonic audio. A monophonic stream has one channel; a stereo stream has two channels.
- A **sample** is single numerical value for a single audio channel in an audio stream.
- A **frame** is a collection of time-coincident samples. For instance, a linear PCM stereo sound file has two samples per frame, one for the left channel and one for the right channel.
- A **packet** is a collection of one or more contiguous frames. A packet defines the smallest meaningful set of frames for a given audio data format, and is the smallest data unit for which time can be measured. In linear PCM audio, a packet holds a single frame. In compressed formats, it typically holds more; in some formats, the number of frames per packet varies.

- The **sample rate** for a stream is the number of frames per second of uncompressed (or, for compressed formats, the equivalent in decompressed) audio.

Availability
Available in iOS 2.0 and later.

Declared In
`CoreAudioTypes.h`

AudioStreamPacketDescription

Describes one packet in a buffer of audio data where the sizes of the packets differ or where there is non-audio data between audio packets.

```
struct AudioStreamPacketDescription {
    SInt64  mStartOffset;
    UInt32  mVariableFramesInPacket;
    UInt32  mDataByteSize;
};
typedef struct AudioStreamPacketDescription AudioStreamPacketDescription;
```

Fields

`mStartOffset`
: The number of bytes from the start of the buffer to the beginning of the packet. For example, if the data buffer contains 5 bytes of data, with one byte per packet, then `mStartOffset` for the last packet is 4 (that is, there are 4 bytes in the buffer before the start of the last packet.

`mVariableFramesInPacket`
: The number of sample frames of data in the packet. For formats with a constant number of frames per packet, this field is set to `0`.

`mDataByteSize`
: The number of bytes in the packet.

Discussion
For data formats where the packet size is not constant, such as variable bit rate data and data where the channels have unequal sizes, this structure is used to supplement the information in the `AudioStreamBasicDescription` (page 20) structure.

Availability
Available in iOS 2.0 and later.

Declared In
`CoreAudioTypes.h`

AudioTimeStamp

Holds multiple representations of a time stamp.

```
struct AudioTimeStamp {
    Float64         mSampleTime;
    UInt64          mHostTime;
    Float64         mRateScalar;
    UInt64          mWordClockTime;
    SMPTETime       mSMPTETime;
    UInt32          mFlags;
    UInt32          mReserved;
};
typedef struct AudioTimeStamp AudioTimeStamp;
```

Fields

`mSampleTime`
The absolute sample frame time.

`mHostTime`
The host machine's time base (see `CoreAudio/HostTime.h`).

`mRateScalar`
The ratio of actual host ticks per sample frame to the nominal host ticks per sample frame.

`mWordClockTime`
The word clock time.

`mSMPTETime`
The SMPTE time (see `SMPTETime` (page 24)).

`mFlags`
A set of flags indicating which representations of the time are valid; see "`Audio Time Stamp Flags`" (page 39) and "`Audio Time Stamp Flag Combination Constant`" (page 40).

`mReserved`
Pads the structure out to force an even 8-byte alignment.

Availability
Available in iOS 2.0 and later.

Declared In
`CoreAudioTypes.h`

AudioValueRange

Holds a pair of numbers that represent a continuous range of values.

```
struct AudioValueRange {
    Float64 mMinimum;
    Float64 mMaximum;
};
typedef struct AudioValueRange AudioValueRange;
```

Fields

`mMinimum`
The minimum value.

`mMaximum`
The maximum value.

Availability
Available in iOS 2.0 and later.

Declared In
`CoreAudioTypes.h`

AudioValueTranslation

Holds buffers used in translation operations.

```
struct AudioValueTranslation {
    void*   mInputData;
    UInt32  mInputDataSize;
    void*   mOutputData;
    UInt32  mOutputDataSize;
};
typedef struct AudioValueTranslation AudioValueTranslation;
```

Fields

`mInputData`
The buffer containing the data to be translated.

`mInputDataSize`
The number of bytes in the buffer pointed at by `mInputData`.

`mOutputData`
The buffer to hold the result of the translation.

`mOutputDataSize`
The number of bytes in the buffer pointed at by `mOutputData`.

Availability
Available in iOS 2.0 and later.

Declared In
`CoreAudioTypes.h`

SMPTETime

Specifies a time stamp as one of the SMPTE time types.

```
struct SMPTETime {
    SInt16  mSubframes;
    SInt16  mSubframeDivisor;
    UInt32  mCounter;
    UInt32  mType;
    UInt32  mFlags;
    SInt16  mHours;
    SInt16  mMinutes;
    SInt16  mSeconds;
    SInt16  mFrames;
};
typedef struct SMPTETime SMPTETime;
```

Fields

`mSubframes`
A subframe offset to the HH:MM:SS:FF time. You can use this field to position a time marker somewhere within the time span represented by a video frame, if necessary.

`mSubframeDivisor`
The number of subframes per video frame (typically 80).

`mCounter`
The total number of messages received. It takes 8 messages to carry a full SMPTE time code.

`mType`
A SMPTE time type constant indicating the kind of SMPTE time used (see "SMPTE Timecode Type Constants" (page 37)).

`mFlags`
A set of flags that indicate the SMPTE state (see "SMPTE State Flags" (page 39)).

`mHours`
The value of the hours portion of the SMPTE time.

`mMinutes`
The value of the minutes portion of the SMPTE time.

`mSeconds`
The value of the seconds portion of the SMPTE time.

`mFrames`
The value of the frames portion of the SMPTE time.

Discussion
SMPTE (Society of Motion Picture and Television Engineers, pronounced "SIMPtee") times are used to correlate a point in an audio stream with an external event. For example, a SMPTE time can be used to correlate a sound in an audio file with a video frame in a movie file.

Note that the frames referred to by this structure are video frames, where a video frame is a single complete image. (Compare with the definition of audio frames in the discussion for `AudioStreamBasicDescription` (page 20).)

A complete SMPTE time description takes 80 bits, including 32 user bits that contain vendor-specific information. The actual time-code portion of the SMPTE time description is normally sent in several messages, each message containing a portion of the time code. (The user bits are sent in a separate message.) Typically, the SMPTE time description is divided up into 8 1-byte messages, with the first nibble of each message specifying which portion of the time code is contained in the message and the second nibble containing the time information. Four such messages are normally sent with each video frame.

Video data contains somewhere from 24 to 60 frames per second (as specified by the SMPTE time type—see "SMPTE Timecode Type Constants" (page 37)) and each video frame has an associated SMPTE time. SMPTE time is based on a 24-hour clock. Each frame's SMPTE time consists of an hour, minute, and second value, plus the number of the frame within the second. Because audio data is sampled at a much higher rate (MP3 data is sampled at over 100,000 bits per second, for example), it is frequently desirable to correlate the audio data with a time within the persistence period of a single video frame. For this reason, the time period during which a single video frame is displayed is subdivided into subframes (typically 80 or 100 subframes per frame, as specified by the `mSubFrameDivisor` field). The `mSubFrames` field specifies the number of subframes into the video frame represented by this time structure.

Availability
Available in iOS 2.0 and later.

Declared In
`CoreAudioTypes.h`

Constants

Sample Type Constants

Constants used for specifying audio format flags

```
#define kAudioUnitSampleFractionBits 24
```

Constants

`kAudioUnitSampleFractionBits`

The number of fractional bits in fixed-point samples.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

kAudioStreamAnyRate

Indicates that an audio format can use any sample rate.

```
enum {
    kAudioStreamAnyRate = 0
};
```

Constants

`kAudioStreamAnyRate`

This constant can appear only in listings of supported formats. It can never be used as part of the description of an `AudioStreamBasicDescription` structure that is used for transporting or processing audio.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

Audio Data Format Identifiers

Identifiers for audio data formats, used in the `AudioStreamBasicDescription` structure.

```
enum {
    kAudioFormatLinearPCM               = 'lpcm',
    kAudioFormatAC3                     = 'ac-3',
    kAudioFormat60958AC3                = 'cac3',
    kAudioFormatAppleIMA4               = 'ima4',
    kAudioFormatMPEG4AAC                = 'aac ',
    kAudioFormatMPEG4CELP               = 'celp',
    kAudioFormatMPEG4HVXC               = 'hvxc',
    kAudioFormatMPEG4TwinVQ             = 'twvq',
    kAudioFormatMACE3                   = 'MAC3',
    kAudioFormatMACE6                   = 'MAC6',
    kAudioFormatULaw                    = 'ulaw',
    kAudioFormatALaw                    = 'alaw',
    kAudioFormatQDesign                 = 'QDMC',
    kAudioFormatQDesign2                = 'QDM2',
    kAudioFormatQUALCOMM                = 'Qclp',
    kAudioFormatMPEGLayer1              = '.mp1',
    kAudioFormatMPEGLayer2              = '.mp2',
    kAudioFormatMPEGLayer3              = '.mp3',
    kAudioFormatTimeCode                = 'time',
    kAudioFormatMIDIStream              = 'midi',
    kAudioFormatParameterValueStream    = 'apvs',
    kAudioFormatAppleLossless           = 'alac'
    kAudioFormatMPEG4AAC_HE             = 'aach',
    kAudioFormatMPEG4AAC_LD             = 'aacl',
    kAudioFormatMPEG4AAC_ELD            = 'aace',
    kAudioFormatMPEG4AAC_HE_V2          = 'aacp',
    kAudioFormatMPEG4AAC_Spatial        = 'aacs',
    kAudioFormatAMR                     = 'samr',
    kAudioFormatAudible                 = 'AUDB',
    kAudioFormatiLBC                    = 'ilbc',
    kAudioFormatDVIIntelIMA             = 0x6D730011,
    kAudioFormatMicrosoftGSM            = 0x6D730031,
    kAudioFormatAES3                    = 'aes3'
};
```

Constants

`kAudioFormatLinearPCM`

A key that specifies linear PCM, a noncompressed audio data format with one frame per packet. Uses the linear PCM format flags in "`AudioStreamBasicDescription Flags`" (page 31).

