Audio File Services Reference

Audio & Video: Audio

2009-08-17

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

Simultaneously published in the United States and Canada.

Contents

Audio File Services Reference

Framework: AudioToolbox/AudioToolbox.h

Declared in AudioFile.h

Overview

This document describes Audio File Services, a C programming interface that enables you to read or write a wide variety of audio data to or from disk or a memory buffer.

With Audio File Services you can:

- Create, initialize, open, and close audio files
- Read and write audio files
- Optimize audio files
- Work with user data and global information

Functions by Task

Creating and Initializing Audio Files

`AudioFileCreateWithURL` (page 10)
Creates a new audio file, or initializes an existing file, specified by a URL.

`AudioFileInitializeWithCallbacks` (page 16)
Deletes the content of an existing file and assigns callbacks to the audio file object.

Opening and Closing Audio Files

`AudioFileOpenURL` (page 18)
Open an existing audio file specified by a URL.

`AudioFileOpenWithCallbacks` (page 19)
Opens an existing file with callbacks you provide.

`AudioFileClose` (page 7)
Closes an audio file.

Reading and Writing Audio Files

`AudioFileReadBytes` (page 20)
Reads bytes of audio data from an audio file.

`AudioFileWriteBytes` (page 26)
Writes bytes of audio data to an audio file.

`AudioFileReadPacketData` (page 21)
Reads packets of audio data from an audio file.

`AudioFileReadPackets` (page 22)
Reads a fixed duration of audio data from an audio file.

`AudioFileWritePackets` (page 27)
Writes packets of audio data to an audio data file.

Getting and Setting Audio File Properties

`AudioFileGetProperty` (page 12)
Gets the value of an audio file property.

`AudioFileGetPropertyInfo` (page 13)
Gets information about an audio file property, including the size of the property value and whether the value is writable.

`AudioFileSetProperty` (page 24)
Sets the value of an audio file property

Working with User Data

`AudioFileCountUserData` (page 8)
Gets the number of user data items with a specified ID in a file.

`AudioFileGetUserDataSize` (page 15)
Gets the size of a user data item in an audio file.

`AudioFileRemoveUserData` (page 24)
Removes a user data item from an audio file.

`AudioFileSetUserData` (page 25)
Sets a user data item in an audio file.

`AudioFileGetUserData` (page 14)
Gets a chunk from an audio file.

Working with Global Information

`AudioFileGetGlobalInfoSize` (page 11)
Gets the size of a global audio file property.

`AudioFileGetGlobalInfo` (page 10)
Copies the value of a global property into a buffer.

Optimizing Audio Files

`AudioFileOptimize` (page 19)
Consolidates audio data and performs other internal optimizations of the file structure.

Audio File Macros

`NumBytesToNumAudioFileMarkers` (page 29)
A macro that returns the number of audio file markers represented by a specified number of bytes.

`NumAudioFileMarkersToNumBytes` (page 29)
A macro that returns the number of bytes corresponding to a specified number of audio file markers.

`NextAudioFileRegion` (page 28)
Finds the next audio file region in a region list.

Deprecated Functions

`AudioFileCreate` (page 8)
(Deprecated. Deprecated. Use `AudioFileCreateWithURL` (page 10) instead.)

`AudioFileInitialize` (page 15)
(Deprecated. Deprecated. Use `AudioFileInitializeWithCallbacks` (page 16) instead.)

`AudioFileOpen` (page 17)
(Deprecated. Deprecated. Use `AudioFileOpenURL` (page 18) instead.)

Functions

AudioFileClose

Closes an audio file.

```
OSStatus AudioFileClose (
   AudioFileID inAudioFile
);
```

Parameters

inAudioFile
The file you want to close.

Return Value
A result code. See "Audio File Result Codes" (page 55).

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

See Also
`AudioFileOpenURL` (page 18)
`AudioFileOpenWithCallbacks` (page 19)

Related Sample Code
AudioQueueTools
ConvertFile
MixMash
QTAudioContextInsert
QTAudioExtractionPanel

Declared In
`AudioFile.h`

AudioFileCountUserData

Gets the number of user data items with a specified ID in a file.

```
OSStatus AudioFileCountUserData (
    AudioFileID inAudioFile,
    UInt32      inUserDataID,
    UInt32      *outNumberItems
);
```

Parameters

`inAudioFile`
The audio file whose user data items are to be counted.

`inUserDataID`
The four-character code (such as `COMM`) of the user data item.

`outNumberItems`
On output, a pointer to the number of user data items of this type in the file.

Return Value
A result code. See "Audio File Result Codes" (page 55).

Discussion
In the context of this function, *user data* refers to chunks in AIFF, CAF, and WAVE files, to resources in Sound Designer II files, and possibly to other types of information in other files.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`AudioFile.h`

AudioFileCreate

(Deprecated. Deprecated. Use `AudioFileCreateWithURL` (page 10) instead.)

```
OSStatus AudioFileCreate (
    const struct FSRef               *inParentRef,
    CFStringRef                      inFileName,
    AudioFileTypeID                  inFileType,
    const AudioStreamBasicDescription *inFormat,
    UInt32                           inFlags,
    struct FSRef                     *outNewFileRef,
    AudioFileID                      *outAudioFile
);
```

Parameters

inParentRef

A pointer to the directory where the new file should be created.

inFileName

The name of the file to be created.

inFileType

The type of audio file to create. See "Built-In Audio File Types" (page 40) for constants that can be used.

inFormat

A pointer to the structure that describes the format of the data.

inFlags

Relevant flags for creating or opening the file. Currently set to 0.

outNewFileRef

On output, a pointer to the location of the newly created file.

outAudioFile

On output, a pointer to the newly created audio file.

Return Value

A result code. See "Audio File Result Codes" (page 55).

Discussion

This deprecated function uses an FSRef type rather than the CFURLRef type used by the AudioFileCreateWithURL (page 10) function.

Availability

Available in Mac OS X v10.2 and later.

Deprecated in Mac OS X v10.6.

See Also

AudioFileCreateWithURL (page 10)

Related Sample Code

ExtractMovieAudioToAIFF

QTAudioContextInsert

QTAudioExtractionPanel

QTExtractAndConvertToAIFF

Declared In

AudioFile.h

AudioFileCreateWithURL

Creates a new audio file, or initializes an existing file, specified by a URL.

```
OSStatus AudioFileCreateWithURL (
    CFURLRef                          inFileRef,
    AudioFileTypeID                   inFileType,
    const AudioStreamBasicDescription *inFormat,
    UInt32                            inFlags,
    AudioFileID                       *outAudioFile
);
```

Parameters

inFileRef

The fully specified path of the file to create or initialize.

inFileType

The type of audio file to create. See "Built-In Audio File Types" (page 40) for constants that can be used.

inFormat

A pointer to the structure that describes the format of the data.

inFlags

Relevant flags for creating or opening the file. If kAudioFileFlags_EraseFile (page 42) is set, it erases an existing file. If the flag is not set, the function fails fails if the URL is an existing file.

outAudioFile

On output, a pointer to a newly created or initialized file.

Return Value

A result code. See "Audio File Result Codes" (page 55).

Discussion

This function uses a CFURLRef type rather than the FSRef type used by the deprecated AudioFileCreate (page 8) function.

Availability

Available in Mac OS X v10.5 and later.

Related Sample Code

AudioQueueTools

ConvertFile

Declared In

AudioFile.h

AudioFileGetGlobalInfo

Copies the value of a global property into a buffer.

```
OSStatus AudioFileGetGlobalInfo (
   AudioFilePropertyID inPropertyID,
   UInt32              inSpecifierSize,
   void                *inSpecifier,
   UInt32              *ioDataSize,
   void                *outPropertyData
);
```

Parameters

inPropertyID

The property whose value you want to get. For possible values, see "Audio File Global Info Properties" (page 53).

inSpecifierSize

The size of the specifier data.

inSpecifier

A pointer to a *specifier*, which, in this context, is a pointer to a buffer containing some data that is different for each property. The type of the data required is described in the description of each property.

ioDataSize

On input, a pointer to the size of the buffer specified in the *outPropertyData* parameter. On output, a pointer to the number of bytes written to the buffer.

outPropertyData

A pointer to the buffer in which to write the property data.

Return Value

A result code. See "Audio File Result Codes" (page 55).

Discussion

This function can be used to get information about the capabilities of Audio File Services data types, for example, to determine which file types can take which data formats, what file types are supported, what file type can hold a particular data type, and so forth. This function cannot be used to get information about the properties of particular files. So the properties whose information you are obtaining are global to the Audio File Services programming interface, not properties specific to any file.

Availability

Available in Mac OS X v10.3 and later.

Related Sample Code

AudioQueueTools

ConvertFile

Declared In

AudioFile.h

AudioFileGetGlobalInfoSize

Gets the size of a global audio file property.

```
OSStatus AudioFileGetGlobalInfoSize (
   AudioFilePropertyID inPropertyID,
   UInt32              inSpecifierSize,
   void                *inSpecifier,
   UInt32              *outDataSize
);
```

Parameters

inPropertyID

The property whose data size you want to get. For possible values, see "Audio File Global Info Properties" (page 53).

inSpecifier

A pointer to a *specifier* (a pointer to a buffer containing some data which is different for each property. The type of the data required is described in the description of each property.)

outDataSize

A pointer to the size in bytes of the current value of the property. To get the size of the property value, you need a buffer of this size.

Return Value

A result code. See "Audio File Result Codes" (page 55).

Discussion

This function can be used to get information about the capabilities of Audio File Service data types, for example, to determine which file types can take which data formats.

Availability

Available in Mac OS X v10.3 and later.

