
Sound Programming Topics for Cocoa

Audio & Video: Audio



2006-11-07



Apple Inc.
© 2006 Apple Computer, 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, Cocoa, Mac, and Mac OS are trademarks of Apple Inc., registered in the United States and other countries.

NeXT is a trademark of NeXT Software, Inc., registered in the United States and other countries.

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

Introduction to Sound Programming Topics for Cocoa 7

Organization of This Document 7

Overview of the NSSound Class 9

Loading Audio Data 11

Loading Sounds By Name 11

Loading Sounds By Pathname 11

Loading Sounds By URL 12

Loading Sounds Using an Open Dialog 12

Managing Sound Playback 13

Starting, Pausing, Resuming, and Canceling Sound Playback 13

Finding Out Whether a Sound Is Playing 13

Finding Out When a Sound Has Finished Playing 14

Document Revision History 15

Listings

Loading Audio Data 11

Listing 1 Loading a sound file using an open dialog 12

Managing Sound Playback 13

Listing 1 Controlling sound playback 13

Listing 2 Determining whether a sound is playing 13

Listing 3 Performing an action when a sound finishes playing 14

Introduction to Sound Programming Topics for Cocoa

This document contains articles that describe Cocoa's audio support.

Organization of This Document

This document contains the following articles:

- [“Overview of the NSSound Class”](#) (page 9) describes the `NSSound` class.
- [“Loading Audio Data”](#) (page 11) explains various ways to load audio data into an `NSSound` instance.
- [“Managing Sound Playback”](#) (page 13) explains how to play audio with an `NSSound` instance.

This document also contains a revision history.

Overview of the NSSound Class

The `NSSound` class makes it extremely simple for Cocoa applications to load and play soundfiles. Instance methods provide standard transport control so that a sound can be programmatically started, stopped and paused.

The class supports the following file and data formats:

- File Formats:
 - AIFF
 - WAV
 - NeXT SND

- Data Formats:
 - 16 bit
 - 44.1 Khz
 - 22.05 KHz
 - Mono
 - Stereo

You can load audio data into an `NSSound` object from three sources:

1. a disk file—using a pathname or URL
2. network connection—using a URL
3. the pasteboard

The `NSSound` class can search for named sound resources in the application's main bundle as well as two standard file system locations: `/Library/Sounds` and `~/Library/Sounds`.

Loading Audio Data

There are several ways to initialize an `NSSound` instance with audio data. Note that `NSSound` objects are immutable so once you have created an instance with one of the `initWith...` methods, you cannot change the instance or associate different audio data with it.

Loading Sounds By Name

The simplest way to create an `NSSound` instance is using the `soundNamed:` class method. Named sounds are instances of `NSSound` that have been given a name using the `setName:` method. The `NSSound` class keeps a list of the named instances created by your application, as well as the named system sounds provided by the Application Kit. The system sounds (located in `/System/Library/Sounds`) have been named by the Application Kit using their filenames without the file extension. The following code listing shows you how to load a system sound by name:

```
NSSound *mySound = [NSSound soundNamed:@"Temple"];
```

The `soundNamed:` method first searches for an existing sound file with the name you've specified, and if one is found, it is returned to you. Since this example refers to a standard system sound, there is no need to search further.

The following code listing shows you how to load a soundfile from disk using `soundNamed:`.

```
NSSound *airplaneSound = [NSSound soundNamed:@"Airplane_44KStereo"];
```

If there is no known sound with the name you've specified, `soundNamed:` searches your application's main bundle, and then the `/Library/Sounds` and `~/Library/Sounds` directories for soundfiles with the specified name. If no data can be found for *name*, no object is created and `nil` is returned.

Note that once again the file extension is not used when loading soundfiles using the `soundNamed:` method. Also note that AIFF files must use the `.aiff` file extension (not `.aif`) in order to be located by `soundNamed:`.

