
Audio Queue Services Reference

Audio & Video: Audio



2009-08-07



Apple Inc.
© 2009 Apple Inc.
All rights reserved.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, mechanical, electronic, photocopying, recording, or otherwise, without prior written permission of Apple Inc., with the following exceptions: Any person is hereby authorized to store documentation on a single computer for personal use only and to print copies of documentation for personal use provided that the documentation contains Apple's copyright notice.

The Apple logo is a trademark of Apple Inc.

Use of the "keyboard" Apple logo (Option-Shift-K) for commercial purposes without the prior written consent of Apple may constitute trademark infringement and unfair competition in violation of federal and state laws.

No licenses, express or implied, are granted with respect to any of the technology described in this document. Apple retains all intellectual property rights associated with the technology described in this document. This document is intended to assist application developers to develop applications only for Apple-labeled computers.

Every effort has been made to ensure that the information in this document is accurate. Apple is not responsible for typographical errors.

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

Apple, the Apple logo, Mac, and Mac OS are trademarks of Apple Inc., registered in the United States and other countries.

iPhone is a trademark of Apple Inc.

Times is a registered trademark of Heidelberger Druckmaschinen AG, available from Linotype Library GmbH.

Simultaneously published in the United States and Canada.

Even though Apple has reviewed this document, APPLE MAKES NO WARRANTY OR REPRESENTATION, EITHER EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT, ITS QUALITY, ACCURACY, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. AS A RESULT, THIS DOCUMENT IS PROVIDED "AS IS," AND YOU, THE READER, ARE

ASSUMING THE ENTIRE RISK AS TO ITS QUALITY AND ACCURACY.

IN NO EVENT WILL APPLE BE LIABLE FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES RESULTING FROM ANY DEFECT OR INACCURACY IN THIS DOCUMENT, even if advised of the possibility of such damages.

THE WARRANTY AND REMEDIES SET FORTH ABOVE ARE EXCLUSIVE AND IN LIEU OF ALL OTHERS, ORAL OR WRITTEN, EXPRESS OR IMPLIED. No Apple dealer, agent, or employee is authorized to make any modification, extension, or addition to this warranty.

Some states do not allow the exclusion or limitation of implied warranties or liability for incidental or consequential damages, so the above limitation or exclusion may not apply to you. This warranty gives you specific legal rights, and you may also have other rights which vary from state to state.

Contents

Audio Queue Services Reference 5

Overview	5
Functions by Task	5
Controlling Audio Queues	5
Creating and Disposing of Audio Queues	6
Handling Audio Queue Buffers	6
Manipulating Audio Queue Parameters	6
Manipulating Audio Queue Properties	6
Handling Timing	7
Performing Offline Rendering	7
Functions	7
AudioQueueAddPropertyListener	7
AudioQueueAllocateBuffer	8
AudioQueueAllocateBufferWithPacketDescriptions	9
AudioQueueCreateTimeline	10
AudioQueueDeviceGetCurrentTime	11
AudioQueueDeviceGetNearestStartTime	11
AudioQueueDeviceTranslateTime	12
AudioQueueDispose	13
AudioQueueDisposeTimeline	13
AudioQueueEnqueueBuffer	14
AudioQueueEnqueueBufferWithParameters	15
AudioQueueFlush	17
AudioQueueFreeBuffer	17
AudioQueueGetCurrentTime	18
AudioQueueGetParameter	19
AudioQueueGetProperty	20
AudioQueueGetPropertySize	20
AudioQueueNewInput	21
AudioQueueNewOutput	22
AudioQueueOfflineRender	23
AudioQueuePause	24
AudioQueuePrime	24
AudioQueueRemovePropertyListener	25
AudioQueueReset	26
AudioQueueSetOfflineRenderFormat	27
AudioQueueSetParameter	27
AudioQueueSetProperty	28
AudioQueueStart	29
AudioQueueStop	30
Callbacks by Task	31

- Handling Audio Queue Buffers for Recording and Playback 31
- Defining a Property Listener 31
- Callbacks 31
 - AudioQueueInputCallback 31
 - AudioQueueOutputCallback 32
 - AudioQueuePropertyListenerProc 33
- Data Types 34
 - AudioQueueBuffer 34
 - AudioQueueBufferRef 35
 - AudioQueueRef 35
 - AudioQueueTimelineRef 36
 - AudioQueueLevelMeterState 36
 - AudioQueueParameterEvent 36
 - AudioQueueParameterID 37
 - AudioQueueParameterValue 37
- Constants 37
 - Audio Queue Property Identifiers 37
 - Audio Queue Parameter Identifiers 40
- Result Codes 40

Document Revision History 43

Index 45

Audio Queue Services Reference

Framework:	AudioToolbox/AudioToolbox.h
Declared in	AudioQueue.h

Overview

This document describes Audio Queue Services, a C programming interface in the Audio Toolbox framework, which is part of Core Audio.

An audio queue is a software object you use for recording or playing audio. An audio queue does the work of:

- Connecting to audio hardware
- Managing memory
- Employing codecs, as needed, for compressed audio formats
- Mediating playback or recording

Audio Queue Services enables you to record and play audio in linear PCM, in compressed formats (such as Apple Lossless and AAC), and in other formats for which users have installed codecs. Audio Queue Services also supports scheduled playback and synchronization of multiple audio queues and synchronization of audio with video.

Note: Audio Queue Services provides features similar to those previously offered by the Sound Manager and in Mac OS X. It adds additional features such as synchronization. The Sound Manager is deprecated in Mac OS X v10.5 and does not work with 64-bit applications. Audio Queue Services is recommended for all new development and as a replacement for the Sound Manager in existing Mac OS X applications.

Functions by Task

Controlling Audio Queues

[AudioQueueStart](#) (page 29)

Begins playing or recording audio.

[AudioQueuePrime](#) (page 24)

Decodes enqueued buffers in preparation for playback.

- [AudioQueueFlush](#) (page 17)
Resets an audio queue's decoder state.
- [AudioQueueStop](#) (page 30)
Stops playing or recording audio.
- [AudioQueuePause](#) (page 24)
Pauses audio playback or recording.
- [AudioQueueReset](#) (page 26)
Resets an audio queue.

Creating and Disposing of Audio Queues

- [AudioQueueNewOutput](#) (page 22)
Creates a new playback audio queue object.
- [AudioQueueNewInput](#) (page 21)
Creates a new recording audio queue object.
- [AudioQueueDispose](#) (page 13)
Disposes of an audio queue.

Handling Audio Queue Buffers

- [AudioQueueAllocateBuffer](#) (page 8)
Asks an audio queue object to allocate an audio queue buffer.
- [AudioQueueAllocateBufferWithPacketDescriptions](#) (page 9)
Asks an audio queue object to allocate an audio queue buffer with space for packet descriptions.
- [AudioQueueFreeBuffer](#) (page 17)
Asks an audio queue to dispose of an audio queue buffer.
- [AudioQueueEnqueueBuffer](#) (page 14)
Adds a buffer to the buffer queue of a recording or playback audio queue.
- [AudioQueueEnqueueBufferWithParameters](#) (page 15)
Adds a buffer to the buffer queue of a playback audio queue object, specifying start time and other settings.

Manipulating Audio Queue Parameters

- [AudioQueueGetParameter](#) (page 19)
Gets an audio queue parameter value.
- [AudioQueueSetParameter](#) (page 27)
Sets a playback audio queue parameter value.

Manipulating Audio Queue Properties

- [AudioQueueGetProperty](#) (page 20)
Gets an audio queue property value.

[AudioQueueSetProperty](#) (page 28)

Sets an audio queue property value.

[AudioQueueGetPropertySize](#) (page 20)

Gets the size of the value of an audio queue property.

[AudioQueueAddPropertyListener](#) (page 7)

Adds a property listener callback to an audio queue.

[AudioQueueRemovePropertyListener](#) (page 25)

Removes a property listener callback from an audio queue.

Handling Timing

[AudioQueueCreateTimeline](#) (page 10)

Creates a timeline object for an audio queue.

[AudioQueueDisposeTimeline](#) (page 13)

Disposes of an audio queue's timeline object.

[AudioQueueDeviceGetCurrentTime](#) (page 11)

Gets the current time of the audio hardware device associated with an audio queue.

[AudioQueueDeviceGetNearestStartTime](#) (page 11)

Gets the start time, for an audio hardware device, that is closest to a requested start time.

[AudioQueueDeviceTranslateTime](#) (page 12)

Converts the time for an audio queue's associated audio hardware device from one time base representation to another.