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioFormatAC3`

A key that specifies an AC-3 codec. Uses no flags.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioFormat60958AC3`

A key that specifies an AC-3 codec that provides data packaged for transport over an IEC 60958 compliant digital audio interface. Uses the standard format flags in "`AudioStreamBasicDescription Flags`" (page 31).

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioFormatAppleIMA4`
A key that specifies Apple's implementation of the IMA 4:1 ADPCM codec. Uses no flags.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioFormatMPEG4AAC`
A key that specifies an MPEG-4 AAC codec. The flags field contains the MPEG-4 audio object type constant listed in "MPEG-4 Audio Object Type Constants" (page 36) indicating the specific kind of data.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioFormatMPEG4CELP`
A key that specifies an MPEG-4 CELP codec. The flags field contains the MPEG-4 audio object type constant listed in "MPEG-4 Audio Object Type Constants" (page 36) indicating the specific kind of data.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioFormatMPEG4HVXC`
A key that specifies an MPEG-4 HVXC codec. The flags field contains the MPEG-4 audio object type constant listed in "MPEG-4 Audio Object Type Constants" (page 36) indicating the specific kind of data.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioFormatMPEG4TwinVQ`
A key that specifies an MPEG-4 TwinVQ codec. The flags field contains the MPEG-4 audio object type constant listed in "MPEG-4 Audio Object Type Constants" (page 36) indicating the specific kind of data.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioFormatMACE3`
MACE 3:1. Uses no flags.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioFormatMACE6`
MACE 6:1. Uses no flags.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioFormatULaw`
µLaw 2:1. Uses no flags.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioFormatALaw`
aLaw 2:1. Uses no flags.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioFormatQDesign`
QDesign music. Uses no flags

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioFormatQDesign2`
QDesign2 music. Uses no flags

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioFormatQUALCOMM`
QUALCOMM PureVoice. Uses no flags

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioFormatMPEGLayer1`
MPEG-1/2, Layer 1 audio. Uses no flags

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioFormatMPEGLayer2`
MPEG-1/2, Layer 2 audio. Uses no flags

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioFormatMPEGLayer3`
MPEG-1/2, Layer 3 audio. Uses no flags

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioFormatTimeCode`
A stream of `IOAudioTimeStamp` structures. Uses the `IOAudioTimeStamp` flags (see "Audio Time Stamp Flags" (page 39) and "Audio Time Stamp Flag Combination Constant" (page 40)).

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioFormatMIDIStream`
A stream of `MIDIPacketList` structures where the time stamps in the `MIDIPacket` structures are sample offsets in the stream. The `mSampleRate` field in the `AudioStreamBasicDescription` structure is used to describe how time is passed in this kind of stream and an audio unit that receives or generates this stream can use this sample rate together with the number of frames it is rendering and the sample offsets within the `MIDIPacketList` to define the time for any MIDI event within this list. Uses no flags.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioFormatParameterValueStream`
A "side-chain" of `Float32` data that can be fed or generated by an audio unit and that is used to send a high density of parameter value control information. An audio unit typically runs a parameter value stream at either the sample rate of the audio unit's audio data, or some integer quotient of this (say a half or a third of the sample rate of the audio). The `mSampleRate` field in the `AudioStreamBasicDescription` structure describes this relationship. Uses no flags.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioFormatAppleLossless`
Apple Lossless. Uses no flags.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioFormatMPEG4AAC_HE`
MPEG-4 High Efficiency AAC audio object. Uses no flags.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioFormatMPEG4AAC_LD`
MPEG-4 AAC Low Delay audio object. Uses no flags.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioFormatMPEG4AAC_ELD`
MPEG-4 AAC Enhanced Low Delay audio object. Uses no flags.

Available in iOS 4.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioFormatMPEG4AAC_HE_V2`
MPEG-4 High Efficiency AAC Version 2 audio object. Uses no flags.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioFormatMPEG4AAC_Spatial`
MPEG-4 Spatial Audio audio object. Uses no flags.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioFormatAMR`
The AMR (Adaptive Multi-Rate) narrow band speech codec.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioFormatAudible`
The codec used for Audible, Inc. audio books. Uses no flags.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioFormatiLBC`
: The iLBC (internet Low Bitrate Codec) narrow band speech codec. Uses no flags.

  Available in iOS 2.0 and later.

  Declared in `CoreAudioTypes.h`.

`kAudioFormatDVIIntelIMA`
: DVI/Intel IMA ADPCM - ACM code 17.

  Available in iOS 3.0 and later.

  Declared in `CoreAudioTypes.h`.

`kAudioFormatMicrosoftGSM`
: Microsoft GSM 6.10 - ACM code 49.

  Available in iOS 3.0 and later.

  Declared in `CoreAudioTypes.h`.

`kAudioFormatAES3`
: The format defined by the AES3-2003 standard. Adopted into MXF and MPEG-2 containers and SDTI transport streams with SMPTE specs 302M-2002 and 331M-2000. Uses no flags.

  Available in iOS 3.0 and later.

  Declared in `CoreAudioTypes.h`.

Discussion
Use these identifiers to test for the presence of audio codecs on a system. If a given codec is present, you can use its identifier to specify that codec for data encoding or decoding, according to the capabilities of the codec. For more information, see *Core Audio Overview*.

AudioStreamBasicDescription Flags

Standard flags for use in the `mFormatFlags` field of the `AudioStreamBasicDescription` (page 20) structure.

```
enum {
    kAudioFormatFlagIsFloat                  = (1 << 0),    // 0x1
    kAudioFormatFlagIsBigEndian              = (1 << 1),    // 0x2
    kAudioFormatFlagIsSignedInteger          = (1 << 2),    // 0x4
    kAudioFormatFlagIsPacked                 = (1 << 3),    // 0x8
    kAudioFormatFlagIsAlignedHigh            = (1 << 4),    // 0x10
    kAudioFormatFlagIsNonInterleaved         = (1 << 5),    // 0x20
    kAudioFormatFlagIsNonMixable             = (1 << 6),    // 0x40
    kAudioFormatFlagsAreAllClear             = (1 << 31),

    kLinearPCMFormatFlagIsFloat              = kAudioFormatFlagIsFloat,
    kLinearPCMFormatFlagIsBigEndian          = kAudioFormatFlagIsBigEndian,
    kLinearPCMFormatFlagIsSignedInteger      = kAudioFormatFlagIsSignedInteger,
    kLinearPCMFormatFlagIsPacked             = kAudioFormatFlagIsPacked,
    kLinearPCMFormatFlagIsAlignedHigh        = kAudioFormatFlagIsAlignedHigh,
    kLinearPCMFormatFlagIsNonInterleaved     = kAudioFormatFlagIsNonInterleaved,
    kLinearPCMFormatFlagIsNonMixable         = kAudioFormatFlagIsNonMixable,
    kLinearPCMFormatFlagsSampleFractionShift = 7,
    kLinearPCMFormatFlagsSampleFractionMask  =
        (0x3F << kLinearPCMFormatFlagsSampleFractionShift),
    kLinearPCMFormatFlagsAreAllClear         = kAudioFormatFlagsAreAllClear,
```

```
    kAppleLosslessFormatFlag_16BitSourceData = 1,
    kAppleLosslessFormatFlag_20BitSourceData = 2,
    kAppleLosslessFormatFlag_24BitSourceData = 3,
    kAppleLosslessFormatFlag_32BitSourceData = 4
};
```

Constants

`kAudioFormatFlagIsFloat`

Set for floating point, clear for integer.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioFormatFlagIsBigEndian`

Set for big endian, clear for little endian.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioFormatFlagIsSignedInteger`

Set for signed integer, clear for unsigned integer. This is only valid if `kAudioFormatFlagIsFloat` is clear.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioFormatFlagIsPacked`

Set if the sample bits occupy the entire available bits for the channel, clear if they are high- or low-aligned within the channel.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioFormatFlagIsAlignedHigh`

Set if the sample bits are placed into the high bits of the channel, clear for low bit placement. This is only valid if `kAudioFormatFlagIsPacked` is clear.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioFormatFlagIsNonInterleaved`

Set if the samples for each channel are located contiguously and the channels are laid out end to end, clear if the samples for each frame are laid out contiguously and the frames laid out end to end. This flag affects the use of the `AudioStreamBasicDescription` and `AudioBufferList` structures; see the discussion of the `AudioStreamBasicDescription` (page 20) structure for details.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioFormatFlagIsNonMixable`

Set to indicate when a format is nonmixable. Note that this flag is only used when interacting with the HAL's stream format information. It is not a valid flag for any other use.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kLinearPCMFormatFlagsSampleFractionShift`
: The linear PCM flags contain a 6-bit bitfield indicating that an integer format is to be interpreted as fixed point. The value indicates the number of bits are used to represent the fractional portion of each sample value. This constant indicates the bit position (counting from the right) of the bitfield in `mFormatFlags` field.

  Available in iOS 2.0 and later.

  Declared in `CoreAudioTypes.h`.

`kLinearPCMFormatFlagsSampleFractionMask`
: *<number_of_fractional_bits>* `= (mFormatFlags & kLinearPCMFormatFlagsSampleFractionMask) >> kLinearPCMFormatFlagsSampleFractionShift`

  Available in iOS 2.0 and later.

  Declared in `CoreAudioTypes.h`.

`kAudioFormatFlagsAreAllClear`
: Set to indicate all the flags are clear. You must use this constant instead of `0`, because a `0` in the `mFormatFlags` field of the `AudioStreamBasicDescription` structure indicates that there are no format flags.

  Available in iOS 2.0 and later.