Related Sample Code

AudioQueueTools

Declared In

`AudioFile.h`

AudioFileGetProperty

Gets the value of an audio file property.

```
OSStatus AudioFileGetProperty (
    AudioFileID         inAudioFile,
    AudioFilePropertyID inPropertyID,
    UInt32              *ioDataSize,
    void                *outPropertyData
);
```

Parameters

inAudioFile

The audio file you want to obtain a property value from.

inPropertyID

The property whose value you want. See "Audio File Properties" (page 48) for possible values.

ioDataSize

On input, the size of the buffer passed in the `outPropertyData` parameter. On output, the number of bytes written to the buffer. Use the `AudioFileGetPropertyInfo` (page 13) function to obtain the size of the property value.

outPropertyData

On output, the value of the property specified in the *inPropertyID* parameter.

Return Value
A result code. See "Audio File Result Codes" (page 55).

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

See Also
`AudioFileGetPropertyInfo` (page 13)
`AudioFileSetProperty` (page 24)

Related Sample Code
AudioQueueTools
CocoaAUHost
ConvertFile
MixMash
PlayFile

Declared In
`AudioFile.h`

AudioFileGetPropertyInfo

Gets information about an audio file property, including the size of the property value and whether the value is writable.

```
OSStatus AudioFileGetPropertyInfo (
    AudioFileID         inAudioFile,
    AudioFilePropertyID inPropertyID,
    UInt32              *outDataSize,
    UInt32              *isWritable
);
```

Parameters

inAudioFile

The audio file you want to obtain property value information from.

inPropertyID

The property whose value information you want. See "Audio File Properties" (page 48) for possible values.

outDataSize

On output, the size in bytes of the property value.

isWritable

On output, equals `1` if the property is writable, or `0` if it is read-only.

Return Value
A result code. See "Audio File Result Codes" (page 55).

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

See Also
AudioFileGetProperty (page 12)

Related Sample Code
AudioQueueTools
ConvertFile
PlayFile

Declared In
AudioFile.h

AudioFileGetUserData

Gets a chunk from an audio file.

```
OSStatus AudioFileGetUserData (
   AudioFileID inAudioFile,
   UInt32      inUserDataID,
   UInt32      inIndex,
   UInt32      *ioUserDataSize,
   void        *outUserData
);
```

Parameters

inAudioFile
The audio file whose chunk you want to get.

inUserDataID
The four-character code of the designated chunk.

inIndex
An index specifying which chunk with the four-character code specified in the inUserDataID parameter you want to query.

ioUserDataSize
On input, a pointer to the size of the buffer containing the designated chunk. On output, a pointer to the size of bytes copied to the buffer.

outUserData
A pointer to a buffer in which to copy the chunk data.

Return Value
A result code. See "Audio File Result Codes" (page 55).

Availability
Available in Mac OS X v10.4 and later.

See Also
AudioFileSetUserData (page 25)

Declared In
AudioFile.h

AudioFileGetUserDataSize

Gets the size of a user data item in an audio file.

```
OSStatus AudioFileGetUserDataSize (
    AudioFileID inAudioFile,
    UInt32      inUserDataID,
    UInt32      inIndex,
    UInt32      *outUserDataSize
);
```

Parameters

inAudioFile

The audio file whose user data item size you want.

inUserDataID

The four-character code of the designated user data item.

inIndex

An index specifying which user data item with the four-character code specified in the `inUserDataID` parameter you want to query.

outUserDataSize

On output, a pointer the size of the user data item.

Return Value

A result code. See "Audio File Result Codes" (page 55).

Discussion

In the context of this function, *user data* refers to chunks in AIFF, CAF, and WAVE files, to resources in Sound Designer II files, and possibly to other types of information in other files.

Availability

Available in Mac OS X v10.4 and later.

Declared In

`AudioFile.h`

AudioFileInitialize

(Deprecated. Deprecated. Use `AudioFileInitializeWithCallbacks` (page 16) instead.)

```
OSStatus AudioFileInitialize (
    const struct FSRef               *inFileRef,
    AudioFileTypeID                  inFileType,
    const AudioStreamBasicDescription *inFormat,
    UInt32                           inFlags,
    AudioFileID                      *outAudioFile
);
```

Parameters

inFileRef

A pointer to the audio file you want to initialize.

inFileType

The type of audio file to initialize the file to.

inFormat
A pointer to the structure that describes the format of the data.

inFlags
Flags for creating or opening the file. Currently set to `0`.

outAudioFile
On output, a pointer to the newly created audio file.

Return Value
A result code. See "Audio File Result Codes" (page 55).

Discussion
This deprecated function deletes the content of an existing audio file to let you write over it.

Availability
Available in Mac OS X v10.2 and later.
Deprecated in Mac OS X v10.6.

Declared In
`AudioFile.h`

AudioFileInitializeWithCallbacks

Deletes the content of an existing file and assigns callbacks to the audio file object.

```
OSStatus AudioFileInitializeWithCallbacks (
   void                              *inClientData,
   AudioFile_ReadProc                inReadFunc,
   AudioFile_WriteProc               inWriteFunc,
   AudioFile_GetSizeProc             inGetSizeFunc,
   AudioFile_SetSizeProc             inSetSizeFunc,
   AudioFileTypeID                   inFileType,
   const AudioStreamBasicDescription *inFormat,
   UInt32                            inFlags,
   AudioFileID                       *outAudioFile
);
```

Parameters

inClientData
A pointer to a constant passed to your callbacks. Th constant should contain any information you use to manage the state for reading data from the file.

inReadFunc
A callback function invoked when the audio file object wants to read data.

inWriteFunc
A callback function invoked when the audio file object wants to write data.

inGetSizeFunc
A callback function invoked when the audio file object wants to know the size of the file.

inSetSizeFunc
A callback function invoked when the audio file object wants to set the size of the file.

inFileType
The type of audio file to initialize

inFormat
The format for the the audio data in the file.

inFlags
Flags for creating or opening the file. Set to `0`.

outAudioFile
On output, a pointer to the newly initialized audio file.

Return Value
A result code. See "Audio File Result Codes" (page 55).

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

See Also
`AudioFileOpenWithCallbacks` (page 19)

Declared In
`AudioFile.h`

AudioFileOpen

(Deprecated. Deprecated. Use `AudioFileOpenURL` (page 18) instead.)

```
OSStatus AudioFileOpen (
    const struct FSRef *inFileRef,
    SInt8              inPermissions,
    AudioFileTypeID    inFileTypeHint,
    AudioFileID        *outAudioFile
);
```

Parameters

inFileRef
A pointer to the audio file you want to open.

inPermissions
The read-write permissions you want to assign to the file. Use the permission constants in "Audio File Permission Flags" (page 42).

inFileTypeHint
A hint to indicate the file type of the designated file. For files without filename extensions and with types not easily or uniquely determined from the data (such as ADTS,AC3), use this hint to indicate the file type. Otherwise, pass `0`. Only use this hint in OS X versions 10.3.1 or greater. In all earlier versions, any attempt to open these files fails.

outAudioFile
On output, a pointer to the newly opened file.

Return Value
A result code. See "Audio File Result Codes" (page 55).

Discussion
This deprecated function opens an existing audio file specified by a file system reference.

Availability
Available in Mac OS X v10.2 and later.
Deprecated in Mac OS X v10.6.

See Also
AudioFileOpenURL (page 18)

Related Sample Code
CocoaAUHost
MixMash

Declared In
AudioFile.h

AudioFileOpenURL

Open an existing audio file specified by a URL.

```
OSStatus AudioFileOpenURL (
   CFURLRef        inFileRef,
   SInt8           inPermissions,
   AudioFileTypeID inFileTypeHint,
   AudioFileID     *outAudioFile
);
```

Parameters

inFileRef

The URL of an existing audio file.

inPermissions

The read-write permissions you want to assign to the file. Use the permission constants in "Audio File Permission Flags" (page 42).

inFileTypeHint

A hint for the file type of the designated file. For files without filename extensions and with types not easily or uniquely determined from the data (such as ADTS or AC3), use this hint to indicate the file type. Otherwise, pass 0. Only use this hint in OS X versions 10.3.1 or greater. In all earlier versions, any attempt to open these files fails.

outAudioFile

On output, a pointer to the newly opened audio file.

Return Value
A result code. See "Audio File Result Codes" (page 55).

Availability
Available in Mac OS X v10.5 and later.

See Also
AudioFileOpenWithCallbacks (page 19)
AudioFileClose (page 7)

Related Sample Code
AudioQueueTools
ConvertFile
PlayFile

Declared In
AudioFile.h

AudioFileOpenWithCallbacks

Opens an existing file with callbacks you provide.

```
OSStatus AudioFileOpenWithCallbacks (
   void                  *inClientData,
   AudioFile_ReadProc    inReadFunc,
   AudioFile_WriteProc   inWriteFunc,
   AudioFile_GetSizeProc inGetSizeFunc,
   AudioFile_SetSizeProc inSetSizeFunc,
   AudioFileTypeID       inFileTypeHint,
   AudioFileID           *outAudioFile
);
```

Parameters

inClientData

A pointer to a constant passed to your callbacks. The constant should contain any information you use to manage the state for reading data from the file.

inReadFunc

A callback function invoked when the audio file object wants to read data.

inWriteFunc

A callback function called when the audio file object wants to write data.

inGetSizeFunc

A callback function called when the audio file object wants to know the file size.

inSetSizeFunc

A callback function called when the audio file object wants to set the file size.

inFileTypeHint

A hint about the type of the designated file. For files with no filename extension and without a type easily or uniquely determined from the data (ADTS,AC3), use this hint to indicate the file type. Otherwise, pass `0` for this parameter. The hint is only available on OS X versions 10.3.1 or greater. In versions prior to OS X 10.3.1, opening files such files fails.

outAudioFile

On output, a pointer to the newly opened file.

Return Value

A result code. See "Audio File Result Codes" (page 55).

Availability

Available in Mac OS X v10.3 and later.

See Also

`AudioFileClose` (page 7)

`AudioFileInitializeWithCallbacks` (page 16)

`AudioFileOpenURL` (page 18)

Declared In

`AudioFile.h`

AudioFileOptimize

Consolidates audio data and performs other internal optimizations of the file structure.

```
OSStatus AudioFileOptimize (
   AudioFileID inAudioFile
);
```

Parameters

inAudioFile

The audio file you want to optimize.

