Audio Codec Services Reference

Audio > Core Audio

2007-10-31

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

Simultaneously published in the United States and Canada.

Contents

Audio Codec Services Reference

Framework: AudioUnit.framework

Declared in AudioCodec.h

Overview

Audio codec objects translate audio data from one format to another. You use the interface described in this document to initialize, configure, and use audio codecs. In addition, because the functions in this interface are all implemented by audio codec objects, you need to be familiar with this reference if you are developing an audio codec.

There are three kinds of audio codec objects defined by Audio Codec Services:

- Decoder objects (`'adec'`) translate data that isn't in linear PCM into linear PCM formatted data.
- Encoder objects (`'aenc'`) translate linear PCM data into some other format.
- Unity codecs (`'acdc'`) translate between different varieties of the same format (for example, 16-bit signed integer linear PCM into 32-bit floating point linear PCM).

Audio Codec Services objects are standard Component Manager components.

The basic workflow for using an audio codec is as follows:

1. Find the appropriate codec object. You can use Component Manager functions to enumerate the codecs on a system; see *Component Manager Reference*. For sample code and an application that displays all the Component Manager components on a system, see *Fiendishthngs* at http://developer.apple.com/samplecode/Fiendishthngs/.
2. Open the codec object using the Component Manager.
3. Configure the codec for the desired input and output formats and other properties using the functions `AudioCodecGetPropertyInfo` (page 9), `AudioCodecGetProperty` (page 8), and `AudioCodecSetProperty` (page 13). Properties are listed in "Global Codec Properties" (page 17) and "Instance Codec Properties" (page 20).
4. Call the `AudioCodecInitialize` (page 10) function to put the codec in the initialized state. In this state, the format information for the translation cannot be changed. The codec has to be in the initialized state before you can send data to the codec or retrieve data from it.
5. Send data to the codec, then retrieve the encoded or decoded data from it, by looping the following functions:
    a. `AudioCodecAppendInputData` (page 7)

   b. `AudioCodecProduceOutputPackets` (page 11)

6. a. If you have another file to process with the same codec and want to change some property values, call the `AudioCodecUninitialize` (page 13) function and return to step 3.

   b. If you have another file to process with the same codec and do not want to change any property values, call the `AudioCodecReset` (page 12) function and return to step 5.

7. Close the codec object.

Mac OS X includes a wide range of audio codecs. Some possible audio codec object subtypes are listed in the format IDs enumeration in the CoreAudioTypes.h header file (see *Core Audio Data Types Reference*).

If you are interested in writing your own audio codec, see the audio codec software development kit (SDK) in `Developer/Examples/CoreAudio/AudioCodecs`.

Functions by Task

Property Management

`AudioCodecGetPropertyInfo` (page 9)
: Retrieves information about a codec property.

`AudioCodecGetProperty` (page 8)
: Retrieves the value of a codec property.

`AudioCodecSetProperty` (page 13)
: Sets the value of a codec property.

Data Handling

`AudioCodecInitialize` (page 10)
: Sets up the specified codec to perform a data format translation.

`AudioCodecUninitialize` (page 13)
: Moves the codec from the initialized state back to the uninitialized state.

`AudioCodecAppendInputData` (page 7)
: Appends audio data to the codec's input buffer.

`AudioCodecProduceOutputPackets` (page 11)
: Retrieves output data from a codec.

`AudioCodecReset` (page 12)
: Flushes all the audio data in the codec and clears the input buffer.

Functions

AudioCodecAppendInputData

Appends audio data to the codec's input buffer.

```
ComponentResult AudioCodecAppendInputData (
   AudioCodec inCodec,
   const void *inInputData,
   UInt32 *ioInputDataByteSize,
   UInt32 *ioNumberPackets,
   const AudioStreamPacketDescription *inPacketDescription
);
```

Parameters

inCodec

An audio codec object. Because an audio codec object is a Component Manger component instance, you can use the Component Manager (for example, the functions `FindNextComponent` and `OpenAComponent`) to obtain an audio codec object.

inInputData

The audio data to be sent to the codec. Indicate there is no more data to process by passing a buffer of `0`bytes.

ioInputDataByteSize

On input, the size in bytes of the data pointed to by the `inInputData` parameter. On output, the number of bytes the codec actually appended to its input buffer.

ioNumberPackets

On input, the number of elements in the `inPacketDescription` array. Pass `NULL` for this parameter if the input data has a constant number of frames per packet. On return, the number of packets actually processed by the codec.

inPacketDescription

For audio data that has a variable number of frames per packet, an array of `AudioStreamPacketDescription` structures that describes the packet layout. Pass `NULL` for this parameter if the input data has a constant number of frames per packet.

Return Value

Returns `NoErr` if successful. Returns `kAudioCodecStateError` if the codec has not been initialized. See "Result Codes" (page 37) for other possible values.

Discussion

A packet is the smallest, indivisible block of data for a given audio format. For linear PCM (pulse-code modulated) data, each packet contains exactly one frame, where a frame is a set of samples representing one sample for each channel. For compressed audio data formats, the number of frames in a packet depends on the encoding. For example, a packet of AAC represents 1024 frames of PCM. In some formats, the number of frames per packet varies. For such formats, you must include an array of `AudioStreamPacketDescription` structures that describes the packet layout.

Input data can be fed into an encoder and some decoders in blocks of any size (even byte by byte). However, if the encoded format of the input data fed to a decoder has a variable number of frames per packet, the data must be provided in multiples of whole packets. A codec's properties provide information about allowable

types of input and output, minimum and maximum buffer sizes, and so forth. Use the `AudioCodecGetProperty` function to read a codec's properties. The properties are described in "Global Codec Properties" (page 17) and "Instance Codec Properties" (page 20).

The combination of the `AudioCodecAppendInputData` and `AudioCodecProduceOutputPackets` functions implement a "push-pull" model of data handling. First, the input data is pushed into the codec, then the resulting output data is pulled out of that same codec.

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

See Also
`AudioCodecInitialize` (page 10)
`AudioCodecGetProperty` (page 8)
`AudioCodecProduceOutputPackets` (page 11)

Declared In
`AudioCodec.h`

AudioCodecGetProperty

Retrieves the value of a codec property.

```
ComponentResult AudioCodecGetProperty (
   AudioCodec inCodec,
   AudioCodecPropertyID inPropertyID,
   UInt32 *ioPropertyDataSize,
   void *outPropertyData
);
```

Parameters

inCodec
An audio codec object. Because an audio codec object is a Component Manger component instance, you can use the Component Manager (for example, the functions `FindNextComponent` and `OpenAComponent`) to obtain an audio codec object.

inPropertyID
Property ID of the property whose value you want to obtain. Codec property IDs are listed in "Global Codec Properties" (page 17) and "Instance Codec Properties" (page 20).

ioPropertyDataSize
On input, the size in bytes of the data buffer pointed to by the `outPropertyData` parameter. On output, the amount of data actually written to the buffer.

outPropertyData
The property data buffer.

Return Value
Returns `NoErr` if successful, otherwise, a result code. See "Result Codes" (page 37) for a list of possible values.

Discussion
All property values can be read regardless of the state of the codec. However, the values of some properties depend on whether the codec is initialized. Before calling this function, call the `AudioCodecGetPropertyInfo` function to determine the size of buffer you need for the property value.

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

See Also
AudioCodecGetPropertyInfo (page 9)
AudioCodecSetProperty (page 13)

Declared In
AudioCodec.h

AudioCodecGetPropertyInfo

Retrieves information about a codec property.

```
ComponentResult AudioCodecGetPropertyInfo (
   AudioCodec inCodec,
   AudioCodecPropertyID inPropertyID,
   UInt32 *outSize,
   Boolean *outWritable
);
```

Parameters

inCodec

An audio codec object. Because an audio codec object is a Component Manger component instance, you can use the Component Manager (for example, the functions FindNextComponent and OpenAComponent) to obtain an audio codec object.

inPropertyID

Property ID of the property about which you want to obtain information. Codec property IDs are listed in "Global Codec Properties" (page 17) and "Instance Codec Properties" (page 20).

outSize

On return, size in bytes of the current value of the property.

outWritable

Returns true if you can change the value of the property, otherwise false.