  Declared in `CoreAudioTypes.h`.

`kLinearPCMFormatFlagIsFloat`
: Synonym for `kAudioFormatFlagIsFloat`.

  Available in iOS 2.0 and later.

  Declared in `CoreAudioTypes.h`.

`kLinearPCMFormatFlagIsBigEndian`
: Synonym for `kAudioFormatFlagIsBigEndian`.

  Available in iOS 2.0 and later.

  Declared in `CoreAudioTypes.h`.

`kLinearPCMFormatFlagIsSignedInteger`
: Synonym for `kAudioFormatFlagIsSignedInteger`.

  Available in iOS 2.0 and later.

  Declared in `CoreAudioTypes.h`.

`kLinearPCMFormatFlagIsPacked`
: Synonym for `kAudioFormatFlagIsPacked`.

  Available in iOS 2.0 and later.

  Declared in `CoreAudioTypes.h`.

`kLinearPCMFormatFlagIsAlignedHigh`
: Synonym for `kAudioFormatFlagIsAlignedHigh`.

  Available in iOS 2.0 and later.

  Declared in `CoreAudioTypes.h`.

`kLinearPCMFormatFlagIsNonInterleaved`
: Synonym for `kAudioFormatFlagIsNonInterleaved`.

  Available in iOS 2.0 and later.

  Declared in `CoreAudioTypes.h`.

`kLinearPCMFormatFlagIsNonMixable`
Synonym for `kAudioFormatFlagIsNonMixable`.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kLinearPCMFormatFlagsAreAllClear`
Synonym for `kAudioFormatFlagsAreAllClear`.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAppleLosslessFormatFlag_16BitSourceData`
This flag is set for Apple Lossless data that was sourced from 16 bit native endian signed integer data.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAppleLosslessFormatFlag_20BitSourceData`
Set for Apple Lossless data that was sourced from 20 bit native endian signed integer data aligned high in 24 bits.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAppleLosslessFormatFlag_24BitSourceData`
Set for Apple Lossless data that was sourced from 24 bit native endian signed integer data.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAppleLosslessFormatFlag_32BitSourceData`
Set for Apple Lossless data that was sourced from 32 bit native endian signed integer data.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

AudioStreamBasicDescription Flag Combinations Constants

Commonly used combinations of data format flags for the `AudioStreamBasicDescription` (page 20) structure.

```
enum {
#if TARGET_RT_BIG_ENDIAN
    kAudioFormatFlagsNativeEndian       = kAudioFormatFlagIsBigEndian,
#else
    kAudioFormatFlagsNativeEndian       = 0,
#endif
```

```
#if !CA_PREFER_FIXED_POINT
    kAudioFormatFlagsCanonical          =   kAudioFormatFlagIsFloat |
                                            kAudioFormatFlagsNativeEndian |
                                            kAudioFormatFlagIsPacked,
    kAudioFormatFlagsAudioUnitCanonical =   kAudioFormatFlagIsFloat |
                                            kAudioFormatFlagsNativeEndian |
                                            kAudioFormatFlagIsPacked |
                                            kAudioFormatFlagIsNonInterleaved,
#else
    kAudioFormatFlagsCanonical          =   kAudioFormatFlagIsSignedInteger |
                                            kAudioFormatFlagsNativeEndian |
                                            kAudioFormatFlagIsPacked,
    kAudioFormatFlagsAudioUnitCanonical =   kAudioFormatFlagIsSignedInteger |
                                            kAudioFormatFlagsNativeEndian |
                                            kAudioFormatFlagIsPacked |
                                            kAudioFormatFlagIsNonInterleaved |
                                (kAudioUnitSampleFractionBits <<
                                 kLinearPCMFormatFlagsSampleFractionShift),

#endif

    kAudioFormatFlagsNativeFloatPacked  =   kAudioFormatFlagIsFloat |
                                            kAudioFormatFlagsNativeEndian |
                                            kAudioFormatFlagIsPacked
};
```

Constants

`kAudioFormatFlagsNativeEndian`

Defined to set or clear `kAudioFormatFlagIsBigEndian` depending on the endianness of the processor at build time.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioFormatFlagsCanonical`

The set of flags for the canonical input-output audio sample type, which match the `AudioSampleType` (page 19) type.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioFormatFlagsAudioUnitCanonical`

The flags for the canonical audio unit and processing sample type, which match the `AudioUnitSampleType` (page 19) type.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioFormatFlagsNativeFloatPacked`

The flags for the canonical format of fully packed, native endian floating point data.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

Discussion

Fixed-point formats are preferred in iOS, while floating-point formats are preferred in Mac OS X.

MPEG-4 Audio Object Type Constants

Used in the `mFormatFlags` field of an `AudioStreamBasicDescription` (page 20) structure that describes an MPEG-4 audio stream to specify the type of MPEG-4 audio data. (Deprecated. Deprecated in Mac OS X v10.5.)

```
enum {
    kMPEG4Object_AAC_Main       = 1,
    kMPEG4Object_AAC_LC         = 2,
    kMPEG4Object_AAC_SSR        = 3,
    kMPEG4Object_AAC_LTP        = 4,
    kMPEG4Object_AAC_SBR        = 5,
    kMPEG4Object_AAC_Scalable   = 6,

    kMPEG4Object_TwinVQ         = 7,
    kMPEG4Object_CELP           = 8,
    kMPEG4Object_HVXC           = 9

};
```

Constants

`kMPEG4Object_AAC_Main`

Advanced audio coding; the basic MPEG-4 technology.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kMPEG4Object_AAC_LC`

Lossless coding; provides compression with no loss of quality.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kMPEG4Object_AAC_SSR`

Scalable sampling rate; provides different sampling frequencies for different targets.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kMPEG4Object_AAC_LTP`

Long term prediction; reduces redundancy in a coded signal.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kMPEG4Object_AAC_SBR`

Spectral band replication; reconstructs high-frequency content from lower frequencies and side information.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kMPEG4Object_AAC_Scalable`

Scalable lossless coding.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kMPEG4Object_TwinVQ`
Transform-domain weighted interleaved vector quantization, an audio codec optimized for audio coding at ultra low bit rates around 8 kbit/s.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kMPEG4Object_CELP`
Code Excited Linear Prediction, a narrow-band/wide-band speech codec.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kMPEG4Object_HVXC`
Harmonic Vector Excitation Coding, a very-low bit-rate parametric speech codec.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

Discussion
See the Moving Picture Experts Group web page (http://www.chiariglione.org/mpeg/) for details about MPEG technologies.

SMPTE Timecode Type Constants

SMPTE timecode types, used in the `SMPTETime` (page 24) structure.

```
enum {
    kSMPTETimeType24        = 0,
    kSMPTETimeType25        = 1,
    kSMPTETimeType30Drop    = 2,
    kSMPTETimeType30        = 3,
    kSMPTETimeType2997      = 4,
    kSMPTETimeType2997Drop  = 5,
    kSMPTETimeType60        = 6,
    kSMPTETimeType5994      = 7,
    kSMPTETimeType60Drop    = 8,
    kSMPTETimeType5994Drop  = 9,
    kSMPTETimeType50        = 10,
    kSMPTETimeType2398      = 11
};
```

Constants
`kSMPTETimeType24`
24 video frames per second—standard for 16mm and 35mm film.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kSMPTETimeType25`
25 video frames per second—standard for PAL and SECAM video.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kSMPTETimeType30Drop`
30 video frames per second, with video-frame numbers adjusted to ensure that the timecode matches elapsed clock time.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kSMPTETimeType30`
30 video frames per second.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kSMPTETimeType2997`
29.97 video frames per second—standard for NTSC video.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kSMPTETimeType2997Drop`
29.97 video frames per second, with video-frame numbers adjusted to ensure that the timecode matches elapsed clock time.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kSMPTETimeType60`
60 video frames per second.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kSMPTETimeType5994`
59.94 video frames per second.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kSMPTETimeType60Drop`
60 video frames per second, with video-frame numbers adjusted to ensure that the timecode matches elapsed clock time.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kSMPTETimeType5994Drop`
59.94 video frames per second, with video-frame numbers adjusted to ensure that the timecode matches elapsed clock time.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kSMPTETimeType50`
50 video frames per second.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kSMPTETimeType2398`
23.98 video frames per second.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

SMPTE State Flags

Flags that describe a SMPTE time state.

```
enum {
    kSMPTETimeValid     = (1 << 0),
    kSMPTETimeRunning   = (1 << 1)
};
```

Constants

`kSMPTETimeValid`
The full time is valid.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kSMPTETimeRunning`
Time is running.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

Audio Time Stamp Flags

These flags indicate the valid fields in an `AudioTimeStamp` (page 22) structure.

```
enum {
    kAudioTimeStampSampleTimeValid      = (1 << 0),
    kAudioTimeStampHostTimeValid        = (1 << 1),
    kAudioTimeStampRateScalarValid      = (1 << 2),
    kAudioTimeStampWordClockTimeValid   = (1 << 3),
    kAudioTimeStampSMPTETimeValid       = (1 << 4)
};
```

Constants

`kAudioTimeStampSampleTimeValid`
The sample frame time is valid.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioTimeStampHostTimeValid`
The host time is valid.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioTimeStampRateScalarValid`
The rate scalar is valid.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioTimeStampWordClockTimeValid`
The word clock time is valid.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioTimeStampSMPTETimeValid`
The SMPTE time is valid.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

Audio Time Stamp Flag Combination Constant

A commonly used combination of audio time stamp flags.

```
enum {
    kAudioTimeStampSampleHostTimeValid  = (kAudioTimeStampSampleTimeValid |
kAudioTimeStampHostTimeValid)
};
```

Constants

`kAudioTimeStampSampleHostTimeValid`
The sample frame time and the host time are valid.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