[AudioQueueGetCurrentTime](#) (page 18)

Gets the current audio queue time.

Performing Offline Rendering

[AudioQueueSetOfflineRenderFormat](#) (page 27)

Sets the rendering mode and audio format for a playback audio queue.

[AudioQueueOfflineRender](#) (page 23)

Exports audio to a buffer, instead of to a device, using a playback audio queue.

Functions

AudioQueueAddPropertyListener

Adds a property listener callback to an audio queue.

```
OSStatus AudioQueueAddPropertyListener (
    AudioQueueRef inAQ,
    AudioQueuePropertyID inID,
    AudioQueuePropertyListenerProc inProc,
    void *inUserData
);
```

Parameters*inAQ*

The audio queue that you want to assign a property listener callback to.

inID

The ID of the property whose changes you want to respond to. See [“Audio Queue Property Identifiers”](#) (page 37).

inProc

The callback to be invoked when the property value changes.

inUserData

Custom data for the property listener callback.

Return Value

A result code. See [“Audio Queue Result Codes”](#) (page 40).

Discussion

Use this function to let your application respond to property value changes in an audio queue. For example, say your application’s user interface has a button that acts as a Play/Stop toggle switch. When an audio file has finished playing, the audio queue stops and the value of the `kAudioQueueProperty_IsRunning` property changes from `true` to `false`. You can use a property listener callback to update the button text appropriately.

Availability

Available in Mac OS X v10.5 and later.

See Also

[AudioQueueRemovePropertyListener](#) (page 25)

Related Sample Code

AudioFileStreamExample

AudioQueueTools

Declared In

AudioQueue.h

AudioQueueAllocateBuffer

Asks an audio queue object to allocate an audio queue buffer.

```
OSStatus AudioQueueAllocateBuffer (
    AudioQueueRef inAQ,
    UInt32 inBufferSize,
    AudioQueueBufferRef *outBuffer
);
```

Parameters*inAQ*

The audio queue you want to allocate a buffer.

inBufferSize

The desired capacity of the new buffer, in bytes. Appropriate capacity depends on the processing you will perform on the data as well as on the audio data format.

outBuffer

On output, points to the newly allocated audio queue buffer.

Return Value

A result code. See [“Audio Queue Result Codes”](#) (page 40).

Discussion

Once allocated, the pointer to the audio queue buffer and the buffer’s capacity cannot be changed. The buffer’s size field, `mAudioDataByteSize`, which indicates the amount of valid data, is initially set to 0.

Availability

Available in Mac OS X v10.5 and later.

See Also

[AudioQueueAllocateBufferWithPacketDescriptions](#) (page 9)

[AudioQueueFreeBuffer](#) (page 17)

Related Sample Code

AudioFileStreamExample

AudioQueueTools

Declared In

AudioQueue.h

AudioQueueAllocateBufferWithPacketDescriptions

Asks an audio queue object to allocate an audio queue buffer with space for packet descriptions.

```
AudioQueueAllocateBufferWithPacketDescriptions(
    AudioQueueRef          inAQ,
    UInt32                 inBufferSize,
    UInt32                 inNumberPacketDescriptions,
    AudioQueueBufferRef    *outBuffer
);
```

Parameters*inAQ*

The audio queue you want to allocate a buffer.

inBufferSize

The desired data capacity of the new buffer, in bytes. Appropriate capacity depends on the processing you will perform on the data as well as on the audio data format.

inNumberPacketDescriptions

The desired size of the packet description array in the new audio queue buffer.

outBuffer

On output, points to the newly allocated audio queue buffer.

Return Value

A result code. See [“Audio Queue Result Codes”](#) (page 40).

Discussion

Use this function when allocating an audio queue buffer for use with a VBR compressed data format.

Once allocated, the pointer to the audio queue buffer and the buffer’s capacity cannot be changed. The buffer’s size field, `mAudioDataByteSize`, which indicates the amount of valid data, is initially set to 0.

Availability

Available in Mac OS X v10.6 and later.

See Also

[AudioQueueAllocateBuffer](#) (page 8)

[AudioQueueFreeBuffer](#) (page 17)

Declared In

AudioQueue.h

AudioQueueCreateTimeline

Creates a timeline object for an audio queue.

```
OSStatus AudioQueueCreateTimeline (
    AudioQueueRef inAQ,
    AudioQueueTimelineRef *outTimeline
);
```

Parameters

inAQ

The audio queue to associate with the new timeline object.

outTimeline

On output, the newly created timeline object.

Return Value

A result code. See [“Audio Queue Result Codes”](#) (page 40).

Discussion

Create a timeline object if you want to get timeline discontinuity information from an audio queue using the [AudioQueueGetCurrentTime](#) (page 18) function.

Availability

Available in Mac OS X v10.5 and later.

See Also

[AudioQueueDisposeTimeline](#) (page 13)

[AudioQueueGetCurrentTime](#) (page 18)

Declared In

AudioQueue.h

AudioQueueDeviceGetCurrentTime

Gets the current time of the audio hardware device associated with an audio queue.

```
OSStatus AudioQueueDeviceGetCurrentTime (
    AudioQueueRef inAQ,
    AudioTimeStamp *outTimeStamp
);
```

Parameters*inAQ*

The audio queue whose associated audio device is to be queried.

outDeviceTime

On output, the current time of the audio hardware device associated with the audio queue. If the device is not running, the only valid field in the audio timestamp structure is `mHostTime`.

Return Value

A result code. See [“Audio Queue Result Codes”](#) (page 40).

Discussion

This function returns a value whether or not the audio hardware device associated with the audio queue is running. The similar `AudioDeviceGetCurrentTime` function, declared in the `AudioHardware.h` header file, returns an error in this case.

Availability

Available in Mac OS X v10.5 and later.

See Also

[AudioQueueGetCurrentTime](#) (page 18)

`AudioDeviceGetCurrentTime`

Declared In

AudioQueue.h

AudioQueueDeviceGetNearestStartTime

Gets the start time, for an audio hardware device, that is closest to a requested start time.

```
OSStatus AudioQueueDeviceGetNearestStartTime (
    AudioQueueRef inAQ,
    AudioTimeStamp *ioRequestedStartTime,
    UInt32 inFlags
);
```

Parameters*inAQ*

The audio queue whose associated audio hardware device’s start time you want to get.

ioRequestedDeviceTime

On input, the requested start time. On output, the actual start time.

inFlags

Reserved for future use. Pass 0.

Return Value

A result code. See [“Audio Queue Result Codes”](#) (page 40).

Discussion

This function asks an audio queue’s associated device for a start time to use for recording or playback. The time returned will be equal to or later than the requested start time, depending on device and system factors. For example, the start time might be shifted to allow for aligning buffer access. The device must be running to use this function.

Availability

Available in Mac OS X v10.5 and later.

See Also

[AudioQueueDeviceGetCurrentTime](#) (page 11)

Declared In

AudioQueue.h

AudioQueueDeviceTranslateTime

Converts the time for an audio queue’s associated audio hardware device from one time base representation to another.

```
OSStatus AudioQueueDeviceTranslateTime (
    AudioQueueRef inAQ,
    const AudioTimeStamp *inTime,
    AudioTimeStamp *outTime
);
```

Parameters

inAQ

The audio queue associated with the device whose times are being translated.

inDeviceTime

The time to be translated.

outDeviceTime

On output, the translated time.

Return Value

A result code. See [“Audio Queue Result Codes”](#) (page 40).

Discussion

The device must be running for this function to provide a result. For an explanation of the various time base representations for an audio hardware device, see `AudioTimeStamp` in *Core Audio Data Types Reference*.

Availability

Available in Mac OS X v10.5 and later.

Declared In

AudioQueue.h

AudioQueueDispose

Disposes of an audio queue.

```
OSStatus AudioQueueDispose (
    AudioQueueRef inAQ,
    Boolean inImmediate
);
```

Parameters

inAQ

The audio queue you want to dispose of.

inImmediate

If you pass `true`, the audio queue is disposed of immediately (that is, synchronously). If you pass `false`, disposal does not take place until all enqueued buffers are processed (that is, asynchronously).

Return Value

A result code. See [“Audio Queue Result Codes”](#) (page 40).

Discussion

Disposing of an audio queue also disposes of its resources, including its buffers. After you call this function, you can no longer interact with the audio queue. In addition, the audio queue no longer invokes any callbacks.

Availability

Available in Mac OS X v10.5 and later.