Return Value

A result code. See "Audio File Result Codes" (page 55).

Discussion

This function optimizes the file so additional audio information can be appended to the existing data. Typically, this function consolidates the file's audio data at the end of the file. This improves performance, such as when writing additional data to the file.

Do not use this potentially expensive and time-consuming operation during time-critical operations. Instead, use the `kAudioFilePropertyIsOptimized` (page 49) property to check the optimization state of a file. You can then optimize when it won't adversely affect your application.

Availability

Available in Mac OS X v10.2 and later.

Declared In

`AudioFile.h`

AudioFileReadBytes

Reads bytes of audio data from an audio file.

```
OSStatus AudioFileReadBytes (
    AudioFileID inAudioFile,
    Boolean   inUseCache,
    SInt64    inStartingByte,
    UInt32    *ioNumBytes,
    void      *outBuffer
);
```

Parameters

inAudioFile

The audio file whose bytes of audio data you want to read.

inUseCache

Set to `true` if you want to cache the data. You should cache reads and writes if you read or write the same portion of a file multiple times. To request that the data not be cached, if possible, set to `false`. You should not cache reads and writes if you read or write data from a file only once.

inStartingByte

The byte offset of the audio data you want to be returned.

ioNumBytes

On input, a pointer to the number of bytes to read. On output, a pointer to the number of bytes actually read.

outBuffer

A pointer to user-allocated memory large enough for the requested bytes.

Return Value
A result code. See "Audio File Result Codes" (page 55).

Discussion
In most cases, you should use `AudioFileReadPackets` (page 22) instead of this function.

This function returns `eofErr` when the read operation encounters the end of the file. Note that Audio File Services only reads one 32-bit chunk of a file at a time.

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

See Also
`AudioFileWriteBytes` (page 26)

Declared In
`AudioFile.h`

AudioFileReadPacketData

Reads packets of audio data from an audio file.

```
OSStatus AudioFileReadPacketData (
    AudioFileID                     inAudioFile,
    Boolean                         inUseCache,
    UInt32                          *ioNumBytes,
    AudioStreamPacketDescription    *outPacketDescriptions,
    SInt64                          inStartingPacket,
    UInt32                          *ioNumPackets,
    void                            *outBuffer
);
```

Parameters

inAudioFile
The audio file whose audio packets you want to read.

inUseCache
Set to `true` to cache the data. Otherwise, set to `false`.

ioNumBytes
On input, the size of the *outBuffer* parameter, in bytes. On output, the number of bytes actually read.

You will see a difference in the input and output values if the byte size for the number of packets you request in the *ioNumPackets* parameter is smaller than the buffer size you pass in the *outBuffer* parameter. In this case, the output value for this parameter is smaller than its input value.

outPacketDescriptions
On output, an array of packet descriptions for the packets that were read. The array that you pass in this parameter must be large enough to accommodate descriptions for the number of packets requested in the *ioNumPackets* parameter.

This parameter applies only to variable bit-rate data. If the file being read contains constant bit-rate (CBR) data, such as linear PCM, this parameter does not get filled. Pass `NULL` if the file's data format is CBR.

inStartingPacket
The packet index of the first packet you want to read.

ioNumPackets
On input, the number of packets to read. On output, the number of packets actually read.

outBuffer
Memory that you allocate to hold the read packets. Determine an appropriate size by multiplying the number of packets requested (in the *ioNumPackets* parameter) by the typical packet size for the audio data in the file. For uncompressed audio formats, a packet is equal to a frame.

Return Value
A result code. See "Audio File Result Codes" (page 55).

Discussion
Using this function is memory efficient when reading variable bit-rate (VBR) audio data, whose packet sizes can vary for a given duration of sound.

If the buffer you provide in the *outBuffer* parameter is too small to hold the packets you request in *ioNumPackets*, the output values of *ioNumPackets* and *ioNumBytes* are reduced to reflect the packets that were placed into the buffer. You also see a difference in the input and output values for *ioNumPackets* when this function has reached the end of the file you are reading. In this case, the output value for this parameter is smaller than its input value.

This function is more efficient than `AudioFileReadPackets` (page 22) when reading compressed file formats that do not have packet tables, such as MP3 or ADTS. This function is a good choice for reading either CBR (constant bit-rate) or VBR data if you do not need to read a fixed duration of audio. If you do need to read a fixed duration of audio, whether CBR or VBR, use `AudioFileReadPackets` instead.

Audio File Services reads one 32-bit chunk of a file at a time.

Availability
Available in Mac OS X v10.6 and later.

See Also
`AudioFileWritePackets` (page 27)

Declared In
`AudioFile.h`

AudioFileReadPackets

Reads a fixed duration of audio data from an audio file.

```
OSStatus AudioFileReadPackets (
    AudioFileID                  inAudioFile,
    Boolean                      inUseCache,
    UInt32                       *outNumBytes,
    AudioStreamPacketDescription *outPacketDescriptions,
    SInt64                       inStartingPacket,
    UInt32                       *ioNumPackets,
    void                         *outBuffer
);
```

Parameters

inAudioFile

The audio file whose audio packets you want to read.

inUseCache

Set to `true` to cache the data. Otherwise, set to `false`.

outNumBytes

On output, the number of bytes actually read.

outPacketDescriptions

On output, an array of packet descriptions for the packets that were read. The array that you pass must be large enough to accommodate descriptions for the number of packets requested in the *ioNumPackets* parameter.

This parameter applies only to variable bit-rate data. If the file being read contains constant bit-rate (CBR) data, such as linear PCM, this parameter does not get filled. Pass `NULL` if the file's data format is CBR.

inStartingPacket

The packet index of the first packet you want to read.

ioNumPackets

On input, the number of packets to read. On output, the number of packets actually read.

You will see a difference in the input and output values when this function has reached the end of the file you are reading. In this case, the output value for this parameter is smaller than its input value.

outBuffer

Memory that you allocate to hold the read packets. Determine an appropriate size by multiplying the number of packets requested (in the *ioNumPackets* parameter) by the maximum (or upper bound for) packet size of the audio file. For uncompressed audio formats, a packet is equal to a frame.

Return Value

A result code. See "Audio File Result Codes" (page 55).

Discussion

If you do not need to read a fixed duration of audio data, but rather want to use your memory buffer most efficiently, use `AudioFileReadPacketData` (page 21) instead of this function.

When reading variable bit-rate (VBR) audio data, using this function requires that you allocate more memory than you would for the `AudioFileReadPacketData` function. See the descriptions for the *outBuffer* parameter in each of these two functions.

In addition, this function is less efficient than `AudioFileReadPacketData` when reading compressed file formats that do not have packet tables, such as MP3 or ADTS. Use this function only when you need to read a fixed duration of audio data, or when you are reading only uncompressed audio.

Audio File Services reads one 32-bit chunk of a file at a time.

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

See Also
`AudioFileWritePackets` (page 27)

Related Sample Code
AudioQueueTools
ConvertFile

Declared In
`AudioFile.h`

AudioFileRemoveUserData

Removes a user data item from an audio file.

```
OSStatus AudioFileRemoveUserData (
    AudioFileID inAudioFile,
    UInt32      inUserDataID,
    UInt32      inIndex
);
```

Parameters

`inAudioFile`
The audio file that contains the user data item you want to remove.

`inUserDataID`
The four-character code such as `COMM` of the user data item.

`inIndex`
An index specifying the chunk to remove. You use this parameter if the file contains more than one user data item with the four-character code specified in the `inUserDataID` parameter.

Return Value
A result code. See "Audio File Result Codes" (page 55).

Availability
Available in Mac OS X v10.5 and later.

Declared In
`AudioFile.h`

AudioFileSetProperty

Sets the value of an audio file property

```
OSStatus AudioFileSetProperty (
    AudioFileID         inAudioFile,
    AudioFilePropertyID inPropertyID,
    UInt32              inDataSize,
    const void          *inPropertyData
);
```

Parameters

inAudioFile

The audio file that you want to set a property value for.

inPropertyID

The property whose value you want to set. See "Audio File Properties" (page 48) for possible values. Use the `AudioFileGetPropertyInfo` (page 13) function to determine whether the property value is writable.

inDataSize

The size of the value you are passing in the *inPropertyData* parameter.

inPropertyData

The new value for the property.

Return Value

A result code. See "Audio File Result Codes" (page 55).

Availability

Available in Mac OS X v10.2 and later.

See Also

`AudioFileGetProperty` (page 12)

`AudioFileGetPropertyInfo` (page 13)

Related Sample Code

ConvertFile

ExtractMovieAudioToAIFF

QTAudioContextInsert

QTAudioExtractionPanel

QTExtractAndConvertToAIFF

Declared In

`AudioFile.h`

AudioFileSetUserData

Sets a user data item in an audio file.

```
OSStatus AudioFileSetUserData (
    AudioFileID inAudioFile,
    UInt32      inUserDataID,
    UInt32      inIndex,
    UInt32      inUserDataSize,
    const void  *inUserData
);
```

Parameters

inAudioFile

The audio file that you want to set a user data item in.

inUserDataID

The four-character code for the user data item.

inIndex

An index specifying the user data item you want to set. You use this parameter if the file contains more than one user data item with the four-character code specified in the `inUserDataID` parameter.

inUserDataSize

On input, the size of the data to copy. On output, the size of the bytes copied from the buffer.

inUserData

A pointer to a buffer from which to copy the user data.

Return Value

A result code. See "Audio File Result Codes" (page 55).

Availability

Available in Mac OS X v10.4 and later.

See Also

`AudioFileGetUserData` (page 14)

Declared In

`AudioFile.h`

AudioFileWriteBytes

Writes bytes of audio data to an audio file.

```
OSStatus AudioFileWriteBytes (
    AudioFileID inAudioFile,
    Boolean     inUseCache,
    SInt64      inStartingByte,
    UInt32      *ioNumBytes,
    const void  *inBuffer
);
```

Parameters

inAudioFile

The audio file to which you want to write bytes of data.

inUseCache

Set to `true` if you want to cache the data. Otherwise, set to `false`.

inStartingByte

The byte offset where the audio data should be written.

ioNumBytes

On input, a pointer the number of bytes to write. On output, a pointer to the number of bytes actually written.

inBuffer

A pointer to a buffer containing the bytes to be written.

