

# Defined Types

## NXSoundDeviceError

**DECLARED IN**      `soundkit/NXSoundDevice.h`

## SYNOPSIS

typedef enum

```

_NXSoundDeviceError {
    NX_SoundDeviceErrorNone,
    NX_SoundDeviceErrorKernel,
    NX_SoundDeviceErrorTimeout,
    NX_SoundDeviceErrorLookUp,
    NX_SoundDeviceErrorHost,
    NX_SoundDeviceErrorNoDevice,
    NX_SoundDeviceErrorNotActive,
    NX_SoundDeviceErrorTag,
    NX_SoundDeviceErrorParameter,
    NX_SoundDeviceErrorMax
} NXSoundDeviceError

```

<b>DESCRIPTION</b>	Error codes returned by Sound Kit methods that access the sound driver.
--------------------	---

## NXSoundParameterTag (Sound Device Keys)

**DECLARED IN**      `soundkit/NXSoundParameterTags.h`

## SYNOPSIS

typedef enum

```

_NXSoundParameterTag {
    NX_SoundDeviceBufferSize,
    NX_SoundDeviceBufferCount,
    NX_SoundDeviceDetectPeaks,
    NX_SoundDeviceRampUp,
    NX_SoundDeviceRampDown,
    NX_SoundDeviceInsertZeros,
    NX_SoundDeviceDeemphasize,
    NX_SoundDeviceMuteSpeaker,
    NX_SoundDeviceMuteHeadphone,
    NX_SoundDeviceMuteLineOut,
    NX_SoundDeviceOutputLoudness,
    NX_SoundDeviceOutputAttenuationStereo,
    NX_SoundDeviceOutputAttenuationLeft,
    NX_SoundDeviceOutputAttenuationRight,
    NX_SoundDeviceAnalogInputSource,
    NX_SoundDeviceMonitorAttenuation,
    NX_SoundDeviceInputGainStereo,
    NX_SoundDeviceInputGainLeft,
    NX_SoundDeviceInputGainRight,

```

... (continued under the next heading)  
} **NXSoundParameterTag**

**DESCRIPTION**      Sound parameter key tags that apply to NXSoundDevice objects.

**NXSoundParameterTag (Sound Device Values)**

**DECLARED IN**      soundkit/NXSoundParameterTags.h

**SYNOPSIS** typedef enum  
\_NXSoundParameterTag {  
    ... (continued from the previous heading)  
    **NX\_SoundDeviceAnalogInputSource\_Microphone,**  
    **NX\_SoundDeviceAnalogInputSource\_LineIn,**  
    ... (continued under the next heading)  
} **NXSoundParameterTag**

**DESCRIPTION**      Sound parameter value tags that apply to NXSoundDevice objects.    These are acceptable values for the NX\_SoundDeviceAnalogInputSource parameter.

**NXSoundParameterTag (Sound Stream Keys)**

**DECLARED IN**      soundkit/NXSoundParameterTags.h

**SYNOPSIS** typedef enum  
\_NXSoundParameterTag {  
    ... (continued from the previous heading)  
    **NX\_SoundStreamDataEncoding,**  
    **NX\_SoundStreamSamplingRate,**  
    **NX\_SoundStreamChannelCount,**  
    **NX\_SoundStreamHighWaterMark,**  
    **NX\_SoundStreamLowWaterMark,**  
    **NX\_SoundStreamSource,**  
    **NX\_SoundStreamSink,**  
    **NX\_SoundStreamDetectPeaks,**  
    **NX\_SoundStreamGainStereo,**  
    **NX\_SoundStreamGainLeft,**  
    **NX\_SoundStreamGainRight,**  
    ... (continued under the next heading)  
} **NXSoundParameterTag**

**DESCRIPTION**      Sound parameter key tags that apply to NXSoundStream objects.

**NXSoundParameterTag (Sound Stream Values)**

**DECLARED IN**      soundkit/NXSoundParameterTags.h

**SYNOPSIS** typedef enum  
\_NXSoundParameterTag {  
    ... (continued from the previous heading)  
    **NX\_SoundStreamDataEncoding\_Linear16,**  
    **NX\_SoundStreamDataEncoding\_Linear8,**

```

    NX_SoundStreamDataEncoding_Mulaw8,
    NX_SoundStreamDataEncoding_Alaw8,
    NX_SoundStreamDataEncoding_AES,
    NX_SoundStreamSource_Analog,
    NX_SoundStreamSource_AES,
    NX_SoundStreamSink_Analog,
    NX_SoundStreamSink_AES
} NXSoundParameterTag