Audio Channel Label Constants

Channel labels for use in the `mChannelLabel` field of an `AudioChannelDescription` (page 17) structure.

```
enum {
    kAudioChannelLabel_Unknown               = 0xFFFFFFFF,
    kAudioChannelLabel_Unused                = 0,
    kAudioChannelLabel_UseCoordinates        = 100,
```

```
    kAudioChannelLabel_Left                 = 1,
    kAudioChannelLabel_Right                = 2,
    kAudioChannelLabel_Center               = 3,
    kAudioChannelLabel_LFEScreen            = 4,
    kAudioChannelLabel_LeftSurround         = 5,
    kAudioChannelLabel_RightSurround        = 6,
    kAudioChannelLabel_LeftCenter           = 7,
    kAudioChannelLabel_RightCenter          = 8,
    kAudioChannelLabel_CenterSurround       = 9,
    kAudioChannelLabel_LeftSurroundDirect   = 10,
    kAudioChannelLabel_RightSurroundDirect  = 11,
    kAudioChannelLabel_TopCenterSurround    = 12,
    kAudioChannelLabel_VerticalHeightLeft   = 13,
    kAudioChannelLabel_VerticalHeightCenter = 14,
    kAudioChannelLabel_VerticalHeightRight  = 15,
    kAudioChannelLabel_TopBackLeft          = 16,
    kAudioChannelLabel_TopBackCenter        = 17,
    kAudioChannelLabel_TopBackRight         = 18,
    kAudioChannelLabel_RearSurroundLeft     = 33,
    kAudioChannelLabel_RearSurroundRight    = 34,
    kAudioChannelLabel_LeftWide             = 35,
    kAudioChannelLabel_RightWide            = 36,
    kAudioChannelLabel_LFE2                 = 37,
    kAudioChannelLabel_LeftTotal            = 38,
    kAudioChannelLabel_RightTotal           = 39,
    kAudioChannelLabel_HearingImpaired      = 40,
    kAudioChannelLabel_Narration            = 41,
    kAudioChannelLabel_Mono                 = 42,
    kAudioChannelLabel_DialogCentricMix     = 43,
    kAudioChannelLabel_CenterSurroundDirect = 44,
    kAudioChannelLabel_Haptic               = 45,

    // first order ambisonic channels
    kAudioChannelLabel_Ambisonic_W          = 200,
    kAudioChannelLabel_Ambisonic_X          = 201,
    kAudioChannelLabel_Ambisonic_Y          = 202,
    kAudioChannelLabel_Ambisonic_Z          = 203,

    // Mid/Side Recording
    kAudioChannelLabel_MS_Mid               = 204,
    kAudioChannelLabel_MS_Side              = 205,

    // X-Y Recording
    kAudioChannelLabel_XY_X                 = 206,
    kAudioChannelLabel_XY_Y                 = 207,

    // other
    kAudioChannelLabel_HeadphonesLeft       = 301,
    kAudioChannelLabel_HeadphonesRight      = 302,
    kAudioChannelLabel_ClickTrack           = 304,
    kAudioChannelLabel_ForeignLanguage      = 305,

    // generic discrete channel
    kAudioChannelLabel_Discrete             = 400,
```

```
    // numbered discrete channel
    kAudioChannelLabel_Discrete_0           = (1<<16) | 0,
    kAudioChannelLabel_Discrete_1           = (1<<16) | 1,
    kAudioChannelLabel_Discrete_2           = (1<<16) | 2,
    kAudioChannelLabel_Discrete_3           = (1<<16) | 3,
    kAudioChannelLabel_Discrete_4           = (1<<16) | 4,
    kAudioChannelLabel_Discrete_5           = (1<<16) | 5,
    kAudioChannelLabel_Discrete_6           = (1<<16) | 6,
    kAudioChannelLabel_Discrete_7           = (1<<16) | 7,
    kAudioChannelLabel_Discrete_8           = (1<<16) | 8,
    kAudioChannelLabel_Discrete_9           = (1<<16) | 9,
    kAudioChannelLabel_Discrete_10          = (1<<16) | 10,
    kAudioChannelLabel_Discrete_11          = (1<<16) | 11,
    kAudioChannelLabel_Discrete_12          = (1<<16) | 12,
    kAudioChannelLabel_Discrete_13          = (1<<16) | 13,
    kAudioChannelLabel_Discrete_14          = (1<<16) | 14,
    kAudioChannelLabel_Discrete_15          = (1<<16) | 15,
    kAudioChannelLabel_Discrete_65535       = (1<<16) | 65535
};
```

Constants

`kAudioChannelLabel_Unknown`

Unknown role or unspecified other use for channel.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLabel_Unused`

The channel is present, but has no intended role or destination.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLabel_UseCoordinates`

The channel is described solely by the `mCoordinates` field of the `AudioChannelDescription` structure.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLabel_Left`

Left channel.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLabel_Right`

Right channel.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLabel_Center`

Center channel.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLabel_LFEScreen`
Low Frequency Effects Screen; a subwoofer located in front of the theater.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLabel_LeftSurround`
Left surround channel; or for WAVE (.wav) files, back left.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLabel_RightSurround`
Right surround channel; or for WAVE (.wav) files, back right.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLabel_LeftCenter`
Left center channel.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLabel_RightCenter`
Right center channel.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLabel_CenterSurround`
Center surround channel; or for WAVE (.wav) files, back center or rear surround.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLabel_LeftSurroundDirect`
Left surround direct channel; or for WAVE (.wav) files, side left.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLabel_RightSurroundDirect`
Right surround direct channel; or for WAVE (.wav) files, side right.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLabel_TopCenterSurround`
Top center surround-sound channel.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLabel_VerticalHeightLeft`
Vertical height left channel; or for WAVE (.wav) files, top front left.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLabel_VerticalHeightCenter`
Vertical height center channel; or for WAVE (.wav) files, top front center.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLabel_VerticalHeightRight`
Vertical height right channel; or for WAVE (.wav) files, top front right.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLabel_TopBackLeft`
Top back left channel.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLabel_TopBackCenter`
Top back center channel.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLabel_TopBackRight`
Top back right channel.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLabel_RearSurroundLeft`
Rear surround left channel.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLabel_RearSurroundRight`
Rear surround right channel.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLabel_LeftWide`
Left wide channel.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLabel_RightWide`
Right wide channel.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLabel_LFE2`
Low Frequency Effects 2.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLabel_LeftTotal`
The left channel of matrix encoded 4 channel audio.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLabel_RightTotal`
The right channel of matrix encoded 4 channel audio.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLabel_HearingImpaired`
Channel carrying audio for the hearing impaired.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLabel_Narration`
Narration channel.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLabel_Mono`
Monaural channel.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLabel_DialogCentricMix`

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLabel_CenterSurroundDirect`
Back center, non diffuse channel.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLabel_Haptic`
A channel for haptic (touch) data.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLabel_Ambisonic_W`
First order Ambisonic channel W.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLabel_Ambisonic_X`
First order Ambisonic channel X.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLabel_Ambisonic_Y`
First order Ambisonic channel Y.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLabel_Ambisonic_Z`
First order Ambisonic channel Z.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLabel_MS_Mid`
Mid channel of a Mid/Side recording.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLabel_MS_Side`
Side channel of a Mid/Side recording.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLabel_XY_X`
X channel of an X-Y recording.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLabel_XY_Y`
Y channel of an X-Y recording.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLabel_HeadphonesLeft`
Left channel of stereo headphones.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLabel_HeadphonesRight`
Right channel of stereo headphones.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLabel_ClickTrack`
Click track channel.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLabel_ForeignLanguage`
Foreign language channel.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLabel_Discrete`
Generic discrete channel.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLabel_Discrete_0`
Discrete channel 0.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLabel_Discrete_1`
Discrete channel 1.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLabel_Discrete_2`
Discrete channel 2.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLabel_Discrete_3`
Discrete channel 3.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLabel_Discrete_4`
Discrete channel 4.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLabel_Discrete_5`
Discrete channel 5.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLabel_Discrete_6`
Discrete channel 6.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLabel_Discrete_7`
Discrete channel 7.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLabel_Discrete_8`
Discrete channel 8.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLabel_Discrete_9`
Discrete channel 9.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLabel_Discrete_10`
Discrete channel 10.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLabel_Discrete_11`
Discrete channel 11.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLabel_Discrete_12`
Discrete channel 12.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLabel_Discrete_13`
Discrete channel 13.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLabel_Discrete_14`
Discrete channel 14.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLabel_Discrete_15`
Discrete channel 15.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLabel_Discrete_65535`
Discrete channel 65536.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

Channel Bitmap Constants

Channel bits for use in the `mChannelBitmap` field of an `AudioChannelLayout` (page 18) structure.

```
enum {
    kAudioChannelBit_Left                       = (1<<0),
    kAudioChannelBit_Right                      = (1<<1),
    kAudioChannelBit_Center                     = (1<<2),
    kAudioChannelBit_LFEScreen                  = (1<<3),
    kAudioChannelBit_LeftSurround               = (1<<4),
    kAudioChannelBit_RightSurround              = (1<<5),
    kAudioChannelBit_LeftCenter                 = (1<<6),
    kAudioChannelBit_RightCenter                = (1<<7),
    kAudioChannelBit_CenterSurround             = (1<<8),
    kAudioChannelBit_LeftSurroundDirect         = (1<<9),
    kAudioChannelBit_RightSurroundDirect        = (1<<10),
    kAudioChannelBit_TopCenterSurround          = (1<<11),
    kAudioChannelBit_VerticalHeightLeft         = (1<<12),
    kAudioChannelBit_VerticalHeightCenter       = (1<<13),
    kAudioChannelBit_VerticalHeightRight        = (1<<14),
    kAudioChannelBit_TopBackLeft                = (1<<15),
    kAudioChannelBit_TopBackCenter              = (1<<16),
    kAudioChannelBit_TopBackRight               = (1<<17)
};
```