Return Value
A result code. See "Audio File Result Codes" (page 55).

Discussion
In most cases, you should use `AudioFileWritePackets` (page 27) instead of this function.

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

See Also
`AudioFileReadBytes` (page 20)

Declared In
`AudioFile.h`

AudioFileWritePackets

Writes packets of audio data to an audio data file.

```
OSStatus AudioFileWritePackets (
    AudioFileID                     inAudioFile,
    Boolean                         inUseCache,
    UInt32                          inNumBytes,
    const AudioStreamPacketDescription *inPacketDescriptions,
    SInt64                          inStartingPacket,
    UInt32                          *ioNumPackets,
    const void                      *inBuffer
);
```

Parameters

inAudioFile

The audio file to write to.

inUseCache

Set to `true` if you want to cache the data. Otherwise, set to `false`.

inNumBytes

The number of bytes of audio data being written.

inPacketDescriptions

A pointer to an array of packet descriptions for the audio data. Not all formats require packet descriptions. If no packet descriptions are required, for instance, if you are writing CBR data, pass `NULL`.

inStartingPacket

The packet index for the placement of the first provided packet.

ioNumPackets

On input, a pointer to the number of packets to write. On output, a pointer to the number of packets actually written.

inBuffer

A pointer to user-allocated memory containing the new audio data to write to the audio data file.

Return Value
A result code. See "Audio File Result Codes" (page 55).

Discussion
For all uncompressed formats, this function equates packets with frames.

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

See Also
`AudioFileReadPackets` (page 22)

Related Sample Code
AudioQueueTools
ConvertFile
QTAudioContextInsert
QTAudioExtractionPanel
QTExtractAndConvertToAIFF

Declared In
`AudioFile.h`

NextAudioFileRegion

Finds the next audio file region in a region list.

```
#define NextAudioFileRegion (inAFRegionPtr) (
    (AudioFileRegion*) ((char*) (inAFRegionPtr) +
        offsetof(AudioFileRegion, mMarkers) +
            ((inAFRegionPtr)->mNumberMarkers) * sizeof (AudioFileMarker))
)
```

Parameters

inAFRegionPtr

A pointer to an audio file region in the region list.

Return Value
A pointer to the next region after the region pointed to by the `inAFRegionPtr` parameter. This value can be beyond the end of the list, so pay attention to the total number of regions in the list.

Discussion
Because audio file regions are of variable length, you cannot easily walk the list. Use this convenience function when you call the `AudioFileGetProperty` (page 12) function with the `kAudioFilePropertyRegionList` (page 51) property to walk through the list of regions returned.

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

See Also
`AudioFileSetProperty` (page 24)

Declared In
AudioFile.h

NumAudioFileMarkersToNumBytes

A macro that returns the number of bytes corresponding to a specified number of audio file markers.

```
#define NumAudioFileMarkersToNumBytes(inNumMarkers) (
    offsetof (AudioFileMarkerList, mMarkers) + (inNumMarkers) *
sizeof(AudioFileMarker)
)
```

Parameters

inNumMarkers

The number of audio file markers for which you wish to know the equivalent number of bytes.

Return Value
The number of bytes required to contain the specified number of audio file markers.

Discussion
Use this convenience function when you call the AudioFileSetProperty function with the kAudioFilePropertyMarkerList (page 51) property to calculate the size of buffer needed to hold a specific number of audio file markers.

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

See Also
AudioFileSetProperty (page 24)

Declared In
AudioFile.h

NumBytesToNumAudioFileMarkers

A macro that returns the number of audio file markers represented by a specified number of bytes.

```
#define NumBytesToNumAudioFileMarkers (inNumBytes) (
    (inNumBytes) < offsetof (AudioFileMarkerList, mMarkers[0]) ? 0 :
        ((inNumBytes) - offsetof (AudioFileMarkerList, mMarkers[0])) / sizeof
(AudioFileMarker)
)
```

Parameters

inNumBytes

The number of bytes for which you wish to know the equivalent number of audio file markers.

Return Value
The number of audio file markers that can be contained in the specified number of bytes.

Discussion
Use this convenience macro when you call the AudioFileGetProperty (page 12) function with the kAudioFilePropertyMarkerList (page 51) property to calculate the number of markers that will be returned.

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

See Also
`AudioFileGetProperty` (page 12)

Declared In
`AudioFile.h`

Callbacks by Task

Reading and Writing Audio File Data

`AudioFile_ReadProc` (page 31)
Reads audio data when used in conjunction with the `AudioFileOpenWithCallbacks` (page 19) or `AudioFileInitializeWithCallbacks` (page 16) functions.)

`AudioFile_WriteProc` (page 33)
A callback for writing file data when used in conjunction with the `AudioFileOpenWithCallbacks` (page 19) or `AudioFileCreateWithURL` (page 10) functions.

Getting the Size of Audio File Data

`AudioFile_GetSizeProc` (page 30)
Gets file data size.

`AudioFile_SetSizeProc` (page 32)
Sets file data size.

Callbacks

AudioFile_GetSizeProc

Gets file data size.

```
typedef SInt64 (*AudioFile_GetSizeProc)(
    void  *inClientData
);
```

If you name your function `MyAudioFile_GetSizeProc`, you would declare it like this:

```
 SInt64 MyAudioFile_GetSizeProc (
    void  *inClientData
);
```

Parameters

inClientData

A pointer to the client data as set in the `inClientData` parameter to the `AudioFileOpenWithCallbacks` (page 19) or `AudioFileInitializeWithCallbacks` (page 16) functions.

Return Value

The callback should return the size of the data.

Discussion

This callback gets invoked by an audio file object when it needs to get audio file data size. You pass this callback as a parameter when calling the `AudioFileOpenWithCallbacks` (page 19) and `AudioFileInitializeWithCallbacks` (page 16) functions.

Availability

Available in Mac OS X v10.3 and later.

Declared In

`AudioFile.h`

AudioFile_ReadProc

Reads audio data when used in conjunction with the `AudioFileOpenWithCallbacks` (page 19) or `AudioFileInitializeWithCallbacks` (page 16) functions.)

```
typedef OSStatus (*AudioFile_ReadProc) (
    void     *inClientData,
    SInt64   inPosition,
    UInt32   requestCount,
    void     *buffer,
    UInt32   *actualCount
);
```

If you name your function `MyAudioFile_ReadProc`, you would declare it like this:

```
OSStatus MyAudioFile_ReadProc (
    void     *inClientData,
    SInt64   inPosition,
    UInt32   requestCount,
    void     *buffer,
    UInt32   *actualCount
);
```

Parameters

inClientData

A pointer to the client data as set in the `inClientData` parameter to `AudioFileOpenWithCallbacks` (page 19) or `AudioFileInitializeWithCallbacks` (page 16).

inPosition

An offset into the data from which to read.

requestCount

The number of bytes to read.

buffer

A pointer to the buffer in which to put the data read.

actualCount

On output, the callback should set this parameter to a pointer to the number of bytes successfully read.

Return Value
A result code. See "Audio File Result Codes" (page 55).

Discussion
This callback function is called when Audio File Services needs to read data.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`AudioFile.h`

AudioFile_SetSizeProc

Sets file data size.

```
typedef SInt64 (*AudioFile_SetSizeProc)(
    void  *inClientData
);
```

If you name your function `MyAudioFile_SetSizeProc`, you would declare it like this:

```
 SInt64 MyAudioFile_SetSizeProc (
    void  *inClientData
);
```

Parameters

inClientData

A pointer to the client data as set in the `inClientData` parameter to the `AudioFileOpenWithCallbacks` (page 19) or `AudioFileInitializeWithCallbacks` (page 16) functions.

Return Value
The callback should return the size of the data.

Discussion
This callback gets invoked by an audio file object when it needs to set audio file data size. You pass this callback as a parameter when calling the `AudioFileOpenWithCallbacks` (page 19) and `AudioFileInitializeWithCallbacks` (page 16) functions.

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

Declared In
`AudioFile.h`

AudioFile_WriteProc

A callback for writing file data when used in conjunction with the `AudioFileOpenWithCallbacks` (page 19) or `AudioFileCreateWithURL` (page 10) functions.

```
typedef OSStatus (*AudioFile_WriteProc) (
    void          *inClientData,
    SInt64        inPosition,
    UInt32        requestCount,
    const void    *buffer,
    UInt32        *actualCount
);
```

If you named your function `MyAudioFile_WriteProc`, you would declare it like this:

```
OSStatus MyAudioFile_WriteProc (
    void    *inClientData,
    SInt64  inPosition,
    UInt32  requestCount,
    void    *buffer,
    UInt32  *actualCount
);
```

Parameters

inClientData

A pointer to the client data as set in the `inClientData` parameter to `AudioFileOpenWithCallbacks` (page 19) or`AudioFileInitializeWithCallbacks` (page 16).

inPosition

An offset into the data from which to read.

requestCount

The number of bytes to write.

buffer

A pointer to the buffer containing the data to write.

actualCount

Upon completion, the callback should set this to a pointer to the number of bytes successfully written.

Return Value

A result code. See "Audio File Result Codes" (page 55).

Discussion

This callback function is invoked when Audio File Services needs to write data.

Availability

Available in Mac OS X v10.3 and later.

Declared In

`AudioFile.h`

Data Types

AudioFileID

An opaque data type that represents an audio file object.

```
typedef struct OpaqueAudioFileID *AudioFileID;
```

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

Declared In
`AudioFile.h`

AudioFilePropertyID

An audio file property identifier.

```
typedef UInt32 AudioFilePropertyID;
```

Discussion
For a list of audio file properties, see "Audio File Properties" (page 48).