See Also

[AudioQueueFlush](#) (page 17)

Related Sample Code

[AudioFileStreamExample](#)

[AudioQueueTools](#)

Declared In

`AudioQueue.h`

AudioQueueDisposeTimeline

Disposes of an audio queue’s timeline object.

```
OSStatus AudioQueueDisposeTimeline (
    AudioQueueRef inAQ,
    AudioQueueTimelineRef inTimeline
);
```

Parameters

inAQ

The audio queue associated with the timeline object you want to dispose of.

inTimeline

The timeline object to dispose of.

Return Value

A result code. See [“Audio Queue Result Codes”](#) (page 40).

Discussion

Disposing of an audio queue automatically disposes of any associated resources, including a timeline object. Call this function only if you want to dispose of a timeline object and not the audio queue associated with it.

Availability

Available in Mac OS X v10.5 and later.

See Also

[AudioQueueCreateTimeline](#) (page 10)

[AudioQueueDispose](#) (page 13)

Declared In

AudioQueue.h

AudioQueueEnqueueBuffer

Adds a buffer to the buffer queue of a recording or playback audio queue.

```
OSStatus AudioQueueEnqueueBuffer (
    AudioQueueRef          inAQ,
    AudioQueueBufferRef    inBuffer,
    UInt32                 inNumPacketDescs,
    const AudioStreamPacketDescription *inPacketDescs
);
```

Parameters

inAQ

The audio queue that owns the audio queue buffer.

inBuffer

The audio queue buffer to add to the buffer queue.

inNumPacketDescs

The number of packets of audio data in the *inBuffer* parameter. Use a value of 0 for any of the following situations:

- When playing a constant bit rate (CBR) format.
- When the audio queue is a recording (input) audio queue.
- When the buffer you are reenqueueing was allocated with the [AudioQueueAllocateBufferWithPacketDescriptions](#) (page 9) function. In this case, your callback should describe the buffer's packets in the buffer's `mPacketDescriptions` and `mPacketDescriptionCount` fields.

inPacketDescs

An array of packet descriptions. Use a value of `NULL` for any of the following situations:

- When playing a constant bit rate (CBR) format.
- When the audio queue is an input (recording) audio queue.
- When the buffer you are reenqueueing was allocated with the [AudioQueueAllocateBufferWithPacketDescriptions](#) (page 9) function. In this case, your callback should describe the buffer's packets in the buffer's `mPacketDescriptions` and `mPacketDescriptionCount` fields.

Return Value

A result code. See [“Audio Queue Result Codes”](#) (page 40).

Discussion

Audio queue callbacks use this function to reenqueue buffers—placing them “last in line” in a buffer queue. A playback (or *output*) callback reenqueuees a buffer after the buffer is filled with fresh audio data (typically from a file). A recording (or *input*) callback reenqueuees a buffer after the buffer’s contents were written (typically to a file).

Availability

Available in Mac OS X v10.5 and later.

See Also

[AudioQueueEnqueueBufferWithParameters](#) (page 15)

Related Sample Code

AudioFileStreamExample

AudioQueueTools

Declared In

AudioQueue.h

AudioQueueEnqueueBufferWithParameters

Adds a buffer to the buffer queue of a playback audio queue object, specifying start time and other settings.

```
OSStatus AudioQueueEnqueueBufferWithParameters (
    AudioQueueRef          inAQ,
    AudioQueueBufferRef    inBuffer,
    UInt32                 inNumPacketDescs,
    const AudioStreamPacketDescription *inPacketDescs,
    UInt32                 inTrimFramesAtStart,
    UInt32                 inTrimFramesAtEnd,
    UInt32                 inNumParamValues,
    const AudioQueueParameterEvent *inParamValues,
    const AudioTimeStamp   inStartTime,
    AudioTimeStamp         *outActualStartTime
);
```

Parameters

inAQ

The audio queue object that owns the audio queue buffer.

inBuffer

The audio queue buffer to add to the buffer queue. Before calling this function, the buffer must contain the audio data to be played.

inNumPacketDescs

The number of packets of audio data in the *inBuffer* parameter. Use a value of 0 for either of the following situations:

- When playing a constant bit rate (CBR) format.
- When the buffer you are reenqueuing was allocated with the [AudioQueueAllocateBufferWithPacketDescriptions](#) (page 9) function. In this case, your callback should describe the buffer's packets in the buffer's `mPacketDescriptions` and `mPacketDescriptionCount` fields.

inPacketDescs

An array of packet descriptions. Use a value of NULL for either of the following situations:

- When playing a constant bit rate (CBR) format.
- When the buffer you are reenqueuing was allocated with the [AudioQueueAllocateBufferWithPacketDescriptions](#) (page 9) function. In this case, your callback should describe the buffer's packets in the buffer's `mPacketDescriptions` and `mPacketDescriptionCount` fields.

inTrimFramesAtStart

The number of priming frames to skip at the start of the buffer.

inTrimFramesAtEnd

The number of frames to skip at the end of the buffer.

inNumParamValues

The number of audio queue parameter values pointed to by the *inParamValues* parameter. If you are not setting parameters, use 0.

inParamValues

An array of parameters to apply to an audio queue buffer. (In Mac OS X v10.5, there is only one audio queue parameter, `kAudioQueueParam_Volume`.) If you are not setting parameters for the buffer, use NULL.

Assign parameter values before playback—they cannot be changed while a buffer is playing. Changes to audio queue buffer parameters take effect when the buffer starts playing.

inStartTime

The desired start time for playing the buffer. To specify a time relative to when the audio queue started, use the `mSampleTime` field of the `AudioTimeStamp` structure. Use NULL to indicate that the buffer should play as soon as possible—which may be after previously queued buffers finish playing.

Buffers play in the order they are enqueued (first in, first out). If multiple buffers are queued, the start times must be in ascending order or NULL; otherwise, an error occurs. This parameter specifies when audio data is to start playing, ignoring any trim frames specified in the *inTrimFramesAtStart* parameter.

outActualStartTime

On output, the time when the buffer will actually start playing.

Return Value

A result code. See [“Audio Queue Result Codes”](#) (page 40).

Discussion

You can exert some control over the buffer queue with this function. You can assign audio queue settings that are, in effect, carried by an audio queue buffer as you enqueue it. Hence, settings take effect when an audio queue buffer begins playing.

This function applies only to playback. Recording audio queues do not take parameters and do not support variable bit rate (VBR) formats (which might require trimming).

Availability

Available in Mac OS X v10.5 and later.

See Also

[AudioQueueEnqueueBuffer](#) (page 14)

Declared In

AudioQueue.h

AudioQueueFlush

Resets an audio queue's decoder state.

```
OSStatus AudioQueueFlush (
    AudioQueueRef inAQ
);
```

Parameters

inAQ

The audio queue to flush.

Return Value

A result code. See [“Audio Queue Result Codes”](#) (page 40).

Discussion

Call `AudioQueueFlush` after enqueueing the last audio queue buffer to ensure that all buffered data, as well as all audio data in the midst of processing, gets recorded or played. If you do not call this function, stale data in the audio queue's decoder may interfere with playback or recording of the next set of buffers.

Call this function before calling [AudioQueueStop](#) (page 30) if you want to ensure that all enqueued data reaches the destination. If you call `AudioQueueStop` with the *inImmediate* parameter set to `false`, calling this function does nothing; under those conditions, `AudioQueueStop` calls this function.

Availability

Available in Mac OS X v10.5 and later.

See Also

[AudioQueueDispose](#) (page 13)

[AudioQueueStop](#) (page 30)

Related Sample Code

AudioFileStreamExample

AudioQueueTools

Declared In

AudioQueue.h

AudioQueueFreeBuffer

Asks an audio queue to dispose of an audio queue buffer.

```
OSStatus AudioQueueFreeBuffer (
    AudioQueueRef inAQ,
    AudioQueueBufferRef inBuffer
);
```

Parameters*inAQ*

The audio queue that owns the audio queue buffer you want to dispose of.

inBuffer

The buffer to dispose of.

Return Value

A result code. See [“Audio Queue Result Codes”](#) (page 40).

Discussion

Disposing of an audio queue also disposes of its buffers. Call this function only if you want to dispose of a particular buffer while continuing to use an audio queue. You can dispose of a buffer only when the audio queue that owns it is stopped (that is, not processing audio data).

Availability

Available in Mac OS X v10.5 and later.