```

<b>DESCRIPTION</b>	Sound parameter value tags that apply to NXSoundStream objects. The first five are values for the NX_SoundStreamDataEncoding parameter; the next two are for the NX_SoundStreamSource parameter; the last two are for the NX_SoundStreamSink parameter.
--------------------	---

## NXSoundStatus

**DECLARED IN**      `soundkit/Sound.h`

```

SYNOPSIS
typedef enum {
    NX_SoundStopped,
    NX_SoundRecording,
    NX_SoundPlaying,
    NX_SoundInitialized,
    NX_SoundRecordingPaused,
    NX_SoundPlayingPaused,
    NX_SoundRecordingPending,
    NX_SoundPlayingPending,
    NX_SoundFreed,
} NXSoundStatus;

```

<b>DESCRIPTION</b>	These represent the activities of a Sound object, as returned by Sound's <b>status</b> method.
--------------------	--

## SNDCompressionSubheader

**DECLARED IN**     `sound/soundstruct.h`

```

SYNOPSIS
    int originalSize
    int method;
    int numDropped;
    int encodeLength;
} SNDCompressionSubheader;

```

**DESCRIPTION** This structure describes the attributes of a compressed sound. It immediately follows the general SNDSoundStruct header. If the sound data isn't compressed, this subheader is absent. The structure's fields are

originalSize	The size of the uncompressed data, in bytes
method	The compression format (see <sup>a</sup> Compression Formats, <sup>o</sup> below)
numDropped	The number of dropped bits, if applicable
encodeLength	The number of samples represented by an encoded block

## SNDError

**DECLARED IN**     sound/sounderror.h

**SYNOPSIS**

typedef enum {

```
SND_ERR_NONE,
SND_ERR_NOT_SOUND,
SND_ERR_BAD_FORMAT,
SND_ERR_BAD_RATE,
SND_ERR_BAD_CHANNEL,
SND_ERR_BAD_SIZE,
SND_ERR_BAD_FILENAME,
SND_ERR_CANNOT_OPEN,
SND_ERR_CANNOT_WRITE,
SND_ERR_CANNOT_READ,
SND_ERR_CANNOT_ALLOC,
SND_ERR_CANNOT_FREE,
SND_ERR_CANNOT_COPY,
SND_ERR_CANNOT_RESERVE,
SND_ERR_NOT_RESERVED,
SND_ERR_CANNOT_RECORD,
SND_ERR_ALREADY_RECORDING,
SND_ERR_NOT_RECORDING,
SND_ERR_CANNOT_PLAY,
SND_ERR_ALREADY_PLAYING,
SND_ERR_NOT_IMPLEMENTED,
SND_ERR_NOT_PLAYING,
SND_ERR_CANNOT_FIND,
SND_ERR_CANNOT_EDIT,
SND_ERR_BAD_SPACE,
SND_ERR_KERNEL,
SND_ERR_BAD_CONFIGURATION,
SND_ERR_CANNOT_CONFIGURE,
SND_ERR_UNDERRUN,
SND_ERR_ABORTED,
SND_ERR_BAD_TAG,
SND_ERR_CANNOT_ACCESS,
SND_ERR_TIMEOUT,
SND_ERR_BUSY,
SND_ERR_CANNOT_ABORT,
SND_ERR_INFO_TOO_BIG,
SND_ERR_UNKNOWN,
} SNDError;
```

**DESCRIPTION**     These are the sound error codes returned by many sound functions. The **SNDSoundError()** function returns a pointer to a string that describes the error given one of these codes as an argument.

## **SNDNotificationFun**

**DECLARED IN**     sound/performsound.h

**SYNOPSIS**

typedef int (\*SNDNotificationFun)

```
(SNDSoundStruct *s,
int tag,
int err);
```

**DESCRIPTION** This is the notification function required as an argument to methods such as **SNDStartPlaying()** and **SNDStartRecording()**.

**SNDSoundStruct**

**DECLARED IN** sound/performsound.h

**SYNOPSIS** typedef struct {  
int **magic**;  
int **dataLocation**;  
int **dataSize**;  
int **dataFormat**;  
int **samplingRate**;  
int **channelCount**;  
char **info**[4];  
} **SNDSoundStruct**;

**DESCRIPTION** This structure defines the header for sound data. It's thoroughly explained in the description of the **SNDAlloc()** function.