Constants

`kAudioChannelBit_Left`

Left channel.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelBit_Right`

Right channel.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelBit_Center`

Center channel.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelBit_LFEScreen`

Low Frequency Effects screen channel.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelBit_LeftSurround`

Left surround channel; or for WAVE (.wav) files, back left.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelBit_RightSurround`

RIght surround channel; or for WAVE (.wav) files, back right.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelBit_LeftCenter`
Left center channel.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelBit_RightCenter`
Right center channel.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelBit_CenterSurround`
Center surround channel; or for WAVE (.wav) files, back center.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelBit_LeftSurroundDirect`
Left surround direct channel; or for WAVE (.wav) files, side left.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelBit_RightSurroundDirect`
Right surround direct channel; or for WAVE (.wav) files, side right.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelBit_TopCenterSurround`
To center surround channel.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelBit_VerticalHeightLeft`
Vertical height left channel; or for WAVE (.wav) files, top front left.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelBit_VerticalHeightCenter`
Vertical height center channel; or for WAVE (.wav) files, top front center.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelBit_VerticalHeightRight`
Vertical height right channel; or for WAVE (.wav) files, top front right.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelBit_TopBackLeft`
Top back left channel.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelBit_TopBackCenter`
Top back center channel.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelBit_TopBackRight`
Top back right channel.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

Channel Coordinate Flags

Used in the `mChannelFlags` field of an `AudioChannelDescription` (page 17) structure.

```
enum {
    kAudioChannelFlags_AllOff                    = 0,
    kAudioChannelFlags_RectangularCoordinates    = (1<<0),
    kAudioChannelFlags_SphericalCoordinates      = (1<<1),
    kAudioChannelFlags_Meters                    = (1<<2)
};
```

Constants

`kAudioChannelFlags_AllOff`
All flags are clear.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelFlags_RectangularCoordinates`
Set to indicate the channel is specified by the Cartesian coordinates of the speaker position. This flag is mutually exclusive with `kAudioChannelFlags_SphericalCoordinates`.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelFlags_SphericalCoordinates`
Set to indicate the channel is specified by the spherical coordinates of the speaker position. This flag is mutually exclusive with `kAudioChannelFlags_RectangularCoordinates`.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelFlags_Meters`
Set to indicate the units are in meters, clear to indicate the units are relative to the unit cube or unit sphere. For relative units, the listener is assumed to be at the center of the cube or sphere and the radius of the sphere or the distance from the center to the midpoint of the side of the cube is 1.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

Channel Coordinate Index Constants

Indexes the fields of the `mCoordinates` array in an `AudioChannelDescription` (page 17) structure.

```
enum {
    kAudioChannelCoordinates_LeftRight  = 0,
    kAudioChannelCoordinates_BackFront  = 1,
    kAudioChannelCoordinates_DownUp     = 2,
    kAudioChannelCoordinates_Azimuth    = 0,
    kAudioChannelCoordinates_Elevation  = 1,
    kAudioChannelCoordinates_Distance   = 2
};
```

Constants

`kAudioChannelCoordinates_LeftRight`
For rectangular coordinates, negative is left and positive is right. The units are specified by the `mChannelFlags` field of the `AudioChannelDescription` structure.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelCoordinates_BackFront`
For rectangular coordinates, negative is back and positive is front. The units are specified by the `mChannelFlags` field.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelCoordinates_DownUp`
For rectangular coordinates, negative is below ground level, `0` is ground level, and positive is above ground level. The units are specified by the `mChannelFlags` field.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelCoordinates_Azimuth`
For spherical coordinates, `0` is front center, positive is right, negative is left, and measurements are in degrees.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelCoordinates_Elevation`
For spherical coordinates, `+90` is zenith, `0` is horizontal, `-90` is nadir, and measurements are in degrees.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelCoordinates_Distance`
For spherical coordinates, distance is radially from the center. The units are specified by the `mChannelFlags` field of the `AudioChannelDescription` structure.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