See Also

[AudioQueueAllocateBuffer](#) (page 8)

[AudioQueueAllocateBufferWithPacketDescriptions](#) (page 9)

[AudioQueueDispose](#) (page 13)

Declared In

AudioQueue.h

AudioQueueGetCurrentTime

Gets the current audio queue time.

```
OSStatus AudioQueueGetCurrentTime (
    AudioQueueRef inAQ,
    AudioQueueTimelineRef inTimeline,
    AudioTimeStamp *outTimeStamp,
    Boolean *outTimelineDiscontinuity
);
```

Parameters*inAQ*

The audio queue whose current time you want to get.

inTimeline

The audio queue timeline object to which timeline discontinuities are reported. Use NULL if the audio queue does not have an associated timeline object.

outTime

On output, the current audio queue time. The `mSampleTime` field represents audio queue time in terms of the audio queue sample rate, relative to when the queue started or will start.

outTimelineDiscontinuity

On output, `true` if there has been a timeline discontinuity, or `false` if there has been no discontinuity. If the audio queue does not have an associated timeline object, this parameter is always `NULL`.

A timeline discontinuity may occur, for example, if the sample rate is changed for the audio hardware device associated with an audio queue.

Return Value

A result code. See “[Audio Queue Result Codes](#)” (page 40).

Availability

Available in Mac OS X v10.5 and later.

See Also

[AudioQueueCreateTimeline](#) (page 10)

[AudioQueueDeviceGetCurrentTime](#) (page 11)

Declared In

`AudioQueue.h`

AudioQueueGetParameter

Gets an audio queue parameter value.

```
OSStatus AudioQueueGetParameter (
    AudioQueueRef inAQ,
    AudioQueueParameterID inParamID,
    AudioQueueParameterValue *outValue
);
```

Parameters

inAQ

The audio queue that you want to get a parameter value from.

inParamID

The ID of the parameter whose value you want to get. In Mac OS X v10.5, audio queues have one parameter available: `kAudioQueueParam_Volume`, which controls playback gain. See “[Audio Queue Parameter Identifiers](#)” (page 40)

outValue

On output, points to the current value of the specified parameter.

Return Value

A result code. See “[Audio Queue Result Codes](#)” (page 40).

Discussion

You can access the current parameter values for an audio queue at any time with this function. An audio queue parameter value is the sum of settings applied at buffer granularity, using the [AudioQueueEnqueueBufferWithParameters](#) (page 15) function, and settings applied to the audio queue per se, using the [AudioQueueSetParameter](#) (page 27) function.

Availability

Available in Mac OS X v10.5 and later.

See Also

[AudioQueueSetParameter](#) (page 27)

Declared In

AudioQueue.h

AudioQueueGetProperty

Gets an audio queue property value.

```
OSStatus AudioQueueGetProperty (
    AudioQueueRef inAQ,
    AudioQueuePropertyID inID,
    void *outData,
    UInt32 *ioDataSize
);
```

Parameters*inAQ*

The audio queue that you want to get a property value from.

*inID*The ID of the property whose value you want to get. See [“Audio Queue Property Identifiers”](#) (page 37).*outData*

On output, the desired property value.

ioDataSize

On input, the maximum bytes of space the caller expects to receive. On output, the actual data size of the property value.

Return ValueA result code. See [“Audio Queue Result Codes”](#) (page 40).**Discussion**

Before calling this function, you can use the [AudioQueueGetPropertySize](#) (page 20) function to determine the size, in bytes, of the value of a specified property. Some properties have values of a specific size, as described in [“Audio Queue Property Identifiers”](#) (page 37).

Availability

Available in Mac OS X v10.5 and later.

See Also[AudioQueueSetProperty](#) (page 28)[AudioQueueGetPropertySize](#) (page 20)**Related Sample Code**

AudioFileStreamExample

AudioQueueTools

Declared In

AudioQueue.h

AudioQueueGetPropertySize

Gets the size of the value of an audio queue property.

```
OSStatus AudioQueueGetPropertySize (
    AudioQueueRef inAQ,
    AudioQueuePropertyID inID,
    UInt32 *outDataSize
);
```

Parameters*inAQ*

The audio queue that has the property value whose size you want to get.

inID

The ID of the property value whose size you want to get. See [“Audio Queue Property Identifiers”](#) (page 37).

outDataSize

On output, the size of the requested property value.

Return Value

A result code. See [“Audio Queue Result Codes”](#) (page 40).

Availability

Available in Mac OS X v10.5 and later.

See Also

[AudioQueueGetProperty](#) (page 20)

Related Sample Code

AudioQueueTools

Declared In

AudioQueue.h

AudioQueueNewInput

Creates a new recording audio queue object.

```
OSStatus AudioQueueNewInput (
    const AudioStreamBasicDescription *inFormat,
    AudioQueueInputCallback           inCallbackProc,
    void                               *inUserData,
    CFRunLoopRef                      inCallbackRunLoop,
    CFStringRef                       inCallbackRunLoopMode,
    UInt32                             inFlags,
    AudioQueueRef                     *outAQ
);
```

Parameters*inFormat*

The compressed or uncompressed audio data format to record to. When recording to linear PCM, only interleaved formats are supported.

inCallbackProc

A callback function to use with the recording audio queue. The audio queue calls this function when the audio queue has finished filling a buffer. See [AudioQueueInputCallback](#) (page 31).

inUserData

A custom data structure for use with the callback function.

inCallbackRunLoop

The event loop on which the callback function pointed to by the *inCallbackProc* parameter is to be called. If you specify `NULL`, the callback is called on one of the audio queue's internal threads.

inCallbackRunLoopMode

The run loop mode in which to invoke the callback function specified in the *inCallbackProc* parameter. Typically, you pass `kCFRunLoopCommonModes` or use `NULL`, which is equivalent. You can choose to create your own thread with your own run loops. For more information on run loops, see *Run Loops* and *CFRunLoop Reference*.

inFlags

Reserved for future use. Must be 0.

outAQ

On output, the newly created recording audio queue.

Return Value

A result code. See [“Audio Queue Result Codes”](#) (page 40).

Availability

Available in Mac OS X v10.5 and later.

See Also

[AudioQueueSetOfflineRenderFormat](#) (page 27)

Related Sample Code

[AudioQueueTools](#)

Declared In

`AudioQueue.h`

AudioQueueNewOutput

Creates a new playback audio queue object.

```
OSStatus AudioQueueNewOutput (
    const AudioStreamBasicDescription *inFormat,
    AudioQueueOutputCallback         inCallbackProc,
    void *inUserData,
    CFRunLoopRef                     inCallbackRunLoop,
    CFStringRef                       inCallbackRunLoopMode,
    UInt32                            inFlags,
    AudioQueueRef                     *outAQ
);
```

Parameters*inFormat*

The data format of the audio to play. For linear PCM, only interleaved formats are supported. Compressed formats are also supported.

inCallbackProc

A callback function to use with the playback audio queue. The audio queue calls this function when the audio queue has finished playing a buffer. See [AudioQueueOutputCallback](#) (page 32).

inUserData

A custom data structure for use with the callback function.

inCallbackRunLoop

The event loop on which the callback function pointed to by the `inCallbackProc` parameter is to be called. If you specify `NULL`, the callback is invoked on one of the audio queue's internal threads.

inCallbackRunLoopMode

The run loop mode in which to invoke the callback function specified in the `inCallbackProc` parameter. Typically, you pass `kCFRunLoopCommonModes` or use `NULL`, which is equivalent. You can choose to create your own thread with your own run loops. For more information on run loops, see *Run Loops* and *CFRunLoop Reference*.

inFlags

Reserved for future use. Must be 0.

outAQ

On output, the newly created playback audio queue object.

Return Value

A result code. See [“Audio Queue Result Codes”](#) (page 40).

Availability

Available in Mac OS X v10.5 and later.

See Also

[AudioQueueOfflineRender](#) (page 23)

Related Sample Code

[AudioFileStreamExample](#)

[AudioQueueTools](#)

Declared In

`AudioQueue.h`

AudioQueueOfflineRender

Exports audio to a buffer, instead of to a device, using a playback audio queue.

```
OSStatus AudioQueueOfflineRender (
    AudioQueueRef inAQ,
    const AudioTimeStamp *inTimeStamp,
    AudioQueueBufferRef ioBuffer,
    UInt32 inNumberFrames
);
```

Parameters*inAQ*

The playback audio queue.

inTimeStamp

The time corresponding to the beginning of the current audio queue buffer. This function uses the `mSampleTime` field of the `AudioTimeStamp` data structure.

ioBuffer

On input, a buffer you supply to hold rendered audio data. On output, the rendered audio data, which you can then write to a file.

inRequestedFrames

The number of frames of audio to render.