Return Value
Returns NoErr if successful, otherwise, a result code. See "Result Codes" (page 37) for a list of possible values.

Discussion
Call this function to:

- get the size of a property value before calling AudioCodecGetProperty to retrieve the value
- find out if a property value can be modified before calling AudioCodecSetProperty to set the value

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

See Also
AudioCodecGetProperty (page 8)
AudioCodecSetProperty (page 13)

Declared In
AudioCodec.h

AudioCodecInitialize

Sets up the specified codec to perform a data format translation.

```
ComponentResult AudioCodecInitialize (
   AudioCodec inCodec,
   const AudioStreamBasicDescription *inInputFormat,
   const AudioStreamBasicDescription *inOutputFormat,
   const void *inMagicCookie,
   UInt32 inMagicCookieByteSize
);
```

Parameters

inCodec

An audio codec object. Because an audio codec object is a Component Manager component instance, you can use the Component Manager (for example, the functions `FindNextComponent` and `OpenAComponent`) to obtain an audio codec object.

inInputFormat

A structure that describes the format of the input data. See *Core Audio Data Types Reference* for a description of this structure and the values of constants that can be used in this structure. If the input data has a variable number of frames per packet, this structure is supplemented with the `AudioStreamPacketDescription` structure passed in the `inPacketDescription` parameter of the `AudioCodecAppendInputData` function.

inOutputFormat

A structure that describes the format desired for the output data.

inMagicCookie

Magic cookie data, if required for the input format.

inMagicCookieByteSize

Size in bytes of the magic cookie data, if any.

Return Value

Returns `NoErr` if successful. Returns `kAudioCodecUnsupportedFormatError` if the codec cannot handle the specified data translation. See "Result Codes" (page 37) for other possible values.

Discussion

This function allocates any buffers needed, sets the input and output formats, and puts the codec into the initialized state. The codec has to be in the initialized state for the `AudioCodecAppendInputData` and `AudioCodecProduceOutputPackets` functions to work. While in this state, the format information for the translation cannot be changed; you must call the `AudioCodecUninitialize` function before making any changes.

A codec's properties provide information about allowable types of input and output, magic cookies, and so forth. Use the `AudioCodecGetProperty` function to read a codec's properties. The properties are described in "Global Codec Properties" (page 17) and "Instance Codec Properties" (page 20).

If any argument is `NULL`, any values previously set for that argument are used. For example, if you are using the same codec repeatedly with the same input and output formats, you only need to enter the formats the first time you initialize the codec. After that, you can uninitialize, change property values as necessary, and then call this function again with `NULL` in the `inInputFormat` and `inOutputFormat` parameters before processing the next set of data.

Availability

Available in Mac OS X v10.2 and later.

See Also
AudioCodecAppendInputData (page 7)
AudioCodecProduceOutputPackets (page 11)
AudioCodecGetProperty (page 8)
AudioCodecUninitialize (page 13)

Declared In
AudioCodec.h

AudioCodecProduceOutputPackets

Retrieves output data from a codec.

```
ComponentResult AudioCodecProduceOutputPackets (
   AudioCodec inCodec,
   void *outOutputData,
   UInt32 *ioOutputDataByteSize,
   UInt32 *ioNumberPackets,
   AudioStreamPacketDescription *outPacketDescription,
   UInt32 *outStatus
);
```

Parameters

inCodec

An audio codec object. Because an audio codec object is a Component Manger component instance, you can use the Component Manager (for example, the functions FindNextComponent and OpenAComponent) to obtain an audio codec object.

outOutputData

The output data buffer.

ioOutputDataByteSize

Indicates the size of the output data buffer.

ioNumberPackets

On input, the number of packets desired. On output, the number of packets actually placed in the output buffer.

outPacketDescription

An array of AudioStreamPacketDescription structures that describes the packet layout of the data returned by the outOutputData parameter. Pass NULL if you do not want this information returned. Note that this information is provided only when the output format is not linear PCM.

outStatus

On output, information about the codec's status to allow for proper data management. See "Output Status Constants" (page 30) for the possible values that can be returned.

Return Value
Returns NoErr if successful. Returns kAudioCodecStateError if the codec has not been initialized. Returns kAudioCodecNotEnoughBufferSpaceError if the output buffer is not large enough for the requested number of packets. See "Result Codes" (page 37) for other possible values.

Discussion
This function causes the codec to produce as many output packets as requested, provided there is sufficient input data. If there is not enough input data to produce the requested number of output packets, the outStatus parameter returns the value kAudioCodecProduceOutputPacketNeedsMoreInputData and

the `ioNumberPackets` parameter indicates the actual number of packets produced. On the other hand, if there is enough input data to produce at least one additional full packet, the `outStatus` parameter returns the value `kAudioCodecProduceOutputPacketSuccessHasMore`.

Note that decoders produce linear PCM data only in multiples of the number of frames in a packet of the encoded format. (See the `AudioCodecAppendInputData` function for definitions of *packet* and *frame* as used by this API.) You can use the `AudioCodecGetProperty` function to obtain this value from the `kAudioCodecPropertyPacketFrameSize` property. Similarly, this property indicates how many frames of linear PCM data an encoder needs in order to produce a packet of the specified output format.

Output data can be produced only in multiples of whole packets.

The combination of the `AudioCodecAppendInputData` and `AudioCodecProduceOutputPackets` functions implement a "push-pull" model of data handling. First, the input data is pushed into the codec, then the resulting output data is pulled out of that same codec.

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

See Also
`AudioCodecInitialize` (page 10)
`AudioCodecAppendInputData` (page 7)
`AudioCodecGetProperty` (page 8)

Declared In
`AudioCodec.h`

AudioCodecReset

Flushes all the audio data in the codec and clears the input buffer.

```
ComponentResult AudioCodecReset (
   AudioCodec inCodec
);
```

Parameters

inCodec

An audio codec object. Because an audio codec object is a Component Manger component instance, you can use the Component Manager (for example, the functions `FindNextComponent` and `OpenAComponent`) to obtain an audio codec object.

Return Value
Returns `NoErr` if successful, otherwise, a result code. See "Result Codes" (page 37) for a list of possible values.

Discussion
The input and output formats, magic cookie data, and other state variables are retained so that you needn't call the `AudioCodecInitialize` function again unless the values of some variables have changed.

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

See Also
`AudioCodecInitialize` (page 10)

Declared In
AudioCodec.h

AudioCodecSetProperty

Sets the value of a codec property.

```
ComponentResult AudioCodecSetProperty (
   AudioCodec inCodec,
   AudioCodecPropertyID inPropertyID,
   UInt32 inPropertyDataSize,
   const void *inPropertyData
);
```

Parameters

inCodec

An audio codec object. Because an audio codec object is a Component Manger component instance, you can use the Component Manager (for example, the functions `FindNextComponent` and `OpenAComponent`) to obtain an audio codec object.

inPropertyID

Property ID of the property whose value you want to set. Settable codec property IDs are listed in "Instance Codec Properties" (page 20).

inPropertyDataSize

Size in bytes of the property value data.

inPropertyData

Pointer to the data buffer containing the property value.

Return Value
Returns `NoErr` if successful, otherwise, a result code. See "Result Codes" (page 37) for a list of possible values.

Discussion
Codec properties are classified as either global properties, which remain the same for all instances of a codec, or instance properties, which may vary from instance to instance. However, not all instance property values can be modified. See "Instance Codec Properties" (page 20) for details. No property values can be modified when the codec is in the initialized state. You must call this function before you call the `AudioCodecInitialize` function, or after you call the `AudioCodecUninitialize` function. Call the `AudioCodecGetProperty` function to retrieve the current value of a property.

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

See Also
`AudioCodecInitialize` (page 10)
`AudioCodecUninitialize` (page 13)
`AudioCodecGetProperty` (page 8)

Declared In
AudioCodec.h

AudioCodecUninitialize

Moves the codec from the initialized state back to the uninitialized state.

```
ComponentResult AudioCodecUninitialize (
   AudioCodec inCodec
);
```

Parameters

inCodec

An audio codec object. Because an audio codec object is a Component Manger component instance, you can use the Component Manager (for example, the functions `FindNextComponent` and `OpenAComponent`) to obtain an audio codec object.