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

Declared In
`AudioFile.h`

AudioFile_SMPTE_Time

A data structure for describing SMPTE (Society of Motion Picture and Television Engineers) time.

```
struct AudioFile_SMPTE_Time {
    SInt8   mHours;
    UInt8   mMinutes;
    UInt8   mSeconds;
    UInt8   mFrames;
    UInt32  mSubFrameSampleOffset;
};
typedef struct AudioFile_SMPTE_Time AudioFile_SMPTE_Time;
```

Fields

`mHours`
The hours.

`mMinutes`
The minutes.

`mSeconds`
The seconds.

`mFrames`
The frames.

`mSubFrameSampleOffset`
The sample offset within a frame.

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

Declared In
`AudioFile.h`

AudioFileMarker

Annotates a position in an audio file.

```
struct AudioFileMarker {
    Float64 mFramePosition;
    CFStringRef           mName;
    SInt32                mMarkerID;
    AudioFile_SMPTE_Time  mSMPTETime;
    UInt32                mType;
    UInt16                mReserved;
    UInt16                mChannel;
};
typedef struct AudioFileMarker AudioFileMarker;
```

Fields
`mFramePosition`
The frame in the file, counting from the start of the audio data.

`mName`
The name of the marker.

`mMarkerID`
A unique ID for the marker.

`mSMPTETime`
The SMPTE time for this marker.

`mType`
The marker type.

`mReserved`
A reserved field. Set to `0`.

`mChannel`
The channel number referred to by the marker. Set to `0` if the marker applies to all channels.

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

Declared In
`AudioFile.h`

AudioFileMarkerList

A list of markers associated with an audio file, including their SMPTE time type, the number of markers, and the markers themselves.

```
struct AudioFileMarkerList {
    UInt32           mSMPTE_TimeType;
    UInt32           mNumberMarkers;
    AudioFileMarker  mMarkers[1];
};
typedef struct AudioFileMarkerList AudioFileMarkerList;
```

Fields

`mSMPTE_TimeType`
The SMPTE time type of the whole list of markers in an audio file.

`mNumberMarkers`
The number of markers in the list.

`mMarkers`
An array of `mNumberMarkers` elements, each of which is an audio file marker.

Availability

Available in Mac OS X v10.3 and later.

Declared In

`AudioFile.h`

AudioFileRegion

An audio file region specifies a segment of audio data.

```
struct AudioFileRegion {
    UInt32            mRegionID;
    CFStringRef       mName;
    UInt32            mFlags;
    UInt32            mNumberMarkers;
    AudioFileMarker   mMarkers[1];
};
typedef struct AudioFileRegion AudioFileRegion;
```

Fields

`mRegionID`
A unique ID associated with the audio file region.

`mName`
The name of the region.

`mFlags`
Audio File Services region flags. For details, see "Audio File Region Flags" (page 44).

`mNumberMarkers`
The number of markers in the array specified in the `mMarkers` parameter.

`mMarkers`
An array of `mNumberMarkers` elements describing where the data in the region starts. For details, see "Audio File Marker Types" (page 44).

Discussion
Typically, a region consists of at least two markers designating the beginning and end of the segment. Other markers might define additional meta information such as sync point.

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

Declared In
`AudioFile.h`

AudioFileRegionList

A list of the audio file regions in a file.

```
struct AudioFileRegionList {
    UInt32            mSMPTE_TimeType;
    UInt32            mNumberRegions;
    AudioFileRegion   mRegions[1];
};
typedef struct AudioFileRegionList AudioFileRegionList;
```

Fields

`mSMPTE_TimeType`

The SMPTE timing scheme used in the file. See Core Audio's `CAFFile.h` header file for the values used here. For more information, see *Core Audio Overview*.

`mNumberRegions`

The number of regions in the list specified in the `mRegions` parameter.

`mRegions`

A variable array of `mNumberRegions` elements containing a list of more than one audio file regions. For information on the `AudioFileRegion` data type, see `AudioFileRegion` (page 36).

Audio file markers are variable length, so this list cannot be accessed as an array. For details on the `AudioFileMarker` data type, see `AudioFileMarker` (page 35).

Use the `NextAudioFileRegion` (page 28) convenience macro for traversing the region list instead. This macro enables you to step to the next region in the data that Audio File Services returns.

Discussion
This structure is used by the `kAudioFilePropertyRegionList` (page 51) property.

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

Declared In
`AudioFile.h`

AudioFramePacketTranslation

A data structure used by the `kAudioFilePropertyPacketToFrame` (page 51) and `kAudioFilePropertyFrameToPacket` (page 51) properties.

```
struct AudioFramePacketTranslation {
    SInt64  mFrame;
    SInt64  mPacket;
    UInt32  mFrameOffsetInPacket;
};
typedef struct AudioFramePacketTranslation AudioFramePacketTranslation;
```

Fields

`mFrame`

A frame number.

`mPacket`

A packet number.

`mFrameOffsetInPacket`

A frame offset in a packet.

Availability

Available in Mac OS X v10.3 and later.

Declared In

`AudioFile.h`

AudioBytePacketTranslation

A data structure used by the `kAudioFilePropertyByteToPacket` (page 51) and `kAudioFilePropertyPacketToByte` (page 51) properties.

```
struct AudioFramePacketTranslation {
    SInt64 mByte;
    SInt64 mPacket;
    UInt32 mByteOffsetInPacket;
    UInt32 mFlags;
};
typedef struct AudioBytePacketTranslation AudioBytePacketTranslation;
```

Fields

`mByte`

A byte number.

`mPacket`

A packet number.

`mByteOffsetInPacket`

A byte offset in a packet.

`mFlags`

A flag from "Audio File Packet Translation Flags" (page 45).

Availability

Available in Mac OS X v10.6 and later.

Declared In

`AudioFile.h`

AudioFilePacketTableInfo

Contains information about the number of valid frames in a file and where they begin and end.

```
struct AudioFilePacketTableInfo {
    SInt64  mNumberValidFrames;
    SInt32  mPrimingFrames;
    SInt32  mRemainderFrames;
};
typedef struct AudioFilePacketTableInfo AudioFilePacketTableInfo;
```

Fields

`mNumberValidFrames`
: The number of valid frames in the file.

`mPrimingFrames`
: The number of invalid frames at the beginning of the file.

`mRemainderFrames`
: The number of invalid frames at the end of the file.

Discussion

Some data formats might have packets with contents that are not completely valid, but that represent priming or remainder frames not intended for playback. For example, a file with 100 packets of AAC is nominally 1024 * 100 = 102400 frames of data. However, the first 2112 frames might be priming frames.

A number of remainder frames might be added to pad out to a full packet of 1024 frames. Discard the priming and remainder frames.

The total number of packets in the file times the frames per packet (or counting each packet's frames individually for a variable frames per packet format) minus `mPrimingFrames`, minus `mRemainderFrames`, should equal `mNumberValidFrames`.

Availability

Available in Mac OS X v10.3 and later.

Declared In

`AudioFile.h`

AudioFileTypeandFormat ID

A specifier for the constant `kAudioFileGlobalInfo_AvailableStreamDescriptionsForFormat` (page 54).

```
struct AudioFileTypeAndFormatID {
    AudioFileTypeID  mFileType;
    UInt32           mFormatID;
};
typedef struct AudioFileTypeAndFormatID AudioFileTypeAndFormatID;
```

Fields

`mFileType`
: A four-character code for the file type (for instance, the `kAudioFileAIFFType` (page 40) type.

`mFormatID`
: A four-character code for the format ID such as `kAudioFormatLinearPCM`, `kAudioFormatMPEG4AAC`, and so forth. (See the `AudioFormat.h` header file for declarations.)

Discussion
This structure specifies a desired audio file type and data format ID so you can obtain a list of stream descriptions of available formats.

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

Declared In
`AudioFile.h`

Constants

Built-In Audio File Types

Operating system constants that indicate the type of file to be written or a hint about what type of file to expect from data provided.

```
enum {
    kAudioFileAIFFType            = 'AIFF',
    kAudioFileAIFCType            = 'AIFC',
    kAudioFileWAVEType            = 'WAVE',
    kAudioFileSoundDesigner2Type  = 'Sd2f',
    kAudioFileNextType            = 'NeXT',
    kAudioFileMP3Type             = 'MPG3',
    kAudioFileMP2Type             = 'MPG2',
    kAudioFileMP1Type             = 'MPG1',
    kAudioFileAC3Type             = 'ac-3',
    kAudioFileAAC_ADTSType        = 'adts',
    kAudioFileMPEG4Type           = 'mp4f',
    kAudioFileM4AType             = 'm4af',
    kAudioFileCAFType             = 'caff',
    kAudioFile3GPType             = '3gpp',
    kAudioFile3GP2Type            = '3gp2',
    kAudioFileAMRType             = 'amrf'
};
typedef UInt32 AudioFileTypeID;
```

Constants

`kAudioFileAIFFType`
An Audio Interchange File Format (AIFF) file.
Available in Mac OS X v10.2 and later.
Declared in `AudioFile.h`.

`kAudioFileAIFCType`
An Audio Interchange File Format Compressed (AIFF-C) file.
Available in Mac OS X v10.2 and later.
Declared in `AudioFile.h`.

`kAudioFileWAVEType`
A Microsoft WAVE file.
Available in Mac OS X v10.2 and later.
Declared in `AudioFile.h`.

`kAudioFileSoundDesigner2Type`
A Sound Designer II file.

Available in Mac OS X v10.2 and later.

Declared in `AudioFile.h`.

`kAudioFileNextType`
A NeXT or Sun Microsystems file.

Available in Mac OS X v10.3 and later.

Declared in `AudioFile.h`.

`kAudioFileMP3Type`
An MPEG Audio Layer 3 (`.mp3`) file.

Available in Mac OS X v10.3 and later.

Declared in `AudioFile.h`.

`kAudioFileMP2Type`
An MPEG Audio Layer 2 (`.mp2`) file.

Available in Mac OS X v10.5 and later.

Declared in `AudioFile.h`.