Return Value

A result code. See “[Audio Queue Result Codes](#)” (page 40).

Discussion

When you change a playback audio queue’s rendering mode to offline, using the [AudioQueueSetOfflineRenderFormat](#) (page 27) function, you gain access to the rendered audio. You can then write the audio to a file, rather than have it play to external hardware such as a loudspeaker.

Availability

Available in Mac OS X v10.5 and later.

See Also

[AudioQueueSetOfflineRenderFormat](#) (page 27)

[AudioQueueStart](#) (page 29)

Related Sample Code

AudioQueueTools

Declared In

AudioQueue.h

AudioQueuePause

Pauses audio playback or recording.

```
OSStatus AudioQueuePause (
    AudioQueueRef inAQ
);
```

Parameters

inAQ

The audio queue to pause.

Return Value

A result code. See “[Audio Queue Result Codes](#)” (page 40).

Discussion

Pausing an audio queue does not affect buffers or reset the audio queue. To resume playback or recording, call [AudioQueueStart](#) (page 29).

Availability

Available in Mac OS X v10.5 and later.

See Also

[AudioQueueStart](#) (page 29)

[AudioQueueStop](#) (page 30)

Declared In

AudioQueue.h

AudioQueuePrime

Decodes enqueued buffers in preparation for playback.

```
OSStatus AudioQueuePrime (
    AudioQueueRef inAQ,
    UInt32 inNumberOfFramesToPrepare,
    UInt32 *outNumberOfFramesPrepared
);
```

Parameters*inAQ*

The audio queue to be primed.

inNumberOfFramesToPrepare

The number of frames to decode before returning. Pass 0 to decode all enqueued buffers.

outNumberOfFramesPrepared

On output, the number of frames actually decoded and prepared for playback. Pass NULL on input if you are not interested in this information.

Return ValueA result code. See [“Audio Queue Result Codes”](#) (page 40).**Discussion**

This function decodes enqueued buffers in preparation for playback. It returns when at least the number of audio sample frames specified in *inNumberOfFramesToPrepare* are decoded and ready to play, or (if you pass 0 for the *inNumberOfFramesToPrepare* parameter), when all enqueued buffers are decoded.

To make a buffer of audio data ready to play, use `AudioQueuePrime` as follows:

1. Call [AudioQueueEnqueueBuffer](#) (page 14).
2. Call `AudioQueuePrime`.
3. Call [AudioQueueStart](#) (page 29).

Availability

Available in Mac OS X v10.5 and later.

Declared In`AudioQueue.h`**AudioQueueRemovePropertyListener**

Removes a property listener callback from an audio queue.

```
OSStatus AudioQueueRemovePropertyListener (
    AudioQueueRef inAQ,
    AudioQueuePropertyID inID,
    AudioQueuePropertyListenerProc inProc,
    void *inUserData
);
```

Parameters*inAQ*

The audio queue that you want to remove a property listener callback from.

inID

The ID of the property whose changes you no longer want to respond to. See [“Audio Queue Property Identifiers”](#) (page 37).

inProc

The callback to be removed.

inUserData

The same custom data for the property listener callback that you passed when calling [AudioQueueAddPropertyListener](#) (page 7).

Return Value

A result code. See [“Audio Queue Result Codes”](#) (page 40).

Availability

Available in Mac OS X v10.5 and later.

See Also

[AudioQueueAddPropertyListener](#) (page 7)

Declared In

AudioQueue.h

AudioQueueReset

Resets an audio queue.

```
OSStatus AudioQueueReset (
    AudioQueueRef inAQ
);
```

Parameters*inAQ*

The audio queue to reset.

Return Value

A result code. See [“Audio Queue Result Codes”](#) (page 40).

Discussion

This function immediately resets an audio queue, flushes any queued buffers (invoking callbacks as necessary), removes all buffers from previously scheduled use, and resets decoder and digital signal processing (DSP) state.

If you queue buffers after calling this function, processing does not begin until the decoder and DSP state of the audio queue are reset. This might create an audible discontinuity (or “glitch”).

This function is called automatically when you call [AudioQueueStop](#) (page 30).

Availability

Available in Mac OS X v10.5 and later.

See Also

[AudioQueuePause](#) (page 24)

[AudioQueueStop](#) (page 30)

Declared In

AudioQueue.h

AudioQueueSetOfflineRenderFormat

Sets the rendering mode and audio format for a playback audio queue.

```
OSStatus AudioQueueSetOfflineRenderFormat (
    AudioQueueRef inAQ,
    const AudioStreamBasicDescription *inFormat,
    const AudioChannelLayout *inLayout
);
```

Parameters*inAQ*

The playback audio queue whose rendering mode and audio format you want to set.

*inFormat*The audio format for offline rendering. The format must be some sort of linear PCM. If the format has more than one channel, it must be interleaved. For more information on the `AudioStreamBasicDescription` structure, see *Core Audio Data Types Reference*.Pass `NULL` to disable offline rendering and return the audio queue to normal output to an audio device.*inLayout*The channel layout for offline rendering. For more information on the `AudioChannelLayout` structure, see *Core Audio Data Types Reference*.Pass `NULL` when using this function to disable offline rendering.**Return Value**A result code. See [“Audio Queue Result Codes”](#) (page 40).**Discussion**

Use this function to set a playback audio queue to perform offline rendering, such as for export to an audio file. In offline rendering mode, a playback audio queue does not connect to external hardware.

You can also use this function to restore an audio queue to normal rendering mode by passing `NULL` in the *inFormat* and *inLayout* parameters.**Availability**

Available in Mac OS X v10.5 and later.

See Also[AudioQueueOfflineRender](#) (page 23)**Related Sample Code**

AudioQueueTools

Declared In

AudioQueue.h

AudioQueueSetParameter

Sets a playback audio queue parameter value.

```
OSStatus AudioQueueSetParameter (
    AudioQueueRef inAQ,
    AudioQueueParameterID inParamID,
    AudioQueueParameterValue inValue
);
```

Parameters*inAQ*

The playback audio queue that you want to set a parameter value on.

inParamID

The ID of the parameter you want to set. In Mac OS X v10.5, audio queues have one parameter available: `kAudioQueueParam_Volume`, which controls playback gain. See [“Audio Queue Parameter Identifiers”](#) (page 40).

inValue

The parameter value to set.

Return Value

A result code. See [“Audio Queue Result Codes”](#) (page 40).

Discussion

Use this function to change the settings for a playback audio queue directly. Changes take effect immediately. To set playback gain at the granularity of an audio queue buffer, use the [AudioQueueEnqueueBufferWithParameters](#) (page 15) function.

Availability

Available in Mac OS X v10.5 and later.

See Also

[AudioQueueGetParameter](#) (page 19)

[AudioQueueEnqueueBufferWithParameters](#) (page 15)

Related Sample Code

AudioQueueTools

Declared In

AudioQueue.h

AudioQueueSetProperty

Sets an audio queue property value.

```
OSStatus AudioQueueSetProperty (
    AudioQueueRef inAQ,
    AudioQueuePropertyID inID,
    const void *inData,
    UInt32 inDataSize
);
```

Parameters*inAQ*

The audio queue that you want to set a property value on.

inID

The ID of the property whose value you want to set. See [“Audio Queue Property Identifiers”](#) (page 37).

inData

The property value to set.

inDataSize

The size of the property data.

Return Value

A result code. See [“Audio Queue Result Codes”](#) (page 40).

Availability

Available in Mac OS X v10.5 and later.

See Also

[AudioQueueGetProperty](#) (page 20)

Related Sample Code

AudioFileStreamExample

AudioQueueTools

Declared In

AudioQueue.h

AudioQueueStart

Begins playing or recording audio.

```
OSStatus AudioQueueStart (
    AudioQueueRef inAQ,
    const AudioTimeStamp *inStartTime
);
```

Parameters*inAQ*

The audio queue to start.

inDeviceStartTime

The time at which the audio queue should start.

To specify a start time relative to the timeline of the associated audio device, use the `mSampleTime` field of the `AudioTimeStamp` structure. Use `NULL` to indicate that the audio queue should start as soon as possible.

Return Value

A result code. See [“Audio Queue Result Codes”](#) (page 40).

Discussion

If the associated audio device is not already running, this function starts it.

Availability

Available in Mac OS X v10.5 and later.