Audio Channel Layout Tags

Identifiers for audio channel layouts. These identifiers specify the channels included in a layout but do not specify a particular ordering of those channels. Used in the `mChannelLayoutTag` field of an `AudioChannelLayout` (page 18) structure.

```
enum {
    kAudioChannelLayoutTag_UseChannelDescriptions  = (0<<16) | 0,
    kAudioChannelLayoutTag_UseChannelBitmap        = (1<<16) | 0,

    //  General layouts
    kAudioChannelLayoutTag_Mono                    = (100<<16) | 1,
    kAudioChannelLayoutTag_Stereo                  = (101<<16) | 2,
    kAudioChannelLayoutTag_StereoHeadphones        = (102<<16) | 2,
    kAudioChannelLayoutTag_MatrixStereo            = (103<<16) | 2,
    kAudioChannelLayoutTag_MidSide                 = (104<<16) | 2,
    kAudioChannelLayoutTag_XY                      = (105<<16) | 2,
    kAudioChannelLayoutTag_Binaural                = (106<<16) | 2,
    kAudioChannelLayoutTag_Ambisonic_B_Format      = (107<<16) | 4,
    kAudioChannelLayoutTag_Quadraphonic            = (108<<16) | 4,
    kAudioChannelLayoutTag_Pentagonal              = (109<<16) | 5,
    kAudioChannelLayoutTag_Hexagonal               = (110<<16) | 6,
    kAudioChannelLayoutTag_Octagonal               = (111<<16) | 8,
    kAudioChannelLayoutTag_Cube                    = (112<<16) | 8,

    //  MPEG defined layouts
    kAudioChannelLayoutTag_MPEG_1_0                = kAudioChannelLayoutTag_Mono,
    kAudioChannelLayoutTag_MPEG_2_0               = kAudioChannelLayoutTag_Stereo,
    kAudioChannelLayoutTag_MPEG_3_0_A              = (113<<16) | 3,
    kAudioChannelLayoutTag_MPEG_3_0_B              = (114<<16) | 3,
    kAudioChannelLayoutTag_MPEG_4_0_A              = (115<<16) | 4,
    kAudioChannelLayoutTag_MPEG_4_0_B              = (116<<16) | 4,
    kAudioChannelLayoutTag_MPEG_5_0_A              = (117<<16) | 5,
    kAudioChannelLayoutTag_MPEG_5_0_B              = (118<<16) | 5,
    kAudioChannelLayoutTag_MPEG_5_0_C              = (119<<16) | 5,
    kAudioChannelLayoutTag_MPEG_5_0_D              = (120<<16) | 5,
    kAudioChannelLayoutTag_MPEG_5_1_A              = (121<<16) | 6,
    kAudioChannelLayoutTag_MPEG_5_1_B              = (122<<16) | 6,
    kAudioChannelLayoutTag_MPEG_5_1_C              = (123<<16) | 6,
    kAudioChannelLayoutTag_MPEG_5_1_D              = (124<<16) | 6,
    kAudioChannelLayoutTag_MPEG_6_1_A              = (125<<16) | 7,
    kAudioChannelLayoutTag_MPEG_7_1_A              = (126<<16) | 8,
    kAudioChannelLayoutTag_MPEG_7_1_B              = (127<<16) | 8,
    kAudioChannelLayoutTag_MPEG_7_1_C              = (128<<16) | 8,
    kAudioChannelLayoutTag_Emagic_Default_7_1      = (129<<16) | 8,
    kAudioChannelLayoutTag_SMPTE_DTV               = (130<<16) | 8,

    //  ITU defined layouts
    kAudioChannelLayoutTag_ITU_1_0                 = kAudioChannelLayoutTag_Mono,
    kAudioChannelLayoutTag_ITU_2_0                = kAudioChannelLayoutTag_Stereo,
    kAudioChannelLayoutTag_ITU_2_1                 = (131<<16) | 3,
    kAudioChannelLayoutTag_ITU_2_2                 = (132<<16) | 4,
    kAudioChannelLayoutTag_ITU_3_0                 =
            kAudioChannelLayoutTag_MPEG_3_0_A,
    kAudioChannelLayoutTag_ITU_3_1                 =
            kAudioChannelLayoutTag_MPEG_4_0_A,
    kAudioChannelLayoutTag_ITU_3_2                 =
            kAudioChannelLayoutTag_MPEG_5_0_A,
    kAudioChannelLayoutTag_ITU_3_2_1               =
            kAudioChannelLayoutTag_MPEG_5_1_A,
    kAudioChannelLayoutTag_ITU_3_4_1               =
            kAudioChannelLayoutTag_MPEG_7_1_C,
```

```
    // DVD defined layouts
    kAudioChannelLayoutTag_DVD_0                  = kAudioChannelLayoutTag_Mono,
    kAudioChannelLayoutTag_DVD_1                  = kAudioChannelLayoutTag_Stereo,
    kAudioChannelLayoutTag_DVD_2                  = kAudioChannelLayoutTag_ITU_2_1,
    kAudioChannelLayoutTag_DVD_3                  = kAudioChannelLayoutTag_ITU_2_2,
    kAudioChannelLayoutTag_DVD_4                  = (133<<16) | 3,
    kAudioChannelLayoutTag_DVD_5                  = (134<<16) | 4,
    kAudioChannelLayoutTag_DVD_6                  = (135<<16) | 5,
    kAudioChannelLayoutTag_DVD_7                  =
            kAudioChannelLayoutTag_MPEG_3_0_A,
    kAudioChannelLayoutTag_DVD_8                  =
             kAudioChannelLayoutTag_MPEG_4_0_A,
    kAudioChannelLayoutTag_DVD_9                  =
            kAudioChannelLayoutTag_MPEG_5_0_A,
    kAudioChannelLayoutTag_DVD_10                 = (136<<16) | 4,
    kAudioChannelLayoutTag_DVD_11                 = (137<<16) | 5,
    kAudioChannelLayoutTag_DVD_12                 =
            kAudioChannelLayoutTag_MPEG_5_1_A,
    kAudioChannelLayoutTag_DVD_13                 = kAudioChannelLayoutTag_DVD_8,
    kAudioChannelLayoutTag_DVD_14                 = kAudioChannelLayoutTag_DVD_9,
    kAudioChannelLayoutTag_DVD_15                 = kAudioChannelLayoutTag_DVD_10,
    kAudioChannelLayoutTag_DVD_16                 = kAudioChannelLayoutTag_DVD_11,
    kAudioChannelLayoutTag_DVD_17                 = kAudioChannelLayoutTag_DVD_12,
    kAudioChannelLayoutTag_DVD_18                 = (138<<16) | 5,
    kAudioChannelLayoutTag_DVD_19                 =
            kAudioChannelLayoutTag_MPEG_5_0_B,
    kAudioChannelLayoutTag_DVD_20                 =
            kAudioChannelLayoutTag_MPEG_5_1_B,

    // These layouts are recommended for AudioUnit use;
    // these are the symmetrical layouts
    kAudioChannelLayoutTag_AudioUnit_4            =
            kAudioChannelLayoutTag_Quadraphonic,
    kAudioChannelLayoutTag_AudioUnit_5            =
            kAudioChannelLayoutTag_Pentagonal,
    kAudioChannelLayoutTag_AudioUnit_6            =
            kAudioChannelLayoutTag_Hexagonal,
    kAudioChannelLayoutTag_AudioUnit_8            =
            kAudioChannelLayoutTag_Octagonal,
    // These are the surround-based layouts
    kAudioChannelLayoutTag_AudioUnit_5_0          =
            kAudioChannelLayoutTag_MPEG_5_0_B,
    kAudioChannelLayoutTag_AudioUnit_6_0          = (139<<16) | 6,
    kAudioChannelLayoutTag_AudioUnit_7_0          = (140<<16) | 7,
    kAudioChannelLayoutTag_AudioUnit_7_0_Front    = (148<<16) | 7,
    kAudioChannelLayoutTag_AudioUnit_5_1          =
             kAudioChannelLayoutTag_MPEG_5_1_A,
    kAudioChannelLayoutTag_AudioUnit_6_1          =
            kAudioChannelLayoutTag_MPEG_6_1_A,
    kAudioChannelLayoutTag_AudioUnit_7_1          =
            kAudioChannelLayoutTag_MPEG_7_1_C,
    kAudioChannelLayoutTag_AudioUnit_7_1_Front    =
            kAudioChannelLayoutTag_MPEG_7_1_A,
```

```
    kAudioChannelLayoutTag_AAC_3_0                  =
            kAudioChannelLayoutTag_MPEG_3_0_B,
    kAudioChannelLayoutTag_AAC_Quadraphonic         =
            kAudioChannelLayoutTag_Quadraphonic,
    kAudioChannelLayoutTag_AAC_4_0                  =
            kAudioChannelLayoutTag_MPEG_4_0_B,
    kAudioChannelLayoutTag_AAC_5_0                  =
            kAudioChannelLayoutTag_MPEG_5_0_D,
    kAudioChannelLayoutTag_AAC_5_1                  =
            kAudioChannelLayoutTag_MPEG_5_1_D,
    kAudioChannelLayoutTag_AAC_6_0                  = (141<<16) | 6,
    kAudioChannelLayoutTag_AAC_6_1                  = (142<<16) | 7,
    kAudioChannelLayoutTag_AAC_7_0                  = (143<<16) | 7,
    kAudioChannelLayoutTag_AAC_7_1                  =
            kAudioChannelLayoutTag_MPEG_7_1_B,
    kAudioChannelLayoutTag_AAC_Octagonal            = (144<<16) | 8,

    kAudioChannelLayoutTag_TMH_10_2_std             = (145<<16) | 16,
    kAudioChannelLayoutTag_TMH_10_2_full            = (146<<16) | 21,

    kAudioChannelLayoutTag_AC3_1_0_1                = (149<<16) | 2,
    kAudioChannelLayoutTag_AC3_3_0                  = (150<<16) | 3,
    kAudioChannelLayoutTag_AC3_3_1                  = (151<<16) | 4,
    kAudioChannelLayoutTag_AC3_3_0_1                = (152<<16) | 4,
    kAudioChannelLayoutTag_AC3_2_1_1                = (153<<16) | 4,
    kAudioChannelLayoutTag_AC3_3_1_1                = (154<<16) | 5,

    kAudioChannelLayoutTag_EAC_6_0_A                = (155<<16) | 6,
    kAudioChannelLayoutTag_EAC_7_0_A                = (156<<16) | 7,
    kAudioChannelLayoutTag_EAC3_6_1_A               = (157<<16) | 7,
    kAudioChannelLayoutTag_EAC3_6_1_B               = (158<<16) | 7,
    kAudioChannelLayoutTag_EAC3_6_1_C               = (159<<16) | 7,
    kAudioChannelLayoutTag_EAC3_7_1_A               = (160<<16) | 8,
    kAudioChannelLayoutTag_EAC3_7_1_B               = (161<<16) | 8,
    kAudioChannelLayoutTag_EAC3_7_1_C               = (162<<16) | 8,
    kAudioChannelLayoutTag_EAC3_7_1_D               = (163<<16) | 8,
    kAudioChannelLayoutTag_EAC3_7_1_E               = (164<<16) | 8,
    kAudioChannelLayoutTag_EAC3_7_1_F               = (165<<16) | 8,
    kAudioChannelLayoutTag_EAC3_7_1_G               = (166<<16) | 8,
    kAudioChannelLayoutTag_EAC3_7_1_H               = (167<<16) | 8,

    kAudioChannelLayoutTag_DTS_3_1                  = (168<<16) | 4,
    kAudioChannelLayoutTag_DTS_4_1                  = (169<<16) | 5,
    kAudioChannelLayoutTag_DTS_6_0_A                = (170<<16) | 6,
    kAudioChannelLayoutTag_DTS_6_0_B                = (171<<16) | 6,
    kAudioChannelLayoutTag_DTS_6_0_C                = (172<<16) | 6,
    kAudioChannelLayoutTag_DTS_6_1_A                = (173<<16) | 7,
    kAudioChannelLayoutTag_DTS_6_1_B                = (174<<16) | 7,
    kAudioChannelLayoutTag_DTS_6_1_C                = (175<<16) | 7,
    kAudioChannelLayoutTag_DTS_6_1_D                = (182<<16) | 7,
    kAudioChannelLayoutTag_DTS_7_0                  = (176<<16) | 7,
    kAudioChannelLayoutTag_DTS_7_1                  = (177<<16) | 8,
    kAudioChannelLayoutTag_DTS_8_0_A                = (178<<16) | 8,
    kAudioChannelLayoutTag_DTS_8_0_B                = (179<<16) | 8,
    kAudioChannelLayoutTag_DTS_8_1_A                = (180<<16) | 9,
    kAudioChannelLayoutTag_DTS_8_1_B                = (181<<16) | 9,
```

```
    kAudioChannelLayoutTag_DiscreteInOrder         = (147<<16) | 0
    kAudioChannelLayoutTag_Unknown                 = 0xFFFF0000
};
```

Constants

`kAudioChannelLayoutTag_UseChannelDescriptions`
Use the array of `AudioChannelDescription` structures to define the layout.

Available in iOS 2.0 and later.

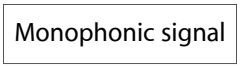
Declared in `CoreAudioTypes.h`.

`kAudioChannelLayoutTag_UseChannelBitmap`
Use the bitmap to define the layout.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

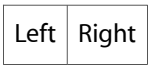
`kAudioChannelLayoutTag_Mono`
A standard monophonic stream.



Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

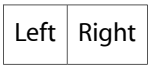
`kAudioChannelLayoutTag_Stereo`
A standard stereophonic stream; playback implied.



Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLayoutTag_StereoHeadphones`
A standard stereo stream; headphone playback implied.



Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLayoutTag_MatrixStereo`
A matrix-encoded stereo stream.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLayoutTag_MidSide`
Mid/side recording.

Center	Sides

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLayoutTag_XY`
Coincident, angled microphone pair.

X (left)	Y (right)

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLayoutTag_Binaural`
Binaural stereo.

Left	Right

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLayoutTag_Ambisonic_B_Format`
Ambisonic B-format.

W	X	Y	Z

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLayoutTag_Quadraphonic`
Quadraphonic, with 90° loudspeaker separation.

Left front	Right front	Left back	Right back

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLayoutTag_Pentagonal`
Pentagonal, with 72° loudspeaker separation.

Left	Right	Left rear	Right rear	Center

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLayoutTag_Hexagonal`
Hexagonal, with 60° loudspeaker separation.

Left	Right	Left rear	Right rear	Front center	Rear center

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLayoutTag_Octagonal`
Octagonal, with 45° loudspeaker separation.

Left front	Right front	Left rear	Right rear	Center front	Center rear	Left side	Right side

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLayoutTag_Cube`
Cubic.

Left front	Right front	Left rear	Right rear	Left front top	Right front top	Left rear top	Right rear top

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLayoutTag_MPEG_1_0`
MPEG 1-channel.

Monophonic signal

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLayoutTag_MPEG_2_0`
MPEG 2-channel.

Left	Right

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLayoutTag_MPEG_3_0_A`
MPEG 3-channel layout A.

Left	Right	Center

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLayoutTag_MPEG_3_0_B`
MPEG 3-channel layout B.

Center	Left	Right

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLayoutTag_MPEG_4_0_A`
MPEG 4- channel layout A.

Left	Right	Center	Center surround

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLayoutTag_MPEG_4_0_B`
MPEG 4-channel layout B.

Center	Left	Right	Center surround

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLayoutTag_MPEG_5_0_A`
MPEG 5-channel layout A.

Left	Right	Center	Left surround	Right surround

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLayoutTag_MPEG_5_0_B`
MPEG 5-channel layout B.