Loading Sounds By Pathname

If you want to load sound files from someplace in the file system other than your application's bundle, you can use the method `initWithContentsOfFile:byReference:`. As you would expect, this method attempts to initialize a newly allocated `NSSound` instance with the audio data in the specified file. If the *byReference:* parameter is `YES`, only the name of the sound is stored with the `NSSound` instance when archived using `encodeWithCoder:`, otherwise the audio data is archived along with the instance. The following code listing shows you how to create an `NSSound` instance and initialize it with a soundfile given a pathname to the file.

```
sound = [[NSSound alloc] initWithContentsOfFile:@"Volumes/Audio/Truck.aiff"  
byReference:YES];
```

Loading Sounds By URL

Loading a sound from a URL is very similar to using a pathname as demonstrated in the section “[Loading Sounds By Pathname](#)” (page 11). The code listing below shows how to use a URL instead of a pathname. Note that only file system URLs are currently supported.

```
NSURL *soundfileURL = [NSURL
fileURLWithString:@"file://~/soundfiles/guitar.aiff"];
NSSound *sound = [[NSSound alloc] initWithContentsOfURL:soundfileURL
byReference:NO];
```

Loading Sounds Using an Open Dialog

Listing 1 shows how to load a sound file using an `NSOpenPanel` object.

Listing 1 Loading a sound file using an open dialog

```
- (IBAction)loadSoundOpenPanel:(id)sender
{
    int result;
    NSOpenPanel *oPanel = [NSOpenPanel openPanel];
    NSArray *filesToOpen;
    NSString *theFileName;
    NSMutableArray *fileTypes = [NSSound soundUnfilteredFileTypes];
    // All file types NSSound understands

    [oPanel setAllowsMultipleSelection:NO];

    result = [oPanel runModalForDirectory:NSHomeDirectory() file:nil
types:fileTypes];

    if (result == NSOKButton) {
        filesToOpen = [oPanel filenames];
        theFileName = [filesToOpen objectAtIndex:0];
        NSLog(@"Open Panel Returned: %@.\n", theFileName);

        [self _loadSoundFromPath:theFileName];
    } else
        [infoTextField setStringValue:@"Sound failed to load..."];
}
```

Managing Sound Playback

This article describes how to manage the playback of a sound using the `NSSound` class.

Starting, Pausing, Resuming, and Canceling Sound Playback

Playing audio data using the `NSSound` class is very simple; instance methods provide transport control. Listing 1 shows several action methods that control the playback of a sound.

Listing 1 Controlling sound playback

```
-(IBAction) playSound:(id)sender
{
    if (loaded && ![sound isPlaying]) {
        [sound play];
        [infoTextField setStringValue:@"Playback in progress"];
    }
}

-(IBAction) pauseSound:(id)sender
{
    [sound pause];
    [infoTextField setStringValue:@"Playback paused"];
}

-(IBAction) resumeSound:(id)sender
{
    [sound resume];
    [infoTextField setStringValue:@"Playback resumed"];
}

-(IBAction) stopSound:(id)sender
{
    [sound stop];
    [infoTextField setStringValue:@"Playback canceled"];
}
```

Finding Out Whether a Sound Is Playing

The `isPlaying` method tells you whether a sound is playing, as shown in Listing 2.

Listing 2 Determining whether a sound is playing

```
-(IBAction) isSoundPlaying:(id)sender
{
```

```
if ([sound isPlaying])
    [infoTextField setStringValue:@"The sound is playing"];
else
    [infoTextField setStringValue:@"The sound is not playing"];
}
```

Finding Out When a Sound Has Finished Playing

Listing 3 shows an example implementation of the `sound:didFinishPlaying:delegate` method, which is called when a sound finishes playing.

Listing 3 Performing an action when a sound finishes playing

```
- (void) sound:(NSSound *)sound didFinishPlaying:(BOOL)playbackSuccessful
{
    if (playbackSuccessful) {
        [infoTextField setStringValue:@"Playback ended successfully"];
    }
    else {
        [infoTextField setStringValue:@"Playback ended abnormally"];
    }
}
```

Document Revision History

This table describes the changes to *Sound Programming Topics for Cocoa*.

Date	Notes
2006-11-07	Updated example code.
	Updated code listings in “Managing Sound Playback” (page 13).
2002-11-12	Revision history was added to existing document. It will be used to record changes to the content of the document.