See Also

[AudioQueueStop](#) (page 30)

[AudioQueuePause](#) (page 24)

Related Sample Code

[AudioFileStreamExample](#)

[AudioQueueTools](#)

Declared In

[AudioQueue.h](#)

AudioQueueStop

Stops playing or recording audio.

```
OSStatus AudioQueueStop (
    AudioQueueRef inAQ,
    Boolean inImmediate
);
```

Parameters

inAQ

The audio queue to stop.

inImmediate

If you pass `true`, stopping occurs immediately (that is, *synchronously*). If you pass `false`, the function returns immediately, but the audio queue does not stop until its queued buffers are played or recorded (that is, the stop occurs *asynchronously*). Audio queue callbacks are invoked as necessary until the queue actually stops.

Return Value

A result code. See [“Audio Queue Result Codes”](#) (page 40).

Discussion

This function resets an audio queue, stops the audio hardware associated with the queue if it is not in use by other audio services, and stops the audio queue. When recording, this function is typically invoked by a user. When playing back, a playback audio queue callback should call this function when there is no more audio to play.

Availability

Available in Mac OS X v10.5 and later.

See Also

[AudioQueueStart](#) (page 29)

[AudioQueueReset](#) (page 26)

[AudioQueuePause](#) (page 24)

Related Sample Code

[AudioFileStreamExample](#)

[AudioQueueTools](#)

Declared In

[AudioQueue.h](#)

Callbacks by Task

Handling Audio Queue Buffers for Recording and Playback

[AudioQueueInputCallback](#) (page 31)

Called by the system when a recording audio queue has finished filling an audio queue buffer.

[AudioQueueOutputCallback](#) (page 32)

Called by the system when an audio queue buffer is available for reuse.

Defining a Property Listener

[AudioQueuePropertyListenerProc](#) (page 33)

Called by the system when a specified audio queue property changes value.

Callbacks

AudioQueueInputCallback

Called by the system when a recording audio queue has finished filling an audio queue buffer.

```
typedef void (*AudioQueueInputCallback) (
    void                    *inUserData,
    AudioQueueRef           inAQ,
    AudioQueueBufferRef     inBuffer,
    const AudioTimeStamp    *inStartTime,
    UInt32                  inNumberPacketDescriptions,
    const AudioStreamPacketDescription *inPacketDescs
);
```

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

```
void MyAudioQueueInputCallback (
    void                    *inUserData,
    AudioQueueRef           inAQ,
    AudioQueueBufferRef     inBuffer,
    const AudioTimeStamp    *inStartTime,
    UInt32                  inNumberPacketDescriptions,
    const AudioStreamPacketDescription *inPacketDescs
);
```

Parameters

inUserData

The custom data you've specified in the `inUserData` parameter of the [AudioQueueOfflineRender](#) (page 23) function. Typically, this includes format and state information for the audio queue.

inAQ

The recording audio queue that invoked the callback.

inBuffer

An audio queue buffer, newly filled by the recording audio queue, containing the new audio data your callback needs to write.

inStartTime

The sample time for the start of the audio queue buffer. This parameter is not used in basic recording.

inNumberPacketDescriptions

The number of packets of audio data sent to the callback in the *inBuffer* parameter. When recording in a constant bit rate (CBR) format, the audio queue sets this parameter to NULL.

inPacketDescs

For compressed formats that require packet descriptions, the set of packet descriptions produced by the encoder for audio data in the *inBuffer* parameter. When recording in a CBR format, the audio queue sets this parameter to NULL.

Discussion

You specify a recording audio queue callback when calling the [AudioQueueOfflineRender](#) (page 23) function. The callback is invoked each time its recording audio queue has filled an audio queue buffer with fresh audio data. Typically, your callback writes the data to a file or other buffer, and then reenqueues the audio queue buffer to receive more data.

Availability

Available in Mac OS X v10.5 and later.

Declared In

AudioQueue.h

AudioQueueOutputCallback

Called by the system when an audio queue buffer is available for reuse.

```
typedef void (*AudioQueueOutputCallback) (
    void                *inUserData,
    AudioQueueRef       inAQ,
    AudioQueueBufferRef inBuffer
);
```

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

```
void MyAudioQueueOutputCallback (
    void                *inUserData,
    AudioQueueRef       inAQ,
    AudioQueueBufferRef inBuffer
);
```

Parameters*inUserData*

The custom data you've specified in the *inUserData* parameter of the [AudioQueueSetOfflineRenderFormat](#) (page 27) function. Typically, this includes format and state information for the audio queue.

inAQ

The playback audio queue that invoked the callback.

inBuffer

An audio queue buffer, newly available to fill because the playback audio queue has acquired its contents.

Discussion

You specify a callback function when creating a new playback audio queue with the [AudioQueueSetOfflineRenderFormat](#) (page 27) function. The callback is invoked each time its playback audio queue has acquired the data from an audio queue buffer, at which point the buffer is available for reuse. Typically, your callback then fills the newly-available buffer with data from a file or other buffer, and then reenqueues the buffer for playback.

When your application receives a call to the `AudioQueueOutputCallback` function, you cannot assume that the audio data from the newly-available buffer has been played. To ensure that all queued audio has finished playing, use the [AudioQueuePropertyListenerProc](#) (page 33) property listener callback function, set to listen to the `kAudioQueueProperty_IsRunning` (page 38) property.

Availability

Available in Mac OS X v10.5 and later.

Declared In

`AudioQueue.h`

AudioQueuePropertyListenerProc

Called by the system when a specified audio queue property changes value.

```
typedef void (*AudioQueuePropertyListenerProc) (
    void                *inUserData,
    AudioQueueRef       inAQ,
    AudioQueuePropertyID inID
);
```

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

```
void MyAudioQueuePropertyListenerProc (
    void                *inUserData,
    AudioQueueRef       inAQ,
    AudioQueuePropertyID inID
);
```

Parameters

inUserData

The custom data you've specified in the `inUserData` parameter of the [AudioQueueAddPropertyListener](#) (page 7) function.

inAQ

The recording or playback audio queue that invoked the callback.

inID

The ID of the property whose value changes you want to observe.

Discussion

Install this callback in an audio queue by calling the [AudioQueueAddPropertyListener](#) (page 7) function. For example, your application can get a notification that audio has finished playing by:

1. Defining this property listener callback function to listen for changes to the [kAudioQueueProperty_IsRunning](#) (page 38) property.
2. Installing this callback function in an audio queue by calling the [AudioQueueAddPropertyListener](#) (page 7) function.

Availability

Available in Mac OS X v10.5 and later.

Declared In

AudioQueue.h

Data Types

AudioQueueBuffer

Defines an audio queue buffer.

```
typedef struct AudioQueueBuffer {
    const UInt32          mAudioDataBytesCapacity;
    void                 *const mAudioData;
    UInt32               mAudioDataByteSize;
    void                 *mUserData;
    const UInt32         mPacketDescriptionCapacity;
    AudioStreamPacketDescription *const mPacketDescriptions;
    UInt32               mPacketDescriptionCount;
} AudioQueueBuffer;
typedef AudioQueueBuffer *AudioQueueBufferRef;
```

Fields

`mAudioDataBytesCapacity`

The size of the audio queue buffer, in bytes. This size is set when a buffer is allocated and cannot be changed.

`mAudioData`

The audio data owned the audio queue buffer. The buffer address cannot be changed.

`mAudioDataByteSize`

The number of bytes of valid audio data in the audio queue buffer's `mAudioData` field, initially set to 0. Your callback must this value for a playback audio queue; for recording, the recording audio queue sets the value.

`mUserData`

The custom data structure you specify, for use by your callback function, when creating a recording or playback audio queue.

`mPacketDescriptionCapacity`

The maximum number of packet descriptions that can be stored in the `mPacketDescriptions` field.

`mPacketDescriptions`

An array of `AudioStreamPacketDescription` structures for the buffer.

`mPacketDescriptionCount`

The number of valid packet descriptions in the buffer. You set this value when providing buffers for playback. The audio queue sets this value when returning buffers from a recording queue.

Discussion

Each audio queue has an associated set of audio queue buffers. To allocate a buffer, call the [AudioQueueAllocateBuffer](#) (page 8) function. To dispose of a buffer, call the [AudioQueueFreeBuffer](#) (page 17) function.

If using a VBR compressed audio data format, you may want to instead use the `AudioQueueAllocateBufferWithPacketDescriptions` function. This function allocates a buffer with additional space for packet descriptions. The `mPacketDescriptionCapacity`, `mPacketDescriptions`, and `mPacketDescriptionCount` fields may only be used with buffers allocated with `AudioQueueAllocateBufferWithPacketDescriptions`.