`kAudioFileMP1Type`
An MPEG Audio Layer 1 (`.mp1`) file.

Available in Mac OS X v10.5 and later.

Declared in `AudioFile.h`.

`kAudioFileAC3Type`
An AC-3 file.

Available in Mac OS X v10.3 and later.

Declared in `AudioFile.h`.

`kAudioFileAAC_ADTSType`
An Advanced Audio Coding (AAC) Audio Data Transport Stream (ADTS) file.

Available in Mac OS X v10.3 and later.

Declared in `AudioFile.h`.

`kAudioFileMPEG4Type`
An MPEG 4 file.

Available in Mac OS X v10.3 and later.

Declared in `AudioFile.h`.

`kAudioFileM4AType`
An M4A file.

Available in Mac OS X v10.3 and later.

Declared in `AudioFile.h`.

`kAudioFileCAFType`
A Core Audio File Format file.

Available in Mac OS X v10.3 and later.

Declared in `AudioFile.h`.

`kAudioFile3GPType`
A 3GPP file, suitable for video content on GSM mobile phones.

Available in Mac OS X v10.5 and later.

Declared in `AudioFile.h`.

`kAudioFile3GP2Type`
A 3GPP2 file, suitable for video content on CDMA mobile phones.

Available in Mac OS X v10.5 and later.

Declared in `AudioFile.h`.

`kAudioFileAMRType`
An AMR (Adaptive Multi-Rate) file suitable for compressed speech.

Available in Mac OS X v10.5 and later.

Declared in `AudioFile.h`.

Declared In
`AudioFile.h`

Audio File Creation Flags

Flags for use with the `AudioFileCreateWithURL` (page 10) and `AudioFileCreate` (page 8) functions.

```
enum {
    kAudioFileFlags_EraseFile                = 1
    kAudioFileFlags_DontPageAlignAudioData   = 2
};
```

Constants
`kAudioFileFlags_EraseFile`
If set, the `AudioFileCreateWithURL` function erases the contents of an existing file. If not set, then the function fails if the file already exists.

Available in Mac OS X v10.5 and later.

Declared in `AudioFile.h`.

`kAudioFileFlags_DontPageAlignAudioData`
Typically, the audio data in a file is page aligned. To make reading the file data as fast as possible, you can use page-aligned data to take advantage of optimized code paths in the file system. However, when space is at a premium, you might want to avoid the additional padding required to attain alignment. To do so, set this flag when calling `AudioFileCreate` (page 8) or `AudioFileCreateWithURL` (page 10).

Available in Mac OS X v10.5 and later.

Declared in `AudioFile.h`.

Declared In
`AudioFile.h`

Audio File Permission Flags

Flags for use with the `AudioFileOpenURL` (page 18) and `AudioFileOpen` (page 17) functions.

```
enum {
    kAudioFileReadPermission      = 0x01,
    kAudioFileWritePermission     = 0x02,
    kAudioFileReadWritePermission = 0x03
};
```

Constants

`kAudioFileReadPermission`
File is read-only.
Available in Mac OS X v10.6 and later.
Declared in `AudioFile.h`.

`kAudioFileWritePermission`
File is write-only.
Available in Mac OS X v10.6 and later.
Declared in `AudioFile.h`.

`kAudioFileReadWritePermission`
File has read-write permission.
Available in Mac OS X v10.6 and later.
Declared in `AudioFile.h`.

Declared In

`AudioFile.h`

Audio File Loop Direction Constants

The playback direction of a looped segment of an audio file.

```
enum {
    kAudioFileLoopDirection_NoLooping = 0,
    kAudioFileLoopDirection_Forward = 1,
    kAudioFileLoopDirection_ForwardAndBackward = 2,
    kAudioFileLoopDirection_Backward = 3
};
```

Constants

`kAudioFileLoopDirection_NoLooping`
The segment is not looped.
Available in Mac OS X v10.3 and later.
Declared in `AudioFile.h`.

`kAudioFileLoopDirection_Forward`
Play the segment forward.
Available in Mac OS X v10.3 and later.
Declared in `AudioFile.h`.

`kAudioFileLoopDirection_Backward`
Play the segment backward.
Available in Mac OS X v10.3 and later.
Declared in `AudioFile.h`.

`kAudioFileLoopDirection_ForwardAndBackward`
Play the segment forward and backward.

Available in Mac OS X v10.3 and later.

Declared in `AudioFile.h`.

Declared In
`AudioFile.h`

Audio File Marker Types

A type of marker within a file used in the `mType` field of the `AudioFileMarker` (page 35) structure.

```
enum {
    kAudioFileMarkerType_Generic = 0,
};
```

Constants
`kAudioFileMarkerType_Generic`
A generic marker.

Available in Mac OS X v10.3 and later.

Declared in `AudioFile.h`.

Declared In
`AudioFile.h`

Audio File Region Flags

Flags that specify a playback direction for an `AudioFileRegion` (page 36) structure.

```
enum {
    kAudioFileRegionFlag_LoopEnable = 1,
    kAudioFileRegionFlag_PlayForward = 2,
    kAudioFileRegionFlag_PlayBackward = 4
};
```

Constants
`kAudioFileRegionFlag_LoopEnable`
If set, the region is looped. You must set one or both of the remaining flags must also be set for the region to be looped.

Available in Mac OS X v10.3 and later.

Declared in `AudioFile.h`.

`kAudioFileRegionFlag_PlayForward`
If set, the region is played forward.

Available in Mac OS X v10.3 and later.

Declared in `AudioFile.h`.

`kAudioFileRegionFlag_PlayBackward`
If set, the region is played backward.

Available in Mac OS X v10.3 and later.

Declared in `AudioFile.h`.

Discussion
You can set one or more of these flags. For example, if both `kAudioFileRegionFlag_LoopEnable` and `kAudioFileRegionFlag_PlayForward` are set, the region plays as a forward loop. If only `kAudioFileRegionFlag_PlayForward` is set, the region is played forward once. if both `kAudioFileRegionFlag_PlayForward` and `kAudioFileRegionFlag_PlayBackward` are set, the region plays forward then backward, then forward.

Declared In
`AudioFile.h`

Audio File Packet Translation Flags

Flag used with the `mFlags` field of the `AudioBytePacketTranslation` (page 38) structure.

```
enum {
    kBytePacketTranslationFlag_IsEstimate = 1
};
```

Constants
`kBytePacketTranslationFlag_IsEstimate`
If set, the result value is an estimate.
Available in Mac OS X v10.6 and later.
Declared in `AudioFile.h`.

Info String Keys

Key values of properties to get and set using Audio File Services functions and provide a common way to get the same information out of several different kinds of files.

```
#define kAFInfoDictionary_Artist                        "artist"
#define kAFInfoDictionary_Album                         "album"
#define kAFInfoDictionary_Tempo                         "tempo"
#define kAFInfoDictionary_KeySignature                  "key signature"
#define kAFInfoDictionary_TimeSignature                 "time signature"
#define kAFInfoDictionary_TrackNumber                   "track number"
#define kAFInfoDictionary_Year                          "year"
#define kAFInfoDictionary_Composer                      "composer"
#define kAFInfoDictionary_Lyricist                      "lyricist"
#define kAFInfoDictionary_Genre                         "genre"
#define kAFInfoDictionary_Title                         "title"
#define kAFInfoDictionary_RecordedDate                  "recorded date"
#define kAFInfoDictionary_Comments                      "comments"
#define kAFInfoDictionary_Copyright                     "copyright"
#define kAFInfoDictionary_SourceEncoder                 "source encoder"
#define kAFInfoDictionary_EncodingApplication           "encoding application"
#define kAFInfoDictionary_NominalBitRate                "nominal bit rate"
#define kAFInfoDictionary_ChannelLayout                 "channel layout"
#define kAFInfoDictionary_ApproximateDurationInSeconds  "approximate duration in
                                                         seconds"
```

Constants

`kAFInfoDictionary_Artist`

An artist.

Available in Mac OS X v10.3 and later.

Declared in `AudioFile.h`.

`kAFInfoDictionary_Album`

An album.

Available in Mac OS X v10.3 and later.

Declared in `AudioFile.h`.

`kAFInfoDictionary_Tempo`

A tempo.

Available in Mac OS X v10.3 and later.

Declared in `AudioFile.h`.

`kAFInfoDictionary_KeySignature`

A key signature.

Available in Mac OS X v10.3 and later.

Declared in `AudioFile.h`.

`kAFInfoDictionary_TimeSignature`

A time signature.

Available in Mac OS X v10.3 and later.

Declared in `AudioFile.h`.

`kAFInfoDictionary_TrackNumber`

A track number.

Available in Mac OS X v10.3 and later.

Declared in `AudioFile.h`.

`kAFInfoDictionary_Year`
A year.
Available in Mac OS X v10.3 and later.
Declared in `AudioFile.h`.

`kAFInfoDictionary_Composer`
A composer.
Available in Mac OS X v10.3 and later.
Declared in `AudioFile.h`.

`kAFInfoDictionary_Lyricist`
A lyricist.
Available in Mac OS X v10.3 and later.
Declared in `AudioFile.h`.

`kAFInfoDictionary_Genre`
A genre.
Available in Mac OS X v10.3 and later.
Declared in `AudioFile.h`.

`kAFInfoDictionary_Title`
A title.
Available in Mac OS X v10.3 and later.
Declared in `AudioFile.h`.

`kAFInfoDictionary_RecordedDate`
A recorded date.
Available in Mac OS X v10.3 and later.
Declared in `AudioFile.h`.

`kAFInfoDictionary_Comments`
Comments.
Available in Mac OS X v10.3 and later.
Declared in `AudioFile.h`.

`kAFInfoDictionary_Copyright`
Copyright.
Available in Mac OS X v10.3 and later.
Declared in `AudioFile.h`.

`kAFInfoDictionary_SourceEncoder`
A source encoder.
Available in Mac OS X v10.3 and later.
Declared in `AudioFile.h`.

`kAFInfoDictionary_EncodingApplication`
An encoding application.
Available in Mac OS X v10.3 and later.
Declared in `AudioFile.h`.