Return Value

Returns `NoErr` if successful, otherwise, a result code. See "Result Codes" (page 37) for a list of possible values.

Discussion

This function returns the codec to the uninitialized state. The codec may then be configured freely. This function does not flush the input buffer or clear input and output formats, magic cookie data, and other state variables. It is not necessary to call this function before closing the codec.

Availability

Available in Mac OS X v10.2 and later.

See Also

`AudioCodecInitialize` (page 10)

`AudioCodecSetProperty` (page 13)

`AudioCodecReset` (page 12)

Declared In

`AudioCodec.h`

Data Types

AudioCodec

An instance of a Component Manager component.

```
typedef ComponentInstance    AudioCodec;
```

Availability

Available in Mac OS X v10.2 and later.

Declared In

`AudioCodec.h`

AudioCodecPropertyID

An integer identifying an audio codec property.

```
typedef UInt32               AudioCodecPropertyID;
```

Availability

Available in Mac OS X v10.2 and later.

Declared In
AudioCodec.h

AudioCodecMagicCookieInfo

A structure holding magic cookie information needed by some codecs.

```
struct AudioCodecMagicCookieInfo
{
    UInt32            mMagicCookieSize;
    const void*        mMagicCookie;
};
typedef struct AudioCodecMagicCookieInfo    AudioCodecMagicCookieInfo;
```

Fields

`mMagicCookieSize`
The size of the magic cookie.

`mMagicCookie`
Generic constant pointer to the magic cookie.

Discussion
This structure is passed as input to the `AudioCodecGetProperty` (page 8) function for the `kAudioCodecPropertyFormatList` property. The first `4 + sizeof(void *)` bytes of the buffer pointed to by the function's `outPropertyData` parameter contains this structure on input.

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

Declared In
AudioCodec.h

AudioCodecPrimeInfo

A structure specifying the number of leading and trailing empty frames to be inserted.

```
typedef struct AudioCodecPrimeInfo
{
    UInt32        leadingFrames;
    UInt32        trailingFrames;
} AudioCodecPrimeInfo;
```

Fields

`leadingFrames`
An unsigned integer specifying the number of leading empty frames.

`trailingFrames`
An unsigned integer specifying the number of trailing empty frames.

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

Declared In
AudioCodec.h

MagicCookieInfo

A structure holding magic cookie information. (**Deprecated.** Renamed `AudioCodecMagicCookieInfo`,)

```
typedef struct AudioCodecMagicCookieInfo MagicCookieInfo;
```

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

Declared In
`AudioCodec.h`

Constants

Audio Codec Component Constants

Audio codec component types.

```
enum
{
    kAudioDecoderComponentType    = 'adec',
    kAudioEncoderComponentType    = 'aenc',
    kAudioUnityCodecComponentType = 'acdc'
};
```

Constants

`kAudioDecoderComponentType`

A codec that translates data in some other format into linear PCM.

The component subtype specifies the input format.

Available in Mac OS X v10.2 and later.

Declared in `AudioCodec.h`.

`kAudioEncoderComponentType`

A codec that translates linear PCM data into some other format

The component subtype specifies the output format.

Available in Mac OS X v10.2 and later.

Declared in `AudioCodec.h`.

`kAudioUnityCodecComponentType`

A codec that translates between different flavors of the same format.

The component subtype specifies the format.

Available in Mac OS X v10.2 and later.

Declared in `AudioCodec.h`.

Discussion
Some possible audio codec component subtypes are listed in the format IDs enumeration in the CoreAudioTypes.h header file (see *Core Audio Data Types Reference*).

Declared In
`AudioCodec.h`

Global Codec Properties

These read-only properties disclose the capabilities of the codec and remain the same for all instances of the codec.

```
enum
{
    kAudioCodecPropertyNameCFString                         = 'lnam',
    kAudioCodecPropertyManufacturerCFString                 = 'lmak',
    kAudioCodecPropertyFormatCFString                       = 'lfor',
    kAudioCodecPropertyHasVariablePacketByteSizes           = 'vpk?',
    kAudioCodecPropertySupportedInputFormats                = 'ifm#',
    kAudioCodecPropertySupportedOutputFormats               = 'ofm#',
    kAudioCodecPropertyAvailableInputSampleRates            = 'aisr',
    kAudioCodecPropertyAvailableOutputSampleRates           = 'aosr',
    kAudioCodecPropertyAvailableBitRateRange                = 'abrt',
    kAudioCodecPropertyMinimumNumberInputPackets            = 'mnip',
    kAudioCodecPropertyMinimumNumberOutputPackets           = 'mnop',
    kAudioCodecPropertyAvailableNumberChannels              = 'cmnc',
    kAudioCodecPropertyDoesSampleRateConversion             = 'lmrc',
    kAudioCodecPropertyAvailableInputChannelLayoutTags      = 'aicl',
    kAudioCodecPropertyAvailableOutputChannelLayoutTags     = 'aocl',
    kAudioCodecPropertyInputFormatsForOutputFormat          = 'if4o',
    kAudioCodecPropertyOutputFormatsForInputFormat          = 'of4i',
    kAudioCodecPropertyFormatInfo                           = 'acfi'
};
```

Constants

`kAudioCodecPropertyNameCFString`

A `CFStringRef` referencing the name of the codec component.

You must release the `CFStringRef` retrieved with this property when you are finished using the reference.

Available in Mac OS X v10.2 and later.

Declared in `AudioCodec.h`.

`kAudioCodecPropertyManufacturerCFString`

A `CFStringRef` referencing the manufacturer of the codec.

You must release the `CFStringRef` retrieved with this property when you are finished using the reference.

Available in Mac OS X v10.2 and later.

Declared in `AudioCodec.h`.

`kAudioCodecPropertyFormatCFString`

A `CFStringRef` referencing the name of the codec's format.

You must release the `CFStringRef` retrieved with this property when you are finished using the reference.

Available in Mac OS X v10.3 and later.

Declared in `AudioCodec.h`.

`kAudioCodecPropertySupportedInputFormats`

An array of `AudioStreamBasicDescription` structures describing the formats the codec supports for input data.

Available in Mac OS X v10.2 and later.

Declared in `AudioCodec.h`.

`kAudioCodecPropertySupportedOutputFormats`
An array of `AudioStreamBasicDescription` structures describing the formats the codec supports for output data.

Available in Mac OS X v10.2 and later.

Declared in `AudioCodec.h`.

`kAudioCodecPropertyAvailableInputSampleRates`
An array of `AudioValueRange` structures indicating the supported values for the input sample rate of the codec.

Note that the `mMinimum` and `mMaximum` fields of an `AudioValueRange` structure are equal when the structure represents a discrete value rather than a range. This property is required for encoders. (See also `kAudioCodecPropertyApplicableInputSampleRates` (page 24), `kAudioCodecPropertyCurrentInputSampleRate` (page 23).)

Available in Mac OS X v10.2 and later.

Declared in `AudioCodec.h`.

`kAudioCodecPropertyAvailableOutputSampleRates`
An array of `AudioValueRange` structures indicating the supported values for the output sample rate of the codec.

Note that the `mMinimum` and `mMaximum` fields of an `AudioValueRange` structure are equal when the structure represents a discrete value rather than a range. This property is required for encoders. (See also `kAudioCodecPropertyApplicableOutputSampleRates` (page 24), `kAudioCodecPropertyCurrentOutputSampleRate` (page 23).)

Available in Mac OS X v10.2 and later.

Declared in `AudioCodec.h`.

`kAudioCodecPropertyAvailableBitRateRange`
An array of `AudioValueRange` structures that indicate the target bit rates supported by the encoder.

Note that the `mMinimum` and `mMaximum` fields of an `AudioValueRange` structure are equal when the structure represents a discrete value rather than a range. This property can contain total bit rate or bit rate per channel as appropriate. This property is relevant to encoders only. (See also `kAudioCodecPropertyApplicableBitRateRange` (page 24), `kAudioCodecPropertyCurrentTargetBitRate` (page 23).)

Available in Mac OS X v10.3 and later.