Availability

Available in Mac OS X v10.5 and later.

Declared In

`AudioQueue.h`

AudioQueueBufferRef

A pointer to an audio queue buffer.

```
typedef AudioQueueBuffer *AudioQueueBufferRef;
```

Availability

Available in Mac OS X v10.5 and later.

Declared In

`AudioQueue.h`

AudioQueueRef

Defines an opaque data type that represents an audio queue.

```
typedef struct OpaqueAudioQueue *AudioQueueRef;
```

Discussion

An audio queue is a software object you use for recording or playing audio in Mac OS X. It does the work of:

- Connecting to audio hardware
- Managing memory
- Employing codecs, as needed, for compressed audio formats
- Mediating recording or playback

You create, use, and dispose of audio queues using the functions described in [“Audio Queue Functions”](#) (page 7).

Availability

Available in Mac OS X v10.5 and later.

Declared In

AudioQueue.h

AudioQueueTimelineRef

Defines an opaque data type that represents an audio queue timeline object.

```
typedef struct OpaqueAudioQueueTimeline *AudioQueueTimelineRef;
```

Discussion

You can use a timeline object to observe time discontinuities in the audio hardware device associated with an audio queue. A discontinuity is, for example, a period of silence when sound was expected. Causes of discontinuities include changes in device state or data processing overloads. See Technical Q&A 1467, *CoreAudio Overload Warnings*. You query a timeline object by passing it as a parameter to the [AudioQueueGetCurrentTime](#) (page 18) function.

Availability

Available in Mac OS X v10.5 and later.

Declared In

AudioQueue.h

AudioQueueLevelMeterState

Specifies the current level metering information for one channel of an audio queue..

```
typedef struct AudioQueueLevelMeterState {
    Float32    mAveragePower;
    Float32    mPeakPower;
}; AudioQueueLevelMeterState;
```

Fields

mAveragePower

The audio channel's average RMS power.

mPeakPower

The audio channel's peak RMS power.

Availability

Available in Mac OS X v10.5 and later.

Declared In

AudioQueue.h

AudioQueueParameterEvent

Specifies an audio queue parameter and associated value.

```
struct AudioQueueParameterEvent {
    AudioQueueParameterID    mID;
    AudioQueueParameterValue  mValue;
}; typedef struct AudioQueueParameterEvent AudioQueueParameterEvent;
```

Fields

mID

The parameter.

mValue

The value of the specified parameter.

Discussion

You use this structure with the [AudioQueueEnqueueBufferWithParameters](#) (page 15) function. See that function, and [“Audio Queue Parameter Identifiers”](#) (page 40), for more information.

Availability

Available in Mac OS X v10.5 and later.

Declared In

AudioQueue.h

AudioQueueParameterID

A UInt32 value that uniquely identifies an audio queue parameter.

```
typedef UInt32 AudioQueueParameterID;
```

Availability

Available in Mac OS X v10.5 and later.

Declared In

AudioQueue.h

AudioQueueParameterValue

A Float32 value for an audio queue parameter.

```
typedef Float32 AudioQueueParameterValue;
```

Availability

Available in Mac OS X v10.5 and later.

Declared In

AudioQueue.h

Constants

Audio Queue Property Identifiers

Identifiers for audio queue properties.

```
enum {
    kAudioQueueProperty_IsRunning           = 'aqrn',
    kAudioQueueDeviceProperty_SampleRate   = 'aqsr',
    kAudioQueueDeviceProperty_NumberChannels = 'aqdc',
    kAudioQueueProperty_CurrentDevice      = 'aqcd',
    kAudioQueueProperty_MagicCookie        = 'aqmc',
    kAudioQueueProperty_MaximumOutputPacketSize = 'xops',
    kAudioQueueProperty_StreamDescription   = 'aqft',
    kAudioQueueProperty_ChannelLayout      = 'aqcl',
    kAudioQueueProperty_EnableLevelMetering = 'aqme',
    kAudioQueueProperty_CurrentLevelMeter   = 'aqmv',
    kAudioQueueProperty_CurrentLevelMeterDB = 'aqmd',
    kAudioQueueProperty_DecodeBufferSizeFrames = 'dcbf'
};
typedef UInt32 AudioQueuePropertyID;
```

Constants

`kAudioQueueProperty_IsRunning`

A read-only property whose value is a `UInt32`. Any nonzero value means running, and 0 means stopped. A notification is sent when the associated audio queue starts or stops, which may occur sometime after the [AudioQueueStart](#) (page 29) or [AudioQueueStop](#) (page 30) function is called.

Available in Mac OS X v10.5 and later.

Declared in `AudioQueue.h`.

`kAudioQueueDeviceProperty_SampleRate`

A read-only property whose value is a `Float64`. The value is the sampling rate of the audio hardware device associated with an audio queue.

Available in Mac OS X v10.5 and later.

Declared in `AudioQueue.h`.

`kAudioQueueDeviceProperty_NumberChannels`

A read-only property whose value is a `UInt32`. The value is the number of channels in the audio hardware device associated with an audio queue.

Available in Mac OS X v10.5 and later.

Declared in `AudioQueue.h`.

`kAudioQueueProperty_CurrentDevice`

A read-write property whose value is of type `CFStringRef`. The value contains the unique identifier (UID) of the audio hardware device associated with an audio queue.

Available in Mac OS X v10.5 and later.

Declared in `AudioQueue.h`.

`kAudioQueueProperty_MagicCookie`

A read-write property whose value is a void pointer to a block of memory, which you set up, containing an audio format magic cookie. If the audio format you are playing or recording to requires a magic cookie, you must set a value for this property before enqueueing any buffers.

Available in Mac OS X v10.5 and later.

Declared in `AudioQueue.h`.

`kAudioQueueProperty_MaximumOutputPacketSize`

A read-only property whose value is a `UInt32`. The value is the size, in bytes, of the largest single packet of data in the output format. Primarily useful when encoding VBR compressed data.

Available in Mac OS X v10.6 and later.

Declared in `AudioQueue.h`.

kAudioQueueProperty_StreamDescription

A read-only property whose value is an `AudioStreamBasicDescription` structure, indicating an audio queue's data format. Primarily useful for obtaining a complete ASBD when recording, in cases where you initially specify a sample rate of 0.

Available in Mac OS X v10.6 and later.

Declared in `AudioQueue.h`.

kAudioQueueProperty_ChannelLayout

A read-write property whose value is an `AudioChannelLayout` structure that describes an audio queue channel layout. The number of channels in the layout must match the number of channels in the audio format. This property is typically not used in the case of one or two channel audio. For more than two channels (such as in the case of 5.1 surround sound), you may need to specify a channel layout to indicate channel order, such as left, then center, then right.

Available in Mac OS X v10.5 and later.

Declared in `AudioQueue.h`.

kAudioQueueProperty_EnableLevelMetering

A read-write property whose value is a `UInt32` that indicates whether audio level metering is enabled for an audio queue. 0 = metering off, 1 = metering on.

Available in Mac OS X v10.5 and later.

Declared in `AudioQueue.h`.

kAudioQueueProperty_CurrentLevelMeter

A read-only property whose value is an array of `AudioQueueLevelMeterState` (page 36) structures, one array element per audio channel. The member values in the structure are in the range 0 (for silence) to 1 (indicating maximum level).

Available in Mac OS X v10.5 and later.

Declared in `AudioQueue.h`.

kAudioQueueProperty_CurrentLevelMeterDB

A read-only property whose value is an array of `AudioQueueLevelMeterState` (page 36) structures, one array element per audio channel. The member values in the structure are in decibels.

Available in Mac OS X v10.5 and later.

Declared in `AudioQueue.h`.

kAudioQueueProperty_DecodeBufferSizeFrames

A read-write property whose value is a `UInt32` that is the size of the buffer into which a playback (output) audio queue decodes buffers. A larger buffer provides more reliability and better long-term performance at the expense of memory and decreased responsiveness in some situations.

Available in Mac OS X v10.6 and later.

Declared in `AudioQueue.h`.

Discussion

To receive a notification that a specific audio queue property has changed:

1. Define a property listener callback, referencing the desired audio queue property ID. Base the callback on the `AudioQueuePropertyListenerProc` (page 33) callback function declaration.
2. Assign the callback to an audio queue using the `AudioQueueAddPropertyListener` (page 7) function.
3. When you get a property-changed notification, call the `AudioQueueGetProperty` (page 20) function to get the current value of the property.

