initFromSection:
initFromPasteboard:
initFromSoundfile:
free

Accessing the Sound name table+ addName:sound:
+ findSoundFor:
+ removeSoundForName:

Accessing the Sound's name setName:
name

Reading and writing sound data readSoundfile:
readSoundFromStream:
writeSoundfile:
writeSoundToStream:
writeToPasteboard:

Modifying sound data convertToFormat:samplingRate:channelCount:
convertToFormat:
setDataSize:dataFormat:samplingRate:
channelCount:infoSize:
setSoundStruct:soundStructSize:
setName:

duration
info
infoSize
isEmpty
compatibleWith:
processingError

Recording and playing pause

pause:
isPlayable
play
play:
record
record:
resume
resume:
stop
stop:
samplesProcessed
status
soundBeingProcessed
soundStructBeingProcessed

Editing sound data isEditable

copySamples:at:count:
copySound:
deleteSamples
deleteSamplesAt:count:
insertSamples:at:
needsCompacting
compactSamples

Archiving the object finishUnarchiving

read:
write:

Accessing the delegate setDelegate:

delegate
tellDelegate:

Accessing the sound hardware+ getVolume::

+ setVolume::
+ isMuted
+ setMute:

(int)channelCount

Returns the number of channels in the Sound.

(int)compactSamples

(BOOL)compatibleWith:aSound

Returns YES if the format, sampling rate, and channel count of aSound's sound data is the same as receiving this message. If one (or both) of the Sounds doesn't contain a sound (its soundStruct is n declared compatible and YES is returned.

(int)convertToFormat:(int)newFormat

This is the same as convertToFormat:samplingRate:channelCount:, except that only the format is c is returned.

(int)convertToFormat:(int)newFormat
samplingRate:(double)newRate
channelCount:(int)newChannelCount

Convert the Sound's data to the given format, sampling rate, and number of channels. The followi possible:

·Arbitrary sampling rate conversion.
·Compression and decompression.
·Floating-point formats (including double-precision) to and from linear formats.
·Mono to stereo.
·CODEC mu-law to and from linear formats.

An error code is returned.

(int)copySamples:aSound
at:(int)startSample
count:(int)sampleCount

Replaces the Sound's sampled data with a copy of a portion of aSound's data. The copied portion startSample'th sample (zero-based) and extends over sampleCount samples. The Sound receiving editable and the two Sounds must be compatible. If the specified portion of aSound is fragmented, this message will also be fragmented. An error code is returned.

(int)copySound:aSound

Replaces the Sound's data with a copy of aSound's data. The Sound receiving this message needn the two Sounds be compatible. An error code is returned.

(unsigned char *)data

Returns a pointer to the Sound's sampled data. You can use the pointer to examine, create, and mo intelligently manipulate the data, you need to be aware of its size, format, sampling rate, and the nu it contains (a query method for each of these attributes is provided by the Sound class). The size o must be respected it's set when the Sound is created or given a new sound (through readSoundfile:

Returns the format of the Sound's data. If the data is fragmented, the format of the samples is retu
SND_FORMAT_INDIRECT is never returned by this method).

(int)dataSize

Return the size (in bytes) of the Sound's data. If you modify the data (through the pointer returned
you must be careful not to exceed its length. If the sound is fragmented, the value returned by this
the Sound's soundStruct and doesn't include the actual data itself.

delegate

Returns the Sound's delegate.

(int)deleteSamples

Deletes all the samples in the Sound's data. The Sound must be editable. An error code is returne

(int)deleteSamplesAt:(int)startSample count:(int)sampleCount

Deletes a range of samples from the Sound: sampleCount samples are deleted starting with the sta
(zero-based). The Sound must be editable and may become fragmented. An error code is returned

(double)duration

Returns the Sound's length in seconds.

finishUnarchiving

You never invoke this method. It's invoked automatically by the read: method to tie up loose ends
Sound.

free

Frees the Sound and deallocates its sound data. The Sound is removed from the named Sound tabl
eligible for reuse.

(char *)info

Returns a pointer to the Sound's info string.

initFromPasteboard:(Pasteboard *)thePboard

Initializes the Sound instance, which must be newly allocated, by copying the sound data from the thePboard. (A Pasteboard can have only one sound entry at a time.) Returns self (an unnamed Sou currently contains a sound entry otherwise, frees the newly allocated Sound and returns nil.

initFromSection:(const char *)sectionName

Initializes the Sound instance, which must be newly allocated, by copying the sound data from sec sound segment of the application's executable file. If the section isn't found, the object looks for a sectionName in the same directory as the application's executable. Returns self (an unnamed Sou was successfully copied otherwise, frees the newly allocated Sound and returns nil.

initFromSoundfile:(const char *)filename

Initializes the Sound instance, which must be newly allocated, from the soundfile filename. Retur Sound) if the file was successfully read otherwise, frees the newly allocated Sound and returns nil.

(int)insertSamples:aSound at:(int)startSample

Pastes the sound data in aSound into the Sound receiving this message, starting at the receiving So sample (zero-based). The receiving Sound doesn't lose any of its original sound dataÐthe sample to startSample are moved to accommodate the inserted sound data. The receiving Sound must be e Sounds must be compatible (as determined by isCompatible:). If the method is successful, the rece fragmented. An error code is returned.

(BOOL)isEditable

Returns YES if the Sound's format indicates that it can be edited, otherwise returns NO.

(BOOL)isEmpty

Returns YES if the Sound doesn't contain any sound data, otherwise returns NO. This always retu isn't editable (as determined by sending it the isEditable message).