Left	Right	Left surround	Right surround	Center

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLayoutTag_MPEG_5_0_C`
MPEG 5-channel layout C.

Left	Center	Right	Left surround	Right surround

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLayoutTag_MPEG_5_0_D`
MPEG 5-channel layout D.

Center	Left	Right	Left surround	Right surround

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLayoutTag_MPEG_5_1_A`
MPEG 5.1-channel layout A.

Left	Right	Center	Low-frequency effects	Left surround	Right surround

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLayoutTag_MPEG_5_1_B`
MPEG 5.1-channel layout B.

Left	Right	Left surround	Right surround	Center	Low-frequency effects

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLayoutTag_MPEG_5_1_C`
MPEG 5.1-channel layout C.

Left	Center	Right	Left surround	Right surround	Low-frequency effects

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLayoutTag_MPEG_5_1_D`
MPEG 5.1-channel layout D.

Center	Left	Right	Left surround	Right surround	Low-frequency effects

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLayoutTag_MPEG_6_1_A`
MPEG 6.1-channel layout A.

Left	Right	Center	Low-frequency effects	Left surround	Right surround	Center surround

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLayoutTag_MPEG_7_1_A`
MPEG 7.1-channel layout A.

Left	Right	Center	Low-frequency effects	Left surround	Right surround	Left center	Right center

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLayoutTag_MPEG_7_1_B`
MPEG 7.1-channel layout A (see ISO/IEC 13818-7 MPEG2-AAC, Table 3.1).

Center	Left center	Right center	Left	Right	Left surround	Right surround	Low-frequency effects

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLayoutTag_MPEG_7_1_C`
MPEG 7.1-channel layout C.

Left	Right	Center	Low-frequency effects	Left surround	Right surround	Left rear surround	Right rear surround

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLayoutTag_Emagic_Default_7_1`
Emagic 7.1-channel default layout.

Left	Right	Left surround	Right surround	Center	Low-frequency effects	Left center	Right center

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLayoutTag_SMPTE_DTV`
SMPTE DTV layout; equivalent to the kAudioChannelLayoutTag_ITU_5_1 layout plus a matrix encoded stereo mix.

Left	Right	Center	Low-frequency effects	Left surround	Right surround	Left matrix total	Right matrix total

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

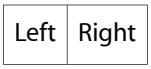
`kAudioChannelLayoutTag_ITU_1_0`
ITU 1-channel layout.

Monophonic signal

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

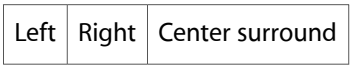
`kAudioChannelLayoutTag_ITU_2_0`
ITU 2-channel layout.



Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

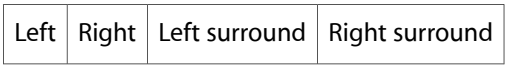
`kAudioChannelLayoutTag_ITU_2_1`
ITU 2.1-channel layout.



Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

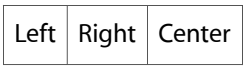
`kAudioChannelLayoutTag_ITU_2_2`
ITU 2.2-channel layout.



Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

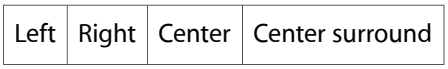
`kAudioChannelLayoutTag_ITU_3_0`
ITU 3-channel layout.



Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

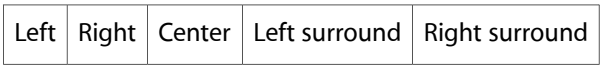
`kAudioChannelLayoutTag_ITU_3_1`
ITU 3.1-channel layout.



Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLayoutTag_ITU_3_2`
ITU 3.2-channel layout.



Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLayoutTag_ITU_3_2_1`
ITU 3.2.1-channel layout.

Left	Right	Center	Low-frequency effects	Left surround	Right surround

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLayoutTag_ITU_3_4_1`
ITU 3.4.1-channel layout.

Left	Right	Center	Low-frequency effects	Left surround	Right surround	Left rear surround	Right rear surround

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLayoutTag_DVD_0`
DVD monaural layout.

Monophonic signal

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLayoutTag_DVD_1`
DVD stereo layout.

Left	Right

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLayoutTag_DVD_2`
DVD 3-channel layout.

Left	Right	Center surround

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLayoutTag_DVD_3`
DVD 4-channel layout.

Left	Right	Left surround	Right surround

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLayoutTag_DVD_4`
DVD 2.1-channel layout.

Left	Right	Low-frequency effects

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLayoutTag_DVD_5`
DVD 3.1-channel layout.

Left	Right	Low-frequency effects	Center surround

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLayoutTag_DVD_6`
DVD 4.1-channel layout.

Left	Right	Low-frequency effects	Left surround	Right surround

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLayoutTag_DVD_7`
DVD 3-channel layout.

Left	Right	Center

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLayoutTag_DVD_8`
DVD 4-channel layout.

Left	Right	Center	Center surround

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLayoutTag_DVD_9`
DVD 5-channel layout.

Left	Right	Center	Left surround	Right surround

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLayoutTag_DVD_10`
DVD 3.1-channel layout.

Left	Right	Center	Low-frequency effects

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLayoutTag_DVD_11`
DVD 4.1-channel layout.

Left	Right	Center	Low-frequency effects	Center surround

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLayoutTag_DVD_12`
DVD 5.1-channel layout.

Left	Right	Center	Low-frequency effects	Left surround	Right surround

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLayoutTag_DVD_13`
DVD 4-channel layout; equivalent to `kAudioChannelLayoutTag_DVD_8` (page 64).

Left	Right	Center	Center surround

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLayoutTag_DVD_14`
DVD 5-channel layout; equivalent to `kAudioChannelLayoutTag_DVD_9` (page 64).

Left	Right	Center	Left surround	Right surround

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLayoutTag_DVD_15`
DVD 3.1-channel layout; equivalent to `kAudioChannelLayoutTag_DVD_10` (page 65).

Left	Right	Center	Low-frequency effects

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLayoutTag_DVD_16`
DVD 4.1-channel layout; equivalent to `kAudioChannelLayoutTag_DVD_11` (page 65).

Left	Right	Center	Low-frequency effects	Center surround

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLayoutTag_DVD_17`
DVD 5.1-channel layout; equivalent to `kAudioChannelLayoutTag_DVD_12` (page 65).

Left	Right	Center	Low-frequency effects	Left surround	Right surround

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLayoutTag_DVD_18`
DVD 4.1-channel layout.

Left	Right	Left surround	Right surround	Low-frequency effects

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLayoutTag_DVD_19`
DVD 5-channel layout.

Left	Right	Left surround	Right surround	Center

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLayoutTag_DVD_20`
DVD 5.1-channel layout.

Left	Right	Left surround	Right surround	Center	Low-frequency effects

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLayoutTag_AudioUnit_4`
Quadraphonic symmetrical layout, recommended for use by audio units.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLayoutTag_AudioUnit_5`
Pentagonal symmetrical layout, recommended for use by audio units.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLayoutTag_AudioUnit_6`
Hexagonal symmetrical layout, recommended for use by audio units.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLayoutTag_AudioUnit_8`
Octagonal symmetrical layout, recommended for use by audio units.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLayoutTag_AudioUnit_5_0`
5-channel surround-based layout, recommended for use by audio units.

Left	Right	Left surround	Right surround	Center

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLayoutTag_AudioUnit_6_0`
6-channel surround-based layout, recommended for use by audio units.

Left	Right	Left surround	Right surround	Center	Center surround

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLayoutTag_AudioUnit_7_0`
7-channel surround-based layout, recommended for use by audio units.

Left	Right	Left surround	Right surround	Center	Left rear surround	Right rear surround

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLayoutTag_AudioUnit_7_0_Front`
Alternate 7-channel surround-based layout, for use by audio units.

Left	Right	Left surround	Right surround	Center	Left center	Right center

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLayoutTag_AudioUnit_5_1`
5.1-channel surround-based layout, recommended for use by audio units.

Left	Right	Center	Low-frequency effects	Left surround	Right surround

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLayoutTag_AudioUnit_6_1`

6.1-channel surround-based layout, recommended for use by audio units.

Left	Right	Center	Low-frequency effects	Left surround	Right surround	Center surround

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLayoutTag_AudioUnit_7_1`

7.1-channel surround-based layout, recommended for use by audio units.

Left	Right	Center	Low-frequency effects	Left surround	Right surround	Left rear surround	Right rear surround

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLayoutTag_AudioUnit_7_1_Front`

7.1-channel surround-based layout, recommended for use by audio units.

Left	Right	Center	Low-frequency effects	Left surround	Right surround	Left center	Right center

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLayoutTag_AAC_3_0`

AAC 3-channel surround-based layout.

Center	Left	Right

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLayoutTag_AAC_Quadraphonic`

AAC quadraphonic surround-based layout.

Left	Right	Left surround	Right surround	Left surround

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLayoutTag_AAC_4_0`

AAC 4-channel surround-based layout.

Center	Left	Right	Center surround

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLayoutTag_AAC_5_0`
AAC 5-channel surround-based layout.

Center	Left	Right	Left surround	Right surround

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLayoutTag_AAC_5_1`
AAC 5.1-channel surround-based layout.

Center	Left	Right	Left surround	Right surround	Low-frequency effects

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLayoutTag_AAC_6_0`
AAC 6-channel surround-based layout.

Center	Left	Right	Left surround	Right surround	Center surround

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLayoutTag_AAC_6_1`
AAC 6.1-channel surround-based layout.

Center	Left	Right	Left surround	Right surround	Center surround	Low-frequency effects

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLayoutTag_AAC_7_0`
AAC 7-channel surround-based layout.

Center	Left	Right	Left surround	Right surround	Left rear surround	Right rear surround

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLayoutTag_AAC_7_1`
AAC 7.1-channel surround-based layout.