Declared In

AudioQueue.h

Audio Queue Parameter Identifiers

Identifiers for audio queue parameters.

```
enum {
    kAudioQueueParam_Volume = 1
};
typedef UInt32 AudioQueueParameterID;
```

Constants

kAudioQueueParam_Volume

A value from 0.0 to 1.0 indicating the linearly scaled gain for an audio queue. A value of 1.0 (the default) indicates unity gain. A value of 0.0 indicates zero gain, or silence.

Available in Mac OS X v10.5 and later.

Declared in AudioQueue.h.

Discussion

In Mac OS X v10.5, audio queues have one parameter available: kAudioQueueParam_Volume. This parameter applies only to playback audio queues.

You can set a playback audio queue parameter in one of two ways:

- Set the value to take effect immediately using the [AudioQueueSetParameter](#) (page 27) function.
- Schedule a value to take effect when a particular audio queue buffer plays. You supply the parameter when you enqueue the buffer. The new value is applied to the audio queue that owns the buffer when that buffer is rendered.

The [AudioQueueGetParameter](#) (page 19) function always returns the current value of the parameter for an audio queue.

Declared In

AudioQueue.h

Result Codes

This table lists result codes defined for Audio Queue Services.

Result Code	Value	Description
kAudioQueueErr_InvalidBuffer	-66687	The specified audio queue buffer does not belong to the specified audio queue. Available in Mac OS X v10.5 and later.

Result Code	Value	Description
kAudioQueueErr_BufferEmpty	-66686	The audio queue buffer is empty (that is, the <code>mAudioDataByteSize</code> field = 0). Available in Mac OS X v10.5 and later.
kAudioQueueErr_DisposalPending	-66685	The function cannot act on the audio queue because it is being asynchronously disposed of. Available in Mac OS X v10.5 and later.
kAudioQueueErr_InvalidProperty	-66684	The specified property ID is invalid. Available in Mac OS X v10.5 and later.
kAudioQueueErr_InvalidPropertySize	-66683	The size of the specified property is invalid. Available in Mac OS X v10.5 and later.
kAudioQueueErr_InvalidParameter	-66682	The specified parameter ID is invalid. Available in Mac OS X v10.5 and later.
kAudioQueueErr_CannotStart	-66681	The audio queue has encountered a problem and cannot start. Available in Mac OS X v10.5 and later.
kAudioQueueErr_InvalidDevice	-66680	The specified audio hardware device could not be located. Available in Mac OS X v10.5 and later.
kAudioQueueErr_BufferInQueue	-66679	The audio queue buffer cannot be disposed of when it is enqueued. Available in Mac OS X v10.5 and later.
kAudioQueueErr_InvalidRunState	-66678	The queue is running but the function can only operate on the queue when it is stopped, or vice versa. Available in Mac OS X v10.5 and later.
kAudioQueueErr_InvalidQueueType	-66677	The queue is an input queue but the function can only operate on an output queue, or vice versa. Available in Mac OS X v10.5 and later.
kAudioQueueErr_Permissions	-66676	You do not have the required permissions to call the function. Available in Mac OS X v10.6 and later.

Result Code	Value	Description
kAudioQueueErr_InvalidPropertyValue	-66675	The property value used is not valid. Available in Mac OS X v10.6 and later.
kAudioQueueErr_PrimeTimedOut	-66674	During a call to the AudioQueuePrime (page 24) function, the audio queue's audio converter failed to convert the requested number of sample frames. Available in Mac OS X v10.6 and later.
kAudioQueueErr_CodecNotFound	-66673	The requested codec was not found. Available in Mac OS X v10.6 and later.
kAudioQueueErr_EnqueueDuringReset	-66632	During a call to the AudioQueueReset (page 26), AudioQueueStop (page 30), or AudioQueueDispose (page 13) functions, the system does not allow you to enqueue buffers. Available in Mac OS X v10.6 and later.
kAudioFormatUnsupportedDataFormatError	1718449215 = 'fmt?'	The playback data format is unsupported (declared in <code>AudioFormat.h</code>). Available in Mac OS X v10.3 and later.

Document Revision History

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

Date	Notes
2009-08-07	Added description for “kAudioQueueErr_InvalidOfflineMode” result code.
2009-06-19	Added description for AudioQueueBufferRef (page 35) type definition.
2009-03-29	Updated for iPhone OS 3.0.
	Added descriptions for new result codes: “kAudioQueueErr_QueueInvalidated” and “kAudioQueueErr_EnqueueDuringReset” (page 42). Added descriptions for constants that support the use of hardware codecs: “Hardware Codec Policy Keys.”
2008-11-12	Added description for new “kAudioQueueErr_PrimeTimedOut” (page 42) result code.
2008-07-08	Updated for platform-specific support.
2008-01-15	Corrected and clarified descriptions of the AudioQueueOutputCallback (page 32) and AudioQueuePropertyListenerProc (page 33) callback functions.
2007-10-31	New document that describes a high-level programming interface for playing and recording audio data.

REVISION HISTORY

Document Revision History

Index

A

Audio Queue Parameter Identifiers 40
Audio Queue Property Identifiers 37
AudioQueueAddPropertyListener function 7
AudioQueueAllocateBuffer function 8
AudioQueueAllocateBufferWithPacketDescriptions function 9
AudioQueueBuffer structure 34
AudioQueueBufferRef data type 35
AudioQueueCreateTimeline function 10
AudioQueueDeviceGetCurrentTime function 11
AudioQueueDeviceGetNearestStartTime function 11
AudioQueueDeviceTranslateTime function 12
AudioQueueDispose function 13
AudioQueueDisposeTimeline function 13
AudioQueueEnqueueBuffer function 14
AudioQueueEnqueueBufferWithParameters function 15
AudioQueueFlush function 17
AudioQueueFreeBuffer function 17
AudioQueueGetCurrentTime function 18
AudioQueueGetParameter function 19
AudioQueueGetProperty function 20
AudioQueueGetPropertySize function 20
AudioQueueInputCallback callback 31
AudioQueueLevelMeterState structure 36
AudioQueueNewInput function 21
AudioQueueNewOutput function 22
AudioQueueOfflineRender function 23
AudioQueueOutputCallback callback 32
AudioQueueParameterEvent structure 36
AudioQueueParameterID data type 37
AudioQueueParameterValue data type 37
AudioQueuePause function 24
AudioQueuePrime function 24
AudioQueuePropertyListenerProc callback 33
AudioQueueRef data type 35
AudioQueueRemovePropertyListener function 25
AudioQueueReset function 26

AudioQueueSetOfflineRenderFormat function 27
AudioQueueSetParameter function 27
AudioQueueSetProperty function 28
AudioQueueStart function 29
AudioQueueStop function 30
AudioQueueTimelineRef data type 36

K

kAudioFormatUnsupportedDataFormatError constant 42
kAudioQueueDeviceProperty_NumberChannels constant 38
kAudioQueueDeviceProperty_SampleRate constant 38
kAudioQueueErr_BufferEmpty constant 41
kAudioQueueErr_BufferInQueue constant 41
kAudioQueueErr_CannotStart constant 41
kAudioQueueErr_CodecNotFound constant 42
kAudioQueueErr_DisposalPending constant 41
kAudioQueueErr_EnqueueDuringReset constant 42
kAudioQueueErr_InvalidBuffer constant 40
kAudioQueueErr_InvalidDevice constant 41
kAudioQueueErr_InvalidParameter constant 41
kAudioQueueErr_InvalidProperty constant 41
kAudioQueueErr_InvalidPropertySize constant 41
kAudioQueueErr_InvalidPropertyValue constant 42
kAudioQueueErr_InvalidQueueType constant 41
kAudioQueueErr_InvalidRunState constant 41
kAudioQueueErr_Permissions constant 41
kAudioQueueErr_PrimeTimedOut constant 42
kAudioQueueParam_Volume constant 40
kAudioQueueProperty_ChannelLayout constant 39
kAudioQueueProperty_CurrentDevice constant 38
kAudioQueueProperty_CurrentLevelMeter constant 39
kAudioQueueProperty_CurrentLevelMeterDB constant 39
kAudioQueueProperty_DecodeBufferSizeFrames constant 39

kAudioQueueProperty_EnableLevelMetering
constant 39

kAudioQueueProperty_IsRunning constant 38

kAudioQueueProperty_MagicCookie constant 38

kAudioQueueProperty_MaximumOutputPacketSize
constant 38

kAudioQueueProperty_StreamDescription constant
39