Declared in `AudioCodec.h`.

`kAudioCodecPropertyMinimumNumberInputPackets`
The minimum number of input packets that need to be supplied to the codec before calling the `AudioCodecProduceOutputPackets` (page 11) function, specified as a `UInt32`.

The actual amount of data the codec processes with one invocation of the `AudioCodecProduceOutputPackets` (page 11) function can be less than this. For most codecs this value is `1`. You use the `AudioCodecAppendInputData` (page 7) function to put data into the codec's input buffer.

Available in Mac OS X v10.3 and later.

Declared in `AudioCodec.h`.

`kAudioCodecPropertyMinimumNumberOutputPackets`
: The minimum number of output packets that you need to be prepared to accept from the codec with one invocation of the `AudioCodecProduceOutputPackets` (page 11) function, specified as a `UInt32`.

  The actual amount of output the codec produces at one time might be less than this (see the `AudioCodecProduceOutputPackets` (page 11) function). For most codecs this value is `1`.

  Available in Mac OS X v10.3 and later.

  Declared in `AudioCodec.h`.

`kAudioCodecPropertyAvailableNumberChannels`
: An array of type `UInt32` that specifies the number of channels an encoder is capable of encoding or a decoder is capable of decoding to.

  The value `0xFFFFFFFF` indicates any number of channels.

  Available in Mac OS X v10.3 and later.

  Declared in `AudioCodec.h`.

`kAudioCodecPropertyDoesSampleRateConversion`
: An integer of type `UInt32` indicating whether the codec can do a sample rate conversion (if necessary) that preserves quality.

  This value is `1` if the codec can preserve quality when it does a sample rate conversion.

  Available in Mac OS X v10.5 and later.

  Declared in `AudioCodec.h`.

`kAudioCodecPropertyAvailableInputChannelLayoutTags`
: An array of type `AudioChannelLayoutTag` that specifies what channel layouts the codec is capable of using on input.

  Values of audio channel layout tags are listed in *Core Audio Data Types Reference*. See also `kAudioCodecPropertyCurrentInputChannelLayout` (page 25).

  Available in Mac OS X v10.5 and later.

  Declared in `AudioCodec.h`.

`kAudioCodecPropertyAvailableOutputChannelLayoutTags`
: An array of type `AudioChannelLayoutTag` that specifies what channel layouts the codec is capable of creating on output.

  Values of audio channel layout tags are listed in *Core Audio Data Types Reference*. See also `kAudioCodecPropertyCurrentOutputChannelLayout` (page 25).

  Available in Mac OS X v10.5 and later.

  Declared in `AudioCodec.h`.

`kAudioCodecPropertyInputFormatsForOutputFormat`
: An array of `AudioStreamBasicDescription` structures indicating what input formats the codec supports given a specific output format.

  To obtain this property value, when you call the `AudioCodecGetProperty` (page 8) function you specify the output format by including it as the first member of the array in the `outPropertyData` parameter. The function returns an array of supported input formats, overwriting the first member of the array in the process.

  This list is always a subset of the array in `kAudioCodecPropertySupportedInputFormats`.

  Available in Mac OS X v10.5 and later.

  Declared in `AudioCodec.h`.

`kAudioCodecPropertyOutputFormatsForInputFormat`

An array of `AudioStreamBasicDescription` structures indicating what output formats the codec supports given a specific input format.

To obtain this property value, when you call the `AudioCodecGetProperty` (page 8) function you specify the input format by including it as the first member of the array in the `outPropertyData` parameter. The function returns an array of supported output formats, overwriting the first member of the array in the process.

This list is always a subset of the array in `kAudioCodecPropertySupportedOutputFormats`.

Available in Mac OS X v10.5 and later.

Declared in `AudioCodec.h`.

`kAudioCodecPropertyFormatInfo`

On input, takes an `AudioFormatInfo` structure that contains a partially-filled `AudioStreamBasicDescription` structure. On output, returns an `AudioFormatInfo` structure that contains a fully initialized and validated `AudioStreamBasicDescription` structure.

The `AudioStreamBasicDescription` structure in the input `AudioFormatInfo` structure may include some fields that are set to `0` and may include a magic cookie. Audio Codec Services uses the information in the magic cookie (if included) and default values to fill in the `0` fields in the `AudioStreamBasicDescription`.

Available in Mac OS X v10.5 and later.

Declared in `AudioCodec.h`.

Discussion
These properties are used with the `AudioCodecGetProperty` (page 8) function.

The values of these properties are independent of the codec's internal state and cannot be changed. They can be read at any time the codec is open.

Declared In
`AudioCodec.h`

Instance Codec Properties

Properties that can be set or read on an instance of the audio codec.

```
enum
{
    kAudioCodecPropertyInputBufferSize              = 'tbuf',
    kAudioCodecPropertyPacketFrameSize              = 'pakf',
    kAudioCodecPropertyMaximumPacketByteSize        = 'pakb',
    kAudioCodecPropertyCurrentInputFormat           = 'ifmt',
    kAudioCodecPropertyCurrentOutputFormat          = 'ofmt',
    kAudioCodecPropertyMagicCookie                  = 'kuki',
    kAudioCodecPropertyUsedInputBufferSize          = 'ubuf',
    kAudioCodecPropertyIsInitialized                = 'init',
    kAudioCodecPropertyCurrentTargetBitRate         = 'brat',
    kAudioCodecPropertyCurrentInputSampleRate       = 'cisr',
    kAudioCodecPropertyCurrentOutputSampleRate      = 'cosr',
    kAudioCodecPropertyQualitySetting               = 'srcq',
    kAudioCodecPropertyApplicableBitRateRange       = 'brta',
    kAudioCodecPropertyApplicableInputSampleRates   = 'isra',
    kAudioCodecPropertyApplicableOutputSampleRates  = 'osra',
    kAudioCodecPropertyPaddedZeros                  = 'pad0',
    kAudioCodecPropertyPrimeMethod                  = 'prmm',
    kAudioCodecPropertyPrimeInfo                    = 'prim',
    kAudioCodecPropertyCurrentInputChannelLayout    = 'icl ',
    kAudioCodecPropertyCurrentOutputChannelLayout   = 'ocl ',
    kAudioCodecPropertySettings                     = 'acs ',
    kAudioCodecPropertyFormatList                   = 'acfl',
    kAudioCodecPropertyBitRateControlMode           = 'acbf',
    kAudioCodecPropertySoundQualityForVBR           = 'vbrq',
    kAudioCodecPropertyMinimumDelayMode             = 'mdel'
};
```

Constants

`kAudioCodecPropertyInputBufferSize`

The maximum input buffer size for the codec in bytes, specified as an integer of type `UInt32`.

This property can vary for some codecs depending on the bit stream format being handled. Not writable.

Available in Mac OS X v10.2 and later.

Declared in `AudioCodec.h`.

`kAudioCodecPropertyPacketFrameSize`

The number of frames of audio data encapsulated in each packet of data in the codec's format, specified as an integer of type `UInt32`.

For encoders, this is the output format. For decoders, this is the input format. Formats with variable frames per packet return a maximum value. Not writable.

Available in Mac OS X v10.2 and later.

Declared in `AudioCodec.h`.

`kAudioCodecPropertyHasVariablePacketByteSizes`
: An integer of type `UInt32` indicating whether all packets in the codec's format have the same byte size.

  A value of `0` indicates all packets have the same byte size (such codecs are sometimes referred to as constant bit rate (CBR) codecs). A `1` indicates that they vary in size (sometimes referred to as variable bit rate (VBR) codecs). The maximum size of a variable packet is up to the one indicated in `kAudioCodecPropertyMaximumPacketByteSize` (page 22). Any codec that reports `1` for this property must be able to handle packet descriptions, though it does not have to require them.

  Available in Mac OS X v10.2 and later.

  Declared in `AudioCodec.h`.

`kAudioCodecPropertyMaximumPacketByteSize`
: The maximum number of bytes of a packet of data in the codec's format, specified as an integer of type `UInt32`.

  If the format is constant bit rate, all packets are this size. If it is variable bit rate, the packets never exceed this size. This property always refers to the encoded data, so for encoders it refers to the output data and for decoders it refers to the input data. Not writable.