Center	Left center	Right center	Left	Right	Left surround	Right surround	Low-frequency effects

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLayoutTag_AAC_Octagonal`

AAC 8-channel surround-based layout.

Center	Left	Right	Left surround	Right surround	Left rear surround	Right rear surround	Center surround

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLayoutTag_TMH_10_2_std`

TMH 10.2, a multiple-channel surround-based layout .

This table contains more than one row to accommodate the page width. There is no relationship between items in the same column.

Left	Right	Center	Vertical height center	Left surround direct	Right surround direct	Left surround	Right surround
Vertical height left	Vertical height right	Left wide	Right wide	Center surround direct	Center surround	Low-frequency effects 1	Low-frequency effects 2

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLayoutTag_TMH_10_2_full`

TMH 10.2 (`kAudioChannelLayoutTag_TMH_10_2_std`) plus additional channels; recommended for use by audio units.

This table contains more than one row to accommodate the page width. There is no relationship between items in the same column.

Left	Right	Center	Vertical height center	Left surround direct	Right surround direct	Left surround	Right surround
Vertical height left	Vertical height right	Left wide	Right wide	Center surround direct	Center surround	Low-frequency effects 1	Low-frequency effects 2
Left center	Right center	HI	VI	Haptic			

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLayoutTag_AC3_1_0_1`

An AC-3 layout.

Center	Low-frequency effects

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLayoutTag_AC3_3_0`
An AC-3 layout.

Left	Center	Right

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLayoutTag_AC3_3_1`
An AC-3 layout.

Left	Center	Right	Center surround

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLayoutTag_AC3_3_0_1`
An AC-3 layout.

Left	Center	Right	Low-frequency effects

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLayoutTag_AC3_2_1_1`
An AC-3 layout.

Left	Right	Center surround	Low-frequency effects

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLayoutTag_AC3_3_1_1`
An AC-3 layout.

Left	Center	Right	Center Surround	Low-frequency Effects

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLayoutTag_EAC_6_0_A`
A Blu-ray Disc audio layout for Enhanced AC-3, also known as Dolby Digital Plus.

Left	Center	Right	Left Surround	Right Surround	Center Surround

Available in iOS 4.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLayoutTag_EAC_7_0_A`

A Blu-ray Disc audio layout for Enhanced AC-3, also known as Dolby Digital Plus.

Left	Center	Right	Left Surround	Right Surround	Left rear surround	Right rear surround

Available in iOS 4.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLayoutTag_EAC3_6_1_A`

A Blu-ray Disc audio layout for Enhanced AC-3, also known as Dolby Digital Plus.

Left	Center	Right	Left Surround	Right Surround	Low-frequency effects	Center surround

Available in iOS 4.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLayoutTag_EAC3_6_1_B`

A Blu-ray Disc audio layout for Enhanced AC-3, also known as Dolby Digital Plus.

Left	Center	Right	Left Surround	Right Surround	Low-frequency effects	Top surround

Available in iOS 4.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLayoutTag_EAC3_6_1_C`

A Blu-ray Disc audio layout for Enhanced AC-3, also known as Dolby Digital Plus.

Left	Center	Right	Left Surround	Right Surround	Low-frequency effects	Vertical height center

Available in iOS 4.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLayoutTag_EAC3_7_1_A`

A Blu-ray Disc audio layout for Enhanced AC-3, also known as Dolby Digital Plus.

Left	Center	Right	Left Surround	Right Surround	Low-frequency effects	Left rear surround	Right rear surround

Available in iOS 4.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLayoutTag_EAC3_7_1_B`

A Blu-ray Disc audio layout for Enhanced AC-3, also known as Dolby Digital Plus.

Left	Center	Right	Left Surround	Right Surround	Low-frequency effects	Left center	Right center

Available in iOS 4.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLayoutTag_EAC3_7_1_C`
A Blu-ray Disc audio layout for Enhanced AC-3, also known as Dolby Digital Plus.

Left	Center	Right	Left Surround	Right Surround	Low-frequency effects	Left surround direct	Right surround direct

Available in iOS 4.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLayoutTag_EAC3_7_1_D`
A Blu-ray Disc audio layout for Enhanced AC-3, also known as Dolby Digital Plus.

Left	Center	Right	Left Surround	Right Surround	Low-frequency effects	Left wide	Right wide

Available in iOS 4.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLayoutTag_EAC3_7_1_E`
A Blu-ray Disc audio layout for Enhanced AC-3, also known as Dolby Digital Plus.

Left	Center	Right	Left Surround	Right Surround	Low-frequency effects	Vertical height left	Vertical height right

Available in iOS 4.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLayoutTag_EAC3_7_1_F`
A Blu-ray Disc audio layout for Enhanced AC-3, also known as Dolby Digital Plus.

Left	Center	Right	Left Surround	Right Surround	Low-frequency effects	Center surround	Top surround

Available in iOS 4.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLayoutTag_EAC3_7_1_G`
A Blu-ray Disc audio layout for Enhanced AC-3, also known as Dolby Digital Plus.

Left	Center	Right	Left Surround	Right Surround	Low-frequency effects	Center surround	Vertical height center

Available in iOS 4.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLayoutTag_EAC3_7_1_H`
A Blu-ray Disc audio layout for Enhanced AC-3, also known as Dolby Digital Plus.

Left	Center	Right	Left Surround	Right Surround	Low-frequency effects	Top surround	Vertical height center

Available in iOS 4.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLayoutTag_DTS_3_1`
A Blu-ray Disc audio layout, defined by DTS (Digital Theater Systems Ltd.).

Center	Left	Right	Low-frequency effects

Available in iOS 4.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLayoutTag_DTS_4_1`
A Blu-ray Disc audio layout, defined by DTS (Digital Theater Systems Ltd.).

Center	Left	Right	Center surround	Low-frequency effects

Available in iOS 4.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLayoutTag_DTS_6_0_A`
A Blu-ray Disc audio layout, defined by DTS (Digital Theater Systems Ltd.).

Left center	Right center	Left	Right	Left surround	Right surround

Available in iOS 4.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLayoutTag_DTS_6_0_B`
A Blu-ray Disc audio layout, defined by DTS (Digital Theater Systems Ltd.).

Center	Left	Right	Left rear surround	Right rear surround	Top surround

Available in iOS 4.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLayoutTag_DTS_6_0_C`
A Blu-ray Disc audio layout, defined by DTS (Digital Theater Systems Ltd.).

Center	Center surround	Left	Right	Left rear surround	Right rear surround

Available in iOS 4.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLayoutTag_DTS_6_1_A`
A Blu-ray Disc audio layout, defined by DTS (Digital Theater Systems Ltd.).

Left center	Right center	Left	Right	Left surround	Right surround	Low-frequency effects

Available in iOS 4.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLayoutTag_DTS_6_1_B`
A Blu-ray Disc audio layout, defined by DTS (Digital Theater Systems Ltd.).

Center	Left	Right	Left rear surround	Right rear surround	Top surround	Low-frequency effects

Available in iOS 4.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLayoutTag_DTS_6_1_C`
A Blu-ray Disc audio layout, defined by DTS (Digital Theater Systems Ltd.).

Center	Center surround	Left	Right	Left rear surround	Right rear surround	Low-frequency effects

Available in iOS 4.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLayoutTag_DTS_6_1_D`
A Blu-ray Disc audio layout, defined by DTS (Digital Theater Systems Ltd.).

Center	Left	Right	Left surround	Right surround	Low-frequency effects	Center surround

Available in iOS 4.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLayoutTag_DTS_7_0`
A Blu-ray Disc audio layout, defined by DTS (Digital Theater Systems Ltd.).

Left center	Center	Right center	Left	Right	Left surround	Right surround

Available in iOS 4.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLayoutTag_DTS_7_1`
A Blu-ray Disc audio layout, defined by DTS (Digital Theater Systems Ltd.).

Left center	Center	Right center	Left	Right	Left surround	Right surround	Low-frequency effects

Available in iOS 4.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLayoutTag_DTS_8_0_A`
A Blu-ray Disc audio layout, defined by DTS (Digital Theater Systems Ltd.).

Left center	Right center	Left	Right	Left surround	Right surround	Left rear surround	Right rear surround

Available in iOS 4.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLayoutTag_DTS_8_0_B`
A Blu-ray Disc audio layout, defined by DTS (Digital Theater Systems Ltd.).

Left center	Center	Right	Left	Right	Left surround	Center surround	Right surround

Available in iOS 4.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLayoutTag_DTS_8_1_A`
A Blu-ray Disc audio layout, defined by DTS (Digital Theater Systems Ltd.).

Left center	Right center	Left	Right	Left surround	Right surround	Left rear surround	Right rear surround	Low-frequency effects

Available in iOS 4.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLayoutTag_DTS_8_1_B`
A Blu-ray Disc audio layout, defined by DTS (Digital Theater Systems Ltd.).

Left center	Center	Right center	Left	Right	Left surround	Center surround	Right surround	Low-frequency effects

Available in iOS 4.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLayoutTag_DiscreteInOrder`
Needs to be `OR`ed with the actual number of channels.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

`kAudioChannelLayoutTag_Unknown`
Needs to be `OR`ed with the actual number of channels.

Available in iOS 2.0 and later.

Declared in `CoreAudioTypes.h`.

Result Codes

This table lists result codes returned from the various C-based audio frameworks.

Result Code	Value	Description
`kAudio_UnimplementedError`	-4	An unimplemented system function was called. Available in iOS 4.0 and later.
`kAudio_ParamError`	-50	An error in the parameter list of the function. Available in iOS 4.0 and later.

Result Code	Value	Description
`kAudio_MemFullError`	-108	Not enough room in the heap zone. Available in iOS 4.0 and later.

Document Revision History

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

Date	Notes
2008-07-08	Updated for iOS.

Document Revision History