(BOOL)isPlayable

Returns YES if the Sound can be played, otherwise returns NO. Some unplayable Sounds just nee another format, sampling rate, or number of channels others are inherently unplayable, such as tho

(BOOL)needsCompacting

Returns YES if the Sound's data is fragmented. Otherwise returns NO.

(int)pause

Pauses the Sound during recording or playback. An error code is returned.

pause:sender

Action method that pauses the Sound. Other than the argument and the return type, this is the sam
Returns self.

(int)play

Initiates playback of the Sound. The method returns immediately while the playback continues as
background. The playback ends when the Sound receives the stop message, or when its data is ex

When playback starts, willPlay: is sent to the Sound's delegate when it stops, didPlay: is sent. An

play:sender

Action method that plays the Sound. Other than the argument and the return type, this is the same
Returns self.

(int)processingError

Returns a constant that represents the last error that was generated. The sound error codes are liste
Constants.º

read:(NXTypedStream *)stream

Reads archived sound data from stream into the Sound. Returns self.

(int)readSoundfile:(const char *)filename

Replaces the Sound's contents with those of the soundfile filename. The Sound loses its current na
code is returned.

(int)record

Initiate recording into the Sound. To record from the CODEC microphone, the Sound's format, sa
channel count must be SND_FORMAT_MULAW_8, SND_RATE_CODEC, and 1, respectively.
set (if the Sound is a newly created object, for example), it defaults to accommodate a CODEC rec
format is SND_FORMAT_DSP_DATA_16, the recording is from the DSP.

The method returns immediately while the recording continues asynchronously in the background.
when the Sound receives the stop message or when the recording has gone on for the duration of th
The default CODEC recording lasts precisely ten minutes if not stopped. To record for a longer tir
size of the sound data with setSoundStruct:soundStructSize: or setDataSize:dataFormat:samplingR
infoSize:.

When the recording begins, willRecord: is sent to the Sound's delegate when the recording stops, c

An error code is returned.

record:sender

Action method that initiates a recording. Other than the argument and return type, this is the same
Returns self.

(int)resume

Resumes the paused Sound's activity. An error code is returned.

resume:sender

Action method that resumes the paused Sound. Returns self.

(int)sampleCount

Returns the number of sample frames, or channel count-independent samples, in the Sound.

(int)samplesProcessed

If the Sound is currently playing or recording, this returns the number of sample frames that have b
so far. Otherwise, the number of sample frames in the Sound is returned. If the sample frame cou
1 is returned.

(double)samplingRate

Allocates new, unfragmented sound data for the Sound, as described by the arguments. The Sound freed. This method is useful for setting a determinate data length prior to a recording or for creatin algorithmic sound creation. An error code is returned.

setDelegate:anObject

Sets the Sound's delegate to anObject. The delegate may implement the following methods:

·willPlay:
·didPlay:
·willRecord:
·didRecord:
·hadError:

Returns self.

setName:(const char *)aName

Sets the Sound's name to aName. If aName is already being used, then the Sound's name isn't set otherwise returns self.

setSoundStruct:(SNDSoundStruct *)aStruct soundStructSize:(int)size

Sets the Sound's sound structure to aStruct. The size in bytes of the new structure, including its so be specified by size. This method can be used to set up a large buffer before recording into an exis the existing soundStruct in the first argument while making size larger than the current size. (The minutes of CODEC sound.) The method is also useful in cases where aStruct already has sound da encapsulated in a Sound object yet. The Sound's status must be NX_SoundInitialized or NX_Sou method to do anything. Returns self.

soundBeingProcessed

Returns the Sound object that's being performed. The default implementation always returns self.

(SNDSoundStruct *)soundStruct

Returns a pointer to the Sound's SNDSoundStruct structure that holds the object's sound data.

(SNDSoundStruct *)soundStructBeingProcessed

Returns a pointer to the SNDSoundStruct structure that's being performed. This may not be the sa returned by the soundStruct methodÐSound object's contain a private sound structure that may be playing. If the Sound isn't currently playing or recording, then this will return the public structure

(int)status

Return the Sound's current status, one of the following integer constants:

·NX_SoundStopped
·NX_SoundRecording
·NX_SoundPlaying
·NX_SoundInitialized
·NX_SoundRecordingPaused
·NX_SoundPlayingPaused
·NX_SoundRecordingPending
·NX_SoundPlayingPending
·NX_SoundFreed

(int)stop

Terminates the Sound's playback or recording. If the Sound was recording, the didRecord: messag
delegate if playing, didPlay: is sent. An error code is returned.

stop:sender

Action method that stops the Sound's playback or recording. Other than the argument and the retu
as the stop method. Returns self.

tellDelegate:(SEL)theMessage

Sends theMessage to the Sound's delegate (only sent if the delegate implements theMessage). You
method directly it's invoked automatically as the result of activities such as recording and playing.
it in designing a subclass of Sound. Returns self.

write:(NXTypedStream *)stream

Archives the Sound by writing its data to stream, which must be open for writing. Returns self.

(int)writeSoundfile:(const char *)filename

Writes the Sound's contents (its SNDSoundStruct and sound data) to the soundfile filename. An e

writeSoundToStream:(NXStream *)stream

Writes the Sound's name (if any), priority, SNDSoundStruct, and sound data (if any) to the NXStr
self.

didPlay:sender

Sent to the delegate when the Sound stops playing.

didRecord:sender

Sent to the delegate when the Sound stops recording.

hadError:sender

Sent to the delegate if an error occurs during recording or playback.

willPlay:sender

Sent to the delegate when the Sound begins to play.

willRecord:sender

Sent to the delegate when the Sound begins to record.