  Available in Mac OS X v10.2 and later.

  Declared in `AudioCodec.h`.

`kAudioCodecPropertyCurrentInputFormat`
: The format the codec expects for its input data, described by an `AudioStreamBasicDescription` structure.

  This property is almost always writable; however, if the codec supports only one input format, it does not have to be.

  Available in Mac OS X v10.2 and later.

  Declared in `AudioCodec.h`.

`kAudioCodecPropertyCurrentOutputFormat`
: The format in which the codec provides its output data, described by an `AudioStreamBasicDescription` structure.

  This property is almost always writable; however, if the codec supports only one output format, it does not have to be.

  Available in Mac OS X v10.2 and later.

  Declared in `AudioCodec.h`.

`kAudioCodecPropertyMagicCookie`
: Configuration data the codec requires in order to process the stream of data.

  Some codecs require configuration data—referred to as a *magic cookie*—that specifies how to process the audio data. This property value is a pointer to a buffer. The size, format, and contents of the buffer are defined by the codec. The magic cookie is opaque and codec-specific binary data that embodies all parameters for a given configuration. To determine whether a codec requires a magic cookie, call the `AudioCodecGetPropertyInfo` (page 9) function with this property. If the function returns a size greater than `0`, then the codec takes a magic cookie. Writable if present.

  Available in Mac OS X v10.2 and later.

  Declared in `AudioCodec.h`.

`kAudioCodecPropertyUsedInputBufferSize`
: The number of bytes of data in the codec's input buffer, specified as an integer of type `UInt32`.

  The amount of buffer space available is the value returned by `kAudioCodecPropertyInputBufferSize` (page 21) minus the value returned by this property. Not writable.

  Available in Mac OS X v10.2 and later.

  Declared in `AudioCodec.h`.

`kAudioCodecPropertyIsInitialized`
: An integer of type `UInt32` indicating whether the codec is in the initialized state.

  A value of `0` indicates that the codec is uninitialized and any other value indicates the codec is initialized. Not writable. Use the `AudioCodecInitialize` (page 10) function to set the codec's state to initialized and the `AudioCodecUninitialize` (page 13) function to set it to uninitialized.

  Available in Mac OS X v10.2 and later.

  Declared in `AudioCodec.h`.

`kAudioCodecPropertyCurrentTargetBitRate`
: The target number of bits per second when encoding data, specified as an integer of type `UInt32`.

  This property is usually only relevant to encoders, but some decoders can retrieve the bit rate from the magic cookie or from the bit stream. if a decoder can know what bit rate it's set to, it may report it. See also `kAudioCodecPropertyApplicableBitRateRange` (page 24),`kAudioCodecPropertyAvailableBitRateRange` (page 18). Writable for encoders if supported.

  Available in Mac OS X v10.2 and later.

  Declared in `AudioCodec.h`.

`kAudioCodecPropertyCurrentInputSampleRate`
: The current input sample rate in Hertz, specified as a floating point number of type `Float64`.

  May be writable, but if only one sample rate is supported, it does not have to be.

  Available in Mac OS X v10.2 and later.

  Declared in `AudioCodec.h`.

`kAudioCodecPropertyCurrentOutputSampleRate`
: The current output sample rate in Hertz, specified as a floating point number of type `Float64`.

  May be writable, but if only one sample rate is supported, it does not have to be.

  Available in Mac OS X v10.2 and later.

  Declared in `AudioCodec.h`.

`kAudioCodecPropertyQualitySetting`
: A sound quality setting for the codec, specified as an integer of type `UInt32`.

  This property sets the tradeoff between sound quality and CPU time consumption. The property value is between `0` and `0x7F`, inclusive, where `0` is minimum sound quality and fastest processing. Some constants that can be used with this property are defined in "Audio Codec Quality Constants" (page 27) for convenience. Writable if supported.

  Available in Mac OS X v10.2 and later.

  Declared in `AudioCodec.h`.

`kAudioCodecPropertyApplicableBitRateRange`

An array of `AudioValueRange` structures indicating the target bit rates supported by the encoder in its current configuration.

Note that the `mMinimum` and `mMaximum` fields of an `AudioValueRange` structure are equal when the structure represents a discrete value rather than a range. This property is relevant only to encoders. See also `kAudioCodecPropertyCurrentTargetBitRate` (page 23), `kAudioCodecPropertyAvailableBitRateRange` (page 18). Not writable.

Available in Mac OS X v10.3 and later.

Declared in `AudioCodec.h`.

`kAudioCodecPropertyApplicableInputSampleRates`

An array of `AudioValueRange` structures indicating the supported values for the input sample rate of the codec for the current bit rate.

Note that the `mMinimum` and `mMaximum` fields of an `AudioValueRange` structure are equal when the structure represents a discrete value rather than a range. This property is relevant only to encoders. See also `kAudioCodecPropertyCurrentInputSampleRate` (page 23) and `kAudioCodecPropertyAvailableInputSampleRates` (page 18). Not writable.

Available in Mac OS X v10.3 and later.

Declared in `AudioCodec.h`.

`kAudioCodecPropertyApplicableOutputSampleRates`

An array of `AudioValueRange` structures indicating the supported values for the output sample rate of the codec for the current bit rate.

Note that the `mMinimum` and `mMaximum` fields of an `AudioValueRange` structure are equal when the structure represents a discrete value rather than a range. This property is relevant only to encoders. See also `kAudioCodecPropertyCurrentOutputSampleRate` (page 23) `kAudioCodecPropertyAvailableOutputSampleRates` (page 18). Not writable.

Available in Mac OS X v10.3 and later.

Declared in `AudioCodec.h`.

`kAudioCodecPropertyPaddedZeros`

The number of zero-valued samples that were appended to the last packet of input data in order to fill out the last packet of encoded data. Specified as an integer of type `UInt32`.

This property is relevant only to encoders and has no default value. Not writable.

Available in Mac OS X v10.5 and later.

Declared in `AudioCodec.h`.

`kAudioCodecPropertyPrimeMethod`

The method used to add priming and trailing frames, specified as an integer of type `UInt32`.

See "Audio Codec Priming Method Constants" (page 27) for possible values. Writable for encoders that offer options for padding out the last packet.

Available in Mac OS X v10.3 and later.

Declared in `AudioCodec.h`.

`kAudioCodecPropertyPrimeInfo`

The numbers of leading and trailing priming frames, specified through a pointer to an `AudioCodecPrimeInfo` structure.

Not writable on encoders. On decoders this may be writable, instructing the decoder to trim the first and last packets to the number of priming frames specified.

Available in Mac OS X v10.3 and later.

Declared in `AudioCodec.h`.

`kAudioCodecPropertyCurrentInputChannelLayout`

The channel layout that the codec is using for input specified as an `AudioChannelLayout` structure.

May be writable. If only one channel layout is supported, it does not have to be. See also `kAudioCodecPropertyAvailableInputChannelLayoutTags` (page 19).

Available in Mac OS X v10.5 and later.

Declared in `AudioCodec.h`.

`kAudioCodecPropertyCurrentOutputChannelLayout`

The channel layout that the codec is using for output specified as an `AudioChannelLayout` structure.

If settable on a encoder, it means the encoder can re-map channels. May be writable. If only one channel layout is supported or the codec does no channel remapping (that is, the output channel layout is always the same as the input channel layout), it does not have to be. See also `kAudioCodecPropertyAvailableOutputChannelLayoutTags` (page 19).

Available in Mac OS X v10.5 and later.

Declared in `AudioCodec.h`.

`kAudioCodecPropertySettings`

A list of all an encoder's settable properties and their values, specified through a `CFDictionaryRef` object.

This property is used by encoders only. The main purpose of this property is to get a snapshot of the current encoder state. Key values are specific to a codec. The keys and values used with codecs provided by Apple, Inc. are reserved for internal use by Apple.

Before setting this property, first use the `AudioCodecGetProperty` (page 8) function to retrieve the current dictionary. Change only one property at a time, as changing one property may affect the values of other properties. Writable if supported.

Available in Mac OS X v10.3 and later.

Declared in `AudioCodec.h`.