`kAFInfoDictionary_NominalBitRate`
: A nominal bit rate.

  Available in Mac OS X v10.3 and later.

  Declared in `AudioFile.h`.

`kAFInfoDictionary_ChannelLayout`
: A channel layout.

  Available in Mac OS X v10.3 and later.

  Declared in `AudioFile.h`.

`kAFInfoDictionary_ApproximateDurationInSeconds`
: An approximate duration in seconds.

  Available in Mac OS X v10.3 and later.

  Declared in `AudioFile.h`.

Declared In
`AudioFile.h`

Audio File Properties

Properties used by the functions described in getting and setting pieces of data in audio files. See "Working with Global Information" (page 6) for details.

```
enum {
    kAudioFilePropertyFileFormat            = 'ffmt',
    kAudioFilePropertyDataFormat            = 'dfmt',
    kAudioFilePropertyIsOptimized           = 'optm',
    kAudioFilePropertyMagicCookieData       = 'mgic',
    kAudioFilePropertyAudioDataByteCount    = 'bcnt',
    kAudioFilePropertyAudioDataPacketCount  = 'pcnt',
    kAudioFilePropertyMaximumPacketSize     = 'psze',
    kAudioFilePropertyDataOffset            = 'doff',
    kAudioFilePropertyChannelLayout         = 'cmap',
    kAudioFilePropertyDeferSizeUpdates      = 'dszu',
    kAudioFilePropertyDataFormatName        = 'fnme',
    kAudioFilePropertyMarkerList            = 'mkls',
    kAudioFilePropertyRegionList            = 'rgls',
    kAudioFilePropertyPacketToFrame         = 'pkfr',
    kAudioFilePropertyFrameToPacket         = 'frpk',
    kAudioFilePropertyPacketToByte          = 'pkby',
    kAudioFilePropertyByteToPacket          = 'bypk',
    kAudioFilePropertyChunkIDs              = 'chid',
    kAudioFilePropertyInfoDictionary        = 'info',
    kAudioFilePropertyPacketTableInfo       = 'pnfo',
    kAudioFilePropertyFormatList            = 'flst',
    kAudioFilePropertyPacketSizeUpperBound  = 'pkub',
    kAudioFilePropertyReserveDuration       = 'rsrv',
    kAudioFilePropertyEstimatedDuration     = 'edur',
    kAudioFilePropertyBitRate               = 'brat',
    kAudioFilePropertyID3Tag                = 'id3t'
};
```

Constants

`kAudioFilePropertyFileFormat`

The format of the audio data file.

Available in Mac OS X v10.2 and later.

Declared in `AudioFile.h`.

`kAudioFilePropertyDataFormat`

An audio stream basic description containing the format of the audio data.

Available in Mac OS X v10.2 and later.

Declared in `AudioFile.h`.

`kAudioFilePropertyFormatList`

To support formats such as AAC SBR in which an encoded data stream can be decoded to multiple destination formats, this property returns an array of audio format list item values (declared in `AudioFormat.h`) of those formats. Typically, this returns an audio format list item with the same audio stream basic description returned by `kAudioFilePropertyDataFormat`.

Available in Mac OS X v10.5 and later.

Declared in `AudioFile.h`.

`kAudioFilePropertyIsOptimized`

Indicates whether a designated audio file has been optimized, that is, ready to start having sound data written to it. A value of `0` indicates the file needs to be optimized. A value of `1` indicates the file is currently optimized.

Available in Mac OS X v10.2 and later.

Declared in `AudioFile.h`.

`kAudioFilePropertyMagicCookieData`
A pointer to memory set up by the caller. Some file types require that a magic cookie be provided before packets can be written to an audio file. Set this property before you call `AudioFileWriteBytes` (page 26) or `AudioFileWritePackets` (page 27) if a magic cookie exists.

Available in Mac OS X v10.2 and later.

Declared in `AudioFile.h`.

`kAudioFilePropertyAudioDataByteCount`
Indicates the number of bytes of audio data in the designated file.

Available in Mac OS X v10.2 and later.

Declared in `AudioFile.h`.

`kAudioFilePropertyAudioDataPacketCount`
Indicates the number of packets of audio data in the designated file.

Available in Mac OS X v10.2 and later.

Declared in `AudioFile.h`.

`kAudioFilePropertyMaximumPacketSize`
Indicates the maximum size of a packet for the data in the designated file.

Available in Mac OS X v10.2 and later.

Declared in `AudioFile.h`.

`kAudioFilePropertyDataOffset`
Indicates the byte offset in the file of the designated audio data.

Available in Mac OS X v10.3 and later.

Declared in `AudioFile.h`.

`kAudioFilePropertyChannelLayout`
An audio channel layout structure.

Available in Mac OS X v10.3 and later.

Declared in `AudioFile.h`.

`kAudioFilePropertyDeferSizeUpdates`
The default value (`0`) always updates header. If set to `1`, updating the files sizes in the header is not performed every time data is written. Instead, the updating is deferred until the file has been read, optimized, or closed. This process is more efficient, but not as safe. If an application crashes before the size has been updated, the file might not be readable.

Available in Mac OS X v10.3 and later.

Declared in `AudioFile.h`.

`kAudioFilePropertyDataFormatName`
This constant is deprecated in Mac OS X v 10.5 and later. Do not use. Instead, use `kAudioFormatProperty_FormatName` (declared in the `AudioFormat.h` header file).

Available in Mac OS X v10.2 and later.

Declared in `AudioFile.h`.

`kAudioFilePropertyMarkerList`
: Accesses the list of markers defined in the file and returns an audio file marker list. The CF string referencing the returned structures must be released by the client. See the `NumBytesToNumAudioFileMarkers` (page 29) and `NumAudioFileMarkersToNumBytes` (page 29) functions to convert between audio file markers and the equivalent number of bytes.

Available in Mac OS X v10.3 and later.

Declared in `AudioFile.h`.

`kAudioFilePropertyRegionList`
: Accesses the list of regions defined in the file and returns an array of audio file region values. The CF string references in the returned structures must be released by the client. See the `NextAudioFileRegion` (page 28) function for a way to walk through the region list.

Available in Mac OS X v10.3 and later.

Declared in `AudioFile.h`.

`kAudioFilePropertyPacketToFrame`
: Passes an audio frame packet translation structure with the `mPacket` field filled out and returns the `mFrame` field. The `mFrameOffsetInPacket` field is ignored.

Available in Mac OS X v10.3 and later.

Declared in `AudioFile.h`.

`kAudioFilePropertyFrameToPacket`
: Passes an audio frame packet translation structure with the `mFrame` field filled out and returns the `mPacket` and `mFrameOffsetInPacket` fields.

Available in Mac OS X v10.3 and later.

Declared in `AudioFile.h`.

`kAudioFilePropertyPacketToByte`
: Passes an audio byte packet translation structure with the `mPacket` field filled out and returns the `mByte` field. The `mByteOffsetInPacket` field is ignored. If the value in the `mByte` field is an estimate then the `kBytePacketTranslationFlag_IsEstimate` flag is set in the `mFlags` field.

Available in Mac OS X v10.6 and later.

Declared in `AudioFile.h`.

`kAudioFilePropertyByteToPacket`
: Passes an audio byte packet translation structure with the `mByte` field filled out and returns the `mPacket` and `mByteOffsetInPacket` fields. If the value in the `mByte` field is an estimate then the `kBytePacketTranslationFlag_IsEstimate` flag is set in the `mFlags` field.

Available in Mac OS X v10.6 and later.

Declared in `AudioFile.h`.

`kAudioFilePropertyChunkIDs`
: Returns an array of four-character codes for each kind of chunk in the file.

Available in Mac OS X v10.3 and later.

Declared in `AudioFile.h`.

`kAudioFilePropertyInfoDictionary`
Returns a CF Dictionary with information about the data in the file. `AudioFileComponents` (declared in `AudioFileComponents.h`) are free to add keys to the dictionaries that they return for this property. The caller is responsible for releasing the `CFObject` (declared in the `AudioFileComponents.h` header file).

Available in Mac OS X v10.3 and later.

Declared in `AudioFile.h`.

`kAudioFilePropertyPacketTableInfo`
Gets or sets an audio file packet table information structure for its supporting file types. When setting the structure, the sum of the values of the `mNumberValidFrames`, `mPrimingFrames` and `mRemainderFrames` fields must be the same as the total number of frames in all packets. If not, a `paramErr` is returned. To ensure this result, get the value of the property and make sure the sum of the three values you set has the same sum as the three values you got.

Available in Mac OS X v10.3 and later.

Declared in `AudioFile.h`.

`kAudioFilePropertyPacketSizeUpperBound`
The theoretical maximum packet size in the file. This value is obtained without actually scanning the whole file to find the largest packet, as could happen with `kAudioFilePropertyMaximumPacketSize`.

Available in Mac OS X v10.5 and later.

Declared in `AudioFile.h`.

`kAudioFilePropertyReserveDuration`
The duration in seconds of the data expected to be written. Set this property before any data has been written to reserve space in the file header for a packet table and other information to appear before the audio data. Otherwise, the packet table might get written at the end of the file, preventing the file from being streamable.

Available in Mac OS X v10.5 and later.

Declared in `AudioFile.h`.

`kAudioFilePropertyEstimatedDuration`
An estimated duration in seconds. If this duration can be calculated without scanning the entire file, or all the audio data packets have been scanned, the value accurately reflects the duration of the audio data.

Available in Mac OS X v10.5 and later.

Declared in `AudioFile.h`.

`kAudioFilePropertyBitRate`
Returns the actual bit rate (number of audio data bits in the file divided by the duration of the file) for some file types, and the nominal bit rate (which bit rate the encoder was set to) for others.

Available in Mac OS X v10.5 and later.

Declared in `AudioFile.h`.

`kAudioFilePropertyID3Tag`
A `void*` value pointing to memory set up by your application to contain a fully formatted ID3 tag.