`kAudioCodecPropertyBitRateControlMode`

The bit rate control mode to be used by an encoder that can produce variable sized packets, specified as an integer of type `UInt32`.

Possible values for the property are listed in "Bit Rate Control Mode Constants" (page 28). Although an encoder's packet size may be variable (such codecs are sometimes referred to as VBR encoders), it might be capable of maintaining a constant bit rate over a transmission channel when encoding in real time with a fixed end-to-end audio delay. For example, MP3 and MPEG-AAC encoders use a bit reservoir mechanism to meet that constraint. Only needs to be writable if the codec supports multiple bit rate control strategies.

Available in Mac OS X v10.5 and later.

Declared in `AudioCodec.h`.

`kAudioCodecPropertyFormatList`

An array of `AudioFormatListItem` structures that lists all the formats that can be handled by the codec.

There are cases where a bit stream can be made of several layers that may have different sample rates, a different number of channels, and so forth. This property lists all these layers. For decoders, when you call the `AudioCodecGetProperty` (page 8) function you must pass in a magic cookie that provides a description of what is in the bit stream.

There is no default value for this property. On input, the `outPropertyData` parameter passed to `AudioCodecGetProperty` should begin with an `AudioCodecMagicCookieInfo` (page 15) structure. On output, this structure is overwritten by the `AudioFormatListItem` structure returned from the property. For encoders, this property returns a list of formats that will be in the bitstream the encoder is about to produce. No input data is required for encoders.

Important: This encoder property is applicable only to audio formats that are made of two or more layers, where the base layers can be decoded by systems that aren't capable of handling the enhancement layers. For example, using the information provided by this property, any AAC decoder can decode the base layer of a High Efficiency AAC bitstream that contains an AAC Low Complexity base layer.

Available in Mac OS X v10.5 and later.

Declared in `AudioCodec.h`.

`kAudioCodecPropertySoundQualityForVBR`

A target sound quality level for an encoder configured to use a variable bit rate control mode, specified as an integer of type `UInt32`.

The property value can be from `0` to `0x7F`, where `0` is the lowest sound quality. See the discussion of the `kAudioCodecBitRateControlMode_Variable` constant in "Bit Rate Control Mode Constants" (page 28). Writable if supported.

Available in Mac OS X v10.5 and later.

Declared in `AudioCodec.h`.

`kAudioCodecPropertyMinimumDelayMode`

Specifies whether the encoder is set in its lowest possible delay mode, specified as an integer of type `UInt32`.

Set this property to 1 to set the encoder to its minimum delay mode. An encoder may prepend zero-valued samples to the input signal in order to cause delays (such as those from a filter) to coincide on a block boundary. This operation, however, results in an increased encoding delay, which may be undesired. You can use this property to turn off this feature to avoid the additional delay. Writable if supported.

Available in Mac OS X v10.5 and later.

Declared in `AudioCodec.h`.

Discussion

These properties are used with the `AudioCodecGetProperty` (page 8) and `AudioCodecSetProperty` (page 13) functions.

These properties are dependent on the codec's current state. A property may be read/write or read only, depending on the data format of the codec.

These properties may have different values depending on whether the codec is in the initialized state. You can set writable properties only when the codec is not initialized. All properties can be read at any time the codec is open.

Declared In
AudioCodec.h

Audio Codec Quality Constants

Sound quality settings to be used with the property kAudioCodecPropertyQualitySetting.

```
enum
{
    kAudioCodecQuality_Max       = 0x7F,
    kAudioCodecQuality_High      = 0x60,
    kAudioCodecQuality_Medium    = 0x40,
    kAudioCodecQuality_Low       = 0x20,
    kAudioCodecQuality_Min       = 0
};
```

Constants

kAudioCodecQuality_Max
Spend as much CPU time as necessary to obtain the highest-quality sound output possible.
Available in Mac OS X v10.2 and later.
Declared in AudioCodec.h.

kAudioCodecQuality_High
Spend sufficient CPU time to achieve high quality sound output.
Available in Mac OS X v10.2 and later.
Declared in AudioCodec.h.

kAudioCodecQuality_Medium
Give CPU time and sound output quality equal consideration.
Available in Mac OS X v10.2 and later.
Declared in AudioCodec.h.

kAudioCodecQuality_Low
Give speed of processing priority over sound quality.
Available in Mac OS X v10.2 and later.
Declared in AudioCodec.h.

kAudioCodecQuality_Min
Do the processing as quickly as possible without concern for sound quality.
Available in Mac OS X v10.2 and later.
Declared in AudioCodec.h.

Declared In
AudioCodec.h

Audio Codec Priming Method Constants

Values used with kAudioCodecPropertyPrimeMethod.

```
enum
{
    kAudioCodecPrimeMethod_Pre      = 0,
    kAudioCodecPrimeMethod_Normal   = 1,
    kAudioCodecPrimeMethod_None     = 2
};
```

Constants

`kAudioCodecPrimeMethod_Pre`

Adds both leading and trailing input frames.

Available in Mac OS X v10.3 and later.

Declared in `AudioCodec.h`.

`kAudioCodecPrimeMethod_Normal`

Adds only trailing input frames (zero latency).

Available in Mac OS X v10.3 and later.

Declared in `AudioCodec.h`.

`kAudioCodecPrimeMethod_None`

Adds no priming or trailing frames.

Available in Mac OS X v10.3 and later.

Declared in `AudioCodec.h`.

Declared In

`AudioCodec.h`

Bit Rate Control Mode Constants

Bit rate control modes to be used with `kAudioCodecPropertyBitRateControlMode`.

```
enum
{
    kAudioCodecBitRateControlMode_Constant                  = 0,
    kAudioCodecBitRateControlMode_LongTermAverage           = 1,
    kAudioCodecBitRateControlMode_VariableConstrained       = 2,
    kAudioCodecBitRateControlMode_Variable                  = 3,
};
```

Constants

`kAudioCodecBitRateControlMode_Constant`

The encoder maintains a constant bit rate suitable for use over a transmission channel when encoding in real time with a fixed end-to-end audio delay.

Although a constant bit rate is maintained in this mode, the number of bits allocated to encode each fixed length of audio data may be variable (that is, packet sizes are variable). For example, MP3 and MPEG-AAC use a bit reservoir mechanism to meet that constraint.

Available in Mac OS X v10.5 and later.

Declared in `AudioCodec.h`.

`kAudioCodecBitRateControlMode_LongTermAverage`

The provided target bit rate is achieved over a long term average (typically after the first 1000 packets).

This mode is similar to `kAudioCodecBitRateControlMode_Constant` in the sense that the target bit rate is maintained in a long term average. However, it does not provide constant delay when using constant bit rate transmission. At encoding bit rates below the maximum bit rate, this mode offers a better sound quality for a given bit rate than `kAudioCodecBitRateControlMode_Constant` can; that is, a qualitatively better encoding is performed.

Available in Mac OS X v10.5 and later.

Declared in `AudioCodec.h`.

`kAudioCodecBitRateControlMode_VariableConstrained`

The encoder dynamically allocates the bit resources according to the characteristics of the underlying signal; however, some constraints are applied in order to limit the variation of the bit rate.

Available in Mac OS X v10.5 and later.

Declared in `AudioCodec.h`.

`kAudioCodecBitRateControlMode_Variable`

Similar to the VBR constrained mode, however the packet size is virtually unconstrained.

The coding process targets constant sound quality. This mode usually provides the best tradeoff between quality and bit rate.

Available in Mac OS X v10.5 and later.

Declared in `AudioCodec.h`.

Discussion
These modes are applicable only to encoders that can produce variable packet sizes, such as AAC.

Declared In
`AudioCodec.h`

Constants for kAudioCodecPropertySettings

Property values reserved by Apple for use with the `kAudioCodecPropertySettings` (page 25) property.

```
#define kAudioSettings_TopLevelKey        "name"
#define kAudioSettings_Version            "version"
#define kAudioSettings_Parameters         "parameters"
#define kAudioSettings_SettingKey         "key"
#define kAudioSettings_SettingName        "name"
#define kAudioSettings_ValueType          "value type"
#define kAudioSettings_AvailableValues    "available values"
#define kAudioSettings_LimitedValues      "limited values"
#define kAudioSettings_CurrentValue       "current value"
#define kAudioSettings_Summary            "summary"
#define kAudioSettings_Hint               "hint"
#define kAudioSettings_Unit               "unit"
```

Declared In
`AudioCodec.h`

Audio Settings Flags

Flags reserved for use by Apple.

```
enum {
    kAudioSettingsFlags_ExpertParameter          = (1L << 0),
    kAudioSettingsFlags_InvisibleParameter       = (1L << 1),
    kAudioSettingsFlags_MetaParameter            = (1L << 2),
    kAudioSettingsFlags_UserInterfaceParameter   = (1L << 3)
};
```

Declared In
`AudioCodec.h`

Output Status Constants

Status values returned from the `AudioCodecProduceOutputPackets` (page 11) function.

```
enum
{
    kAudioCodecProduceOutputPacketFailure                   = 1,
    kAudioCodecProduceOutputPacketSuccess                   = 2,
    kAudioCodecProduceOutputPacketSuccessHasMore            = 3,
    kAudioCodecProduceOutputPacketNeedsMoreInputData        = 4,
    kAudioCodecProduceOutputPacketAtEOF                     = 5
};
```

Constants

`kAudioCodecProduceOutputPacketFailure`
Couldn't complete the request due to an error.

It's possible that some output data was produced; if so, the number of packets produced is returned in the `ioNumberPackets` parameter.

Available in Mac OS X v10.2 and later.

Declared in `AudioCodec.h`.

`kAudioCodecProduceOutputPacketSuccess`
The number of requested output packets was produced without errors and there isn't any more input data to process.

Available in Mac OS X v10.2 and later.

Declared in `AudioCodec.h`.

`kAudioCodecProduceOutputPacketSuccessHasMore`
The number of requested output packets was produced without errors and there is enough input data remaining to produce at least one more packet of output data.

Available in Mac OS X v10.2 and later.

Declared in `AudioCodec.h`.

`kAudioCodecProduceOutputPacketNeedsMoreInputData`
There was insufficient input data to produce the requested number of output packets.

The `ioNumberPackets` parameter returns the number of output packets produced.

Available in Mac OS X v10.2 and later.

Declared in `AudioCodec.h`.

`kAudioCodecProduceOutputPacketAtEOF`
The end-of-file marker was hit during the processing.

Fewer than the requested number of output packets may have been produced. Check the value returned in `ioNumberPackets` for the actual number produced. Note that not all formats have EOF markers in them.

Available in Mac OS X v10.2 and later.

Declared in `AudioCodec.h`.

Declared In
`AudioCodec.h`

Audio Codec Routine Selectors

Selectors used by the Component Manager to call routines implemented by the codec and exposed to developers through the Audio Codec Services API. These selectors are for use by codec developers; if you are calling Audio Codec Services functions, you don't need to use these constants.

```
enum
{
    kAudioCodecGetPropertyInfoSelect                    = 0x0001,
    kAudioCodecGetPropertySelect                        = 0x0002,
    kAudioCodecSetPropertySelect                        = 0x0003,
    kAudioCodecInitializeSelect                         = 0x0004,
    kAudioCodecUninitializeSelect                       = 0x0005,
    kAudioCodecAppendInputDataSelect                    = 0x0006,
    kAudioCodecProduceOutputDataSelect                  = 0x0007,
    kAudioCodecResetSelect                              = 0x0008
};
```

Constants
`kAudioCodecGetPropertyInfoSelect`
Corresponds to the `AudioCodecGetPropertyInfo` (page 9) function.

Available in Mac OS X v10.2 and later.

Declared in `AudioCodec.h`.

`kAudioCodecGetPropertySelect`
Corresponds to the `AudioCodecGetProperty` (page 8) function.

Available in Mac OS X v10.2 and later.

Declared in `AudioCodec.h`.

`kAudioCodecSetPropertySelect`
Corresponds to the `AudioCodecSetProperty` (page 13) function.

Available in Mac OS X v10.2 and later.

Declared in `AudioCodec.h`.

`kAudioCodecInitializeSelect`
Corresponds to the `AudioCodecInitialize` (page 10) function.

Available in Mac OS X v10.2 and later.

Declared in `AudioCodec.h`.

`kAudioCodecUninitializeSelect`
Corresponds to the `AudioCodecUninitialize` (page 13) function.

Available in Mac OS X v10.2 and later.

Declared in `AudioCodec.h`.

`kAudioCodecAppendInputDataSelect`
Corresponds to the `AudioCodecAppendInputData` (page 7) function.

Available in Mac OS X v10.2 and later.

Declared in `AudioCodec.h`.

`kAudioCodecProduceOutputDataSelect`
Corresponds to the `AudioCodecProduceOutputPackets` (page 11) function.

Available in Mac OS X v10.2 and later.

Declared in `AudioCodec.h`.

`kAudioCodecResetSelect`
Corresponds to the `AudioCodecReset` (page 12) function.

Available in Mac OS X v10.2 and later.

Declared in `AudioCodec.h`.

Declared In
`AudioCodec.h`

Deprecated Audio Codec Properties

(**Deprecated.** Use the properties in "Global Codec Properties" (page 17) and "Instance Codec Properties" (page 20) instead.)

```
enum
{
    kAudioCodecPropertyRequiresPacketDescription           = 'pakd',
    kAudioCodecPropertyAvailableBitRates                   = 'brt#',
    kAudioCodecExtendFrequencies                           = 'acef',
    kAudioCodecUseRecommendedSampleRate                     = 'ursr',
    kAudioCodecOutputPrecedence                             = 'oppr',
    kAudioCodecBitRateFormat                                =
kAudioCodecPropertyBitRateControlMode,
    kAudioCodecDoesSampleRateConversion                     =
kAudioCodecPropertyDoesSampleRateConversion,
    kAudioCodecInputFormatsForOutputFormat                  =
kAudioCodecPropertyInputFormatsForOutputFormat,
    kAudioCodecOutputFormatsForInputFormat                  =
kAudioCodecPropertyOutputFormatsForInputFormat,
    kAudioCodecPropertyInputChannelLayout                   =
kAudioCodecPropertyCurrentInputChannelLayout,
    kAudioCodecPropertyOutputChannelLayout                  =
kAudioCodecPropertyCurrentOutputChannelLayout,
    kAudioCodecPropertyAvailableInputChannelLayouts         =
kAudioCodecPropertyAvailableInputChannelLayoutTags,
    kAudioCodecPropertyAvailableOutputChannelLayouts        =
kAudioCodecPropertyAvailableOutputChannelLayoutTags,
    kAudioCodecPropertyZeroFramesPadded                     =
kAudioCodecPropertyPaddedZeros
};
```

Constants

`kAudioCodecPropertyRequiresPacketDescription`

A non-zero value indicates that the format the codec implements requires that an `AudioStreamPacketDescription` array must be supplied with any data in that format. (**Deprecated.** Redundant with `kAudioCodecPropertyHasVariablePacketByteSizes`.)

Available in Mac OS X v10.2 and later.

Declared in `AudioCodec.h`.

`kAudioCodecPropertyAvailableBitRates`

indicates the target bit rates supported by the encoder. (**Deprecated.** Use `kAudioCodecPropertyAvailableBitRateRange` instead.)

Available in Mac OS X v10.2 and later.

Declared in `AudioCodec.h`.

`kAudioCodecExtendFrequencies`

indicates whether an encoder should extend its cutoff frequency if such an option exists. (**Deprecated.** Never used.)

Available in Mac OS X v10.3 and later.

Declared in `AudioCodec.h`.

`kAudioCodecUseRecommendedSampleRate`

For encoders that do sample rate conversion, indicates whether the encoder is using the recommended sample rate for the given input. (**Deprecated.** Redundant, as a sample rate of `0.0` is interpreted to mean "let the codec decide.")

Available in Mac OS X v10.3 and later.

Declared in `AudioCodec.h`.

`kAudioCodecOutputPrecedence`

For encoders that do sample rate conversion, indicates whether the bit rate, sample rate, or neither has precedence over the other. (**Deprecated.** Redundant because precedence is implicitly set by either providing a non-zero bit rate or sample rate and setting the other to zero (which allows the encoder to choose any applicable rate).)