When setting, this property must be set before calling the `AudioFileWritePackets` (page 27) function. This property is gettable and settable when using ID3 version 2. It is gettable only for version ID3 version 1. A sound file's ID3 tag itself is not manipulated when getting or setting this property.

Available in Mac OS X v10.6 and later.

Declared in `AudioFile.h`.

Declared In
`AudioFile.h`

Audio File Global Info Properties

Properties used when "Working with Global Information" (page 6).

```
enum
{
    kAudioFileGlobalInfo_ReadableTypes                          = 'afrf',
    kAudioFileGlobalInfo_WritableTypes                          = 'afwf',
    kAudioFileGlobalInfo_FileTypeName                           = 'ftnm',
    kAudioFileGlobalInfo_AvailableStreamDescriptionsForFormat   = 'sdid',
    kAudioFileGlobalInfo_AvailableFormatIDs                     = 'fmid',

    kAudioFileGlobalInfo_AllExtensions                          = 'alxt',
    kAudioFileGlobalInfo_AllHFSTypeCodes                        = 'ahfs',
    kAudioFileGlobalInfo_AllUTIs                                = 'auti',
    kAudioFileGlobalInfo_AllMIMETypes                           = 'amim',

    kAudioFileGlobalInfo_ExtensionsForType                      = 'fext',
    kAudioFileGlobalInfo_HFSTypeCodesForType                    = 'fhfs',
    kAudioFileGlobalInfo_UTIsForType                            = 'futi',
    kAudioFileGlobalInfo_MIMETypesForType                       = 'fmim',

    kAudioFileGlobalInfo_TypesForMIMEType                      = 'tmim',
    kAudioFileGlobalInfo_TypesForUTI                           = 'tuti',
    kAudioFileGlobalInfo_TypesForHFSTypeCode                   = 'thfs',
    kAudioFileGlobalInfo_TypesForExtension                     = 'text'
};
```

Constants

`kAudioFileGlobalInfo_ReadableTypes`

No specifier needed. (A *specifier* in this context is a pointer to a buffer containing some data which is different for each property. The type of the data required is described in the description of each property.) Must be set to `NULL`. Returns an array of `UInt32` values containing the file types (such as AIFF, WAVE, and so forth) that can be opened for reading.

Available in Mac OS X v10.3 and later.

Declared in `AudioFile.h`.

`kAudioFileGlobalInfo_WritableTypes`

No specifier needed. Must be set to `NULL`. Returns an array of `UInt32` values containing the file types (such as AIFF, WAVE, and so forth) that can be opened for writing.

Available in Mac OS X v10.3 and later.

Declared in `AudioFile.h`.

`kAudioFileGlobalInfo_FileTypeName`

A pointer to an audio file type ID containing a file type. Returns a `CFString` containing the name for the file type.

Available in Mac OS X v10.3 and later.

Declared in `AudioFile.h`.

`kAudioFileGlobalInfo_AvailableFormatIDs`
A pointer to an audio file type ID containing a file type. Returns an array of format IDs for formats that can be read.

Available in Mac OS X v10.3 and later.

Declared in `AudioFile.h`.

`kAudioFileGlobalInfo_AvailableStreamDescriptionsForFormat`
A pointer to an audio file type and format ID structure. Returns an array of audio stream basic description structures, which contain all the formats for a particular file type and format ID. The audio stream basic description structures have the following fields filled in: `mFormatID`, `mFormatFlags`, and `mBitsPerChannel` for writing new files.

Available in Mac OS X v10.3 and later.

Declared in `AudioFile.h`.

`kAudioFileGlobalInfo_AllExtensions`
No specifier needed. Must be set to `NULL`. Returns a `CFArray` of `CFStrings` containing all recognized file extensions. You can use this array when creating an `NSOpenPanel` (declared in the AppKit's `NSOpenPanel.h` header file).

Available in Mac OS X v10.3 and later.

Declared in `AudioFile.h`.

`kAudioFileGlobalInfo_AllHFSTypeCodes`
No specifier needed. Must be set to `NULL`. Returns an array of HFS type codes containing all recognized HFS type codes. For more information on HFS type codes, see Audio Toolbox's `ExtendedAudioFile.h` header file.

Available in Mac OS X v10.3 and later.

Declared in `AudioFile.h`.

`kAudioFileGlobalInfo_AllUTIs`
No specifier needed. Must be set to `NULL`. Returns a `CFArray` of `CFString` of all UTIs (Universal Type Identifiers) recognized by Audio File Services. The caller is responsible for releasing the `CFArray`.

Available in Mac OS X v10.5 and later.

Declared in `AudioFile.h`.

`kAudioFileGlobalInfo_AllMIMETypes`
No specifier needed. Must be set to `NULL`. Returns a `CFArray` of CF strings of all MIME types are recognized by Audio File Services. The caller is responsible for releasing the `CFArray`.

Available in Mac OS X v10.5 and later.

Declared in `AudioFile.h`.

`kAudioFileGlobalInfo_ExtensionsForType`
A pointer to a audio file type ID containing a file type. Returns a `CFArray` of CF strings containing the recognized file extensions for the designated type.

Available in Mac OS X v10.3 and later.

Declared in `AudioFile.h`.

`kAudioFileGlobalInfo_HFSTypeCodesForType`
A pointer to an audio file type ID. Returns an array of HFS type codes corresponding to the designated file type. The first type in the array is the preferred one to use.

Available in Mac OS X v10.3 and later.

Declared in `AudioFile.h`.

`kAudioFileGlobalInfo_UTIsForType`
A pointer to an audio file type ID. Returns a `CFArray` of `CFString` of all Universal Type Identifiers recognized by the file type. The caller is responsible for releasing the `CFArray`.

Available in Mac OS X v10.5 and later.

Declared in `AudioFile.h`.

`kAudioFileGlobalInfo_MIMETypesForType`
A pointer to an audio file type ID. Returns a `CFArray` of `CFString` of all MIME types recognized by the designated file type. The caller is responsible for releasing the `CFArray`.

Available in Mac OS X v10.5 and later.

Declared in `AudioFile.h`.

`kAudioFileGlobalInfo_TypesForExtension`
A `CFStringRef` containing a file extension. Returns an array of all audio file type IDs that support the extension.

Available in Mac OS X v10.5 and later.

Declared in `AudioFile.h`.

`kAudioFileGlobalInfo_TypesForHFSTypeCode`
An `HFSTypeCode`. Returns an array of all audio file type IDs that support the designated `HFSTypeCode`.

Available in Mac OS X v10.5 and later.

Declared in `AudioFile.h`.

`kAudioFileGlobalInfo_TypesForUTI`
A `CFStringRef` containing a Universal Type Identifier. Returns an array of all audio file type IDs that support the UTI.

Available in Mac OS X v10.5 and later.

Declared in `AudioFile.h`.

`kAudioFileGlobalInfo_TypesForMIMEType`
A `CFStringRef` containing a MIME Type. Returns an array of all audio file type IDs that support the MIME type.

Available in Mac OS X v10.5 and later.

Declared in `AudioFile.h`.

Declared In
`AudioFile.h`

Result Codes

This table lists the result codes defined for Audio File Services.

Result Code	Value	Description
`noErr`	0	No error. Available in Mac OS X v10.0 and later.
`kAudioFileUnspecifiedError`	'wht?'	An unspecified error has occurred. Available in Mac OS X v10.2 and later.

Result Code	Value	Description
`kAudioFileUnsupportedFileTypeError`	'typ?'	The file type is not supported. Available in Mac OS X v10.2 and later.
`kAudioFileUnsupportedDataFormatError`	"fmt?'	The data format is not supported by this file type. Available in Mac OS X v10.2 and later.
`kAudioFileUnsupportedPropertyError`	'pty?'	The property is not supported. Available in Mac OS X v10.2 and later.
`kAudioFileBadPropertySizeError`	'!siz'	The size of the property data was not correct. Available in Mac OS X v10.2 and later.
`kAudioFilePermissionsError`	'prm?'	The operation violated the file permissions. For example, an attempt was made to write to a file opened with the `kAudioFileReadPermission` constant. Available in Mac OS X v10.2 and later.
`kAudioFileNotOptimizedError`	'optm'	The chunks following the audio data chunk are preventing the extension of the audio data chunk. To write more data, you must optimize the file. Available in Mac OS X v10.2 and later.
`kAudioFileInvalidChunkError`	'chk?'	Either the chunk does not exist in the file or it is not supported by the file. Available in Mac OS X v10.2 and later.
`kAudioFileDoesNotAllow64BitDataSizeError`	'off?'	The file offset was too large for the file type. The AIFF and WAVE file format types have 32-bit file size limits. Available in Mac OS X v10.2 and later.
`kAudioFileInvalidPacketOffsetError`	'pck?'	A packet offset was past the end of the file, or not at the end of the file when a VBR format was written, or a corrupt packet size was read when the packet table was built. Available in Mac OS X v10.3 and later.
`kAudioFileInvalidFileError`	'dta?'	The file is malformed, or otherwise not a valid instance of an audio file of its type. Available in Mac OS X v10.3 and later.

Result Code	Value	Description
kAudioFileOperationNotSupportedError	0x6F703F3F	The operation cannot be performed. For example, setting the kAudioFilePropertyAudioDataByteCount (page 50) constant to increase the size of the audio data in a file is not a supported operation. Write the data instead. Available in Mac OS X v10.3 and later.

Document Revision History

This table describes the changes to *Audio File Services Reference*.

Date	Notes
2009-08-17	Added descriptions for `kAudioFile3GPType` (page 42), `kAudioFile3GP2Type` (page 42), and `kAudioFileAMRType` (page 42) constants.
2008-11-17	Updated for iOS 2.2.
	Added description for `AudioFileReadPacketData` (page 21) function. Updated description for `AudioFileReadPackets` (page 22) function.
2008-09-09	Updated for iOS 2.1.
2008-07-08	Added new properties for packet to byte and byte to packet translations.
2007-05-29	New document describing a C programming interface for reading or writing audio files.

Document Revision History