Available in Mac OS X v10.3 and later.

Declared in `AudioCodec.h`.

`kAudioCodecBitRateFormat`

Indicates which bit rate control mode is applied to encoders that can produce variable packet sizes. (**Deprecated.** Renamed `kAudioCodecPropertyBitRateControlMode` (page 25).)

Available in Mac OS X v10.3 and later.

Declared in `AudioCodec.h`.

`kAudioCodecDoesSampleRateConversion`

indicating whether the codec wants to do a sample rate conversion. (**Deprecated.** Renamed `kAudioCodecPropertyDoesSampleRateConversion` (page 19).)

Available in Mac OS X v10.3 and later.

Declared in `AudioCodec.h`.

`kAudioCodecInputFormatsForOutputFormat`

An array indicating what the codec supports for input data given an output format. (**Deprecated.** Renamed `kAudioCodecPropertyInputFormatsForOutputFormat` (page 19))

Available in Mac OS X v10.3 and later.

Declared in `AudioCodec.h`.

`kAudioCodecOutputFormatsForInputFormat`

An array of `AudioStreamBasicDescription` structures indicating what the codec supports for output data given an input format. (**Deprecated.** Renamed `kAudioCodecPropertyOutputFormatsForInputFormat` (page 20).)

Available in Mac OS X v10.3 and later.

Declared in `AudioCodec.h`.

`kAudioCodecPropertyInputChannelLayout`

An `AudioChannelLayout` structure that specifies the channel layout that the codec is using for input. (**Deprecated.** Renamed `kAudioCodecPropertyCurrentInputChannelLayout` (page 25).)

Available in Mac OS X v10.3 and later.

Declared in `AudioCodec.h`.

`kAudioCodecPropertyOutputChannelLayout`

An `AudioChannelLayout` structure that specifies the channel layout that the codec is using for output. (**Deprecated.** Renamed `kAudioCodecPropertyCurrentOutputChannelLayout` (page 25).)

Available in Mac OS X v10.3 and later.

Declared in `AudioCodec.h`.

`kAudioCodecPropertyAvailableInputChannelLayouts`

An array that specifies what channel layouts the codec is capable of using on input. (**Deprecated.** Renamed `kAudioCodecPropertyAvailableInputChannelLayoutTags` (page 19).)

Available in Mac OS X v10.3 and later.

Declared in `AudioCodec.h`.

`kAudioCodecPropertyAvailableOutputChannelLayouts`

An array that specifies what channel layouts the codec is capable of using on output. (**Deprecated.** Renamed `kAudioCodecPropertyAvailableOutputChannelLayoutTags` (page 19).)

Available in Mac OS X v10.3 and later.

Declared in `AudioCodec.h`.

`kAudioCodecPropertyZeroFramesPadded`

A `UInt32` indicating the number of zeros (samples) that were appended to the last packet of input data to make a complete packet encoding. (**Deprecated.** Renamed `kAudioCodecPropertyPaddedZeros` (page 24).)

Available in Mac OS X v10.3 and later.

Declared in `AudioCodec.h`.

Declared In
`AudioCodec.h`

Deprecated Constants Used With kAudioCodecBitRateFormat

(**Deprecated.** Use "Bit Rate Control Mode Constants" (page 28) instead.)

```
enum
{
    kAudioCodecBitRateFormat_CBR     =    kAudioCodecBitRateControlMode_Constant,
    kAudioCodecBitRateFormat_ABR     =
kAudioCodecBitRateControlMode_LongTermAverage,
    kAudioCodecBitRateFormat_VBR     =
kAudioCodecBitRateControlMode_VariableConstrained
};
```

Constants
`kAudioCodecBitRateFormat_CBR`

The encoder maintains a constant bit rate. (**Deprecated.** Replaced with `kAudioCodecBitRateControlMode_Constant` (page 28).)

Available in Mac OS X v10.3 and later.

Declared in `AudioCodec.h`.

`kAudioCodecBitRateFormat_ABR`

The provided target bit rate is achieved over a long term average. (**Deprecated.** Replaced with `kAudioCodecBitRateControlMode_LongTermAverage` (page 29).)

Available in Mac OS X v10.3 and later.

Declared in `AudioCodec.h`.

`kAudioCodecBitRateFormat_VBR`

Encoder dynamically allocates the bit resources according to the characteristics of the underlying signal. (**Deprecated.** Replaced with `kAudioCodecBitRateControlMode_VariableConstrained` (page 29).)

Available in Mac OS X v10.3 and later.

Declared in `AudioCodec.h`.

Declared In
`AudioCodec.h`

Deprecated Constants Used With kAudioCodecOutputPrecedence

(**Deprecated.** There is no replacement, because precedence is implicitly set by either providing a non-zero bit rate or sample rate and setting the other to zero (which allows the encoder to choose any applicable rate).)

```
enum
{
    kAudioCodecOutputPrecedenceNone           = 0,
    kAudioCodecOutputPrecedenceBitRate        = 1,
    kAudioCodecOutputPrecedenceSampleRate     = 2
};
```

Constants
`kAudioCodecOutputPrecedenceNone`

Changes in the bit rate or the sample rate are constrained by the other value. (**Deprecated.** There is no replacement.)

Available in Mac OS X v10.3 and later.

Declared in `AudioCodec.h`.

`kAudioCodecOutputPrecedenceBitRate`

The bit rate may be changed freely, adjusting the sample rate if necessary. (**Deprecated.** There is no replacement.)

Available in Mac OS X v10.3 and later.

Declared in `AudioCodec.h`.

`kAudioCodecOutputPrecedenceSampleRate`

The sample rate may be changed freely, adjusting the bit rate if necessary. (**Deprecated.** There is no replacement.)

Available in Mac OS X v10.3 and later.

Declared in `AudioCodec.h`.

Declared In
`AudioCodec.h`

Deprecated Constants Used With kAudioSettings_Hint

(**Deprecated.** Replaced with the constants in "Audio Settings Flags" (page 29).)

```
enum
{
    kHintBasic       = 0,
    kHintAdvanced    = 1,
    kHintHidden      = 2
};
```

Constants

`kHintBasic`

(**Deprecated.** No direct replacement; see "Audio Settings Flags" (page 29),)

Available in Mac OS X v10.3 and later.

Declared in `AudioCodec.h`.

`kHintAdvanced`

(**Deprecated.** No direct replacement; see "Audio Settings Flags" (page 29),)

Available in Mac OS X v10.3 and later.

Declared in `AudioCodec.h`.

`kHintHidden`

(**Deprecated.** No direct replacement; see "Audio Settings Flags" (page 29),)

Available in Mac OS X v10.3 and later.

Declared in `AudioCodec.h`.

Declared In
`AudioCodec.h`

Result Codes

Result Code	Value	Description
`kAudioCodecNoError`	0	No error. Available in Mac OS X v10.2 and later.
`kAudioCodecUnspecifiedError`	'what'	Cause of error is unknown. Available in Mac OS X v10.2 and later.
`kAudioCodecUnknownPropertyError`	'who?'	The specified property identifier is unknown. Available in Mac OS X v10.2 and later.

Result Code	Value	Description
`kAudioCodecBadPropertySizeError`	'!siz'	The buffer provided for the property is too small. Available in Mac OS X v10.2 and later.
`kAudioCodecIllegalOperationError`	'nope'	This operation can't be done; for example, an attempt to set a property that is not writable. Available in Mac OS X v10.2 and later.
`kAudioCodecUnsupportedFormatError`	'!dat'	This codec does not support the specified format. Available in Mac OS X v10.2 and later.
`kAudioCodecStateError`	'!stt'	The codec is in the wrong state (initialized or uninitialized) to perform this operation. Available in Mac OS X v10.2 and later.
`kAudioCodecNotEnoughBufferSpaceError`	'!buf'	The output buffer is too small for the amount of data processed. Available in Mac OS X v10.2 and later.

Document Revision History

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

Date	Notes
2007-10-31	New document that describes the programming interface used to configure audio codecs and to encode and decode audio data.

Document Revision History

Index

M

O