**snddriver\_handlers**

**DECLARED IN** sound/sounddriver.h

**SYNOPSIS** typedef struct snddriver\_handlers {  
void \***arg**;  
int **timeout**;  
sndreply\_tagged\_t **started**;  
sndreply\_tagged\_t **completed**;  
sndreply\_tagged\_t **aborted**;  
sndreply\_tagged\_t **paused**;  
sndreply\_tagged\_t **resumed**;  
sndreply\_tagged\_t **overflow**;  
sndreply\_recorded\_data\_t **recorded\_data**;  
sndreply\_dsp\_cond\_true\_t **condition\_true**;  
sndreply\_dsp\_msg\_t **dsp\_message**;  
sndreply\_dsp\_msg\_t **dsp\_error**;  
} **snddriver\_handlers\_t**;

**DESCRIPTION** This structure is required as an argument by the **snddriver\_reply\_handler()** function. It declares, primarily, a series of call-back functions that are used by the sound driver to communicate with your program.

**sndreply\_dsp\_cond\_true\_t**

**DECLARED IN** sound/sounddriver.h

**SYNOPSIS** typedef void  
(\***sndreply\_dsp\_cond\_true\_t**)  
(void \**arg*,  
unsigned int *mask*,

unsigned int *flags*,  
unsigned int *regs*);

**DESCRIPTION**      Function type used by the sound driver's reply handler.

**sndreply\_dsp\_msg\_t**

**DECLARED IN**      sound/sounddriver.h

**SYNOPSIS**

(**sndreply\_dsp\_msg\_t**)

(void \**arg*,  
int *data*,  
int *size*);

typedef void

**DESCRIPTION**      Function type used by the sound driver's reply handler.

**sndreply\_recorded\_data\_t**

**DECLARED IN**      sound/sounddriver.h

**SYNOPSIS**

(**sndreply\_recorded\_data\_t**)

(void \**arg*,  
int *tag*,  
void \**data*,  
int *size*);

typedef void

**DESCRIPTION**      Function type used by the sound driver's reply handler.

**sndreply\_tagged\_t**

**DECLARED IN**      sound/sounddriver.h

**SYNOPSIS**

(void \**arg*,  
int *tag*);

typedef void (**sndreply\_tagged\_t**)

**DESCRIPTION**      Function type used by the sound driver's reply handler.

# Symbolic Constants

**ATC Frame Size**

**DECLARED IN**      sound/soundstruct.h

SYNOPSIS	ATC_FRAME_SIZE
DESCRIPTION	This constant represents the size of a single ATC (Audio Transform Compression) frame.

### Compression Formats

DECLARED IN	sound/soundstruct.h
SYNOPSIS	SND_CFORMAT_BITS_DROPPED SND_CFORMAT_BIT_FAITHFUL SND_CFORMAT_ATC
DESCRIPTION	These constants represent the three types of sound data compression.

### DSP Host Commands

DECLARED IN	sound/sounddriver.h
SYNOPSIS	SNDDRIVER_DSP_HC_HOST_RD SNDDRIVER_DSP_HC_HOST_WD SNDDRIVER_DSP_HC_SYS_CALL
DESCRIPTION	These constants represent the DSP host commands that can be passed as an argument to <code>snddriver_dsp_host_cmd()</code> .

### DSP Protocol Options

DECLARED IN	sound/sounddriver.h
SYNOPSIS	SNDDRIVER_DSP_PROTO_DSPERR SNDDRIVER_DSP_PROTO_C_DMA SNDDRIVER_DSP_PROTO_S_DMA SNDDRIVER_DSP_PROTO_HFABORT SNDDRIVER_DSP_PROTO_DSPMSG SNDDRIVER_DSP_PROTO_RAW
DESCRIPTION	These constants represent the DSP protocols that can be passed as an argument to <code>snddriver_dsp_protocol()</code> .

### Executable File Segment Name

DECLARED IN	soundkit/Sound.h
SYNOPSIS	NX_SOUND_SEGMENT_NAME
DESCRIPTION	This represents the segment of an executable file in which sounds are stored.





SYNOPSIS

NX\_SOUNDDEVICE\_ERROR\_MIN  
NX\_SOUNDDEVICE\_ERROR\_MAX

DESCRIPTION

The minimum and maximum NXSoundDeviceError values.

### Sound Parameter Tag Bases

DECLARED IN

soundkit/NXSoundParameterTags.h

SYNOPSIS

NX\_SoundDeviceParameterKeyBase  
NX\_SoundDeviceParameterValueBase  
NX\_SoundStreamParameterKeyBase  
NX\_SoundStreamParameterValueBase  
NX\_SoundParameterTagMax

DESCRIPTION

Lowest tag values for the four sets of parameter tags. The parameter tag values start at 0; NX\_SoundParmaeterTagMax is the highest parameter tag value reserved by the Sound Kit.

### Sound Stream Control Codes

DECLARED IN

sound/sounddriver.h

SYNOPSIS

SNDDRIVER\_AWAIT\_STREAM  
SNDDRIVER\_ABORT\_STREAM  
SNDDRIVER\_PAUSE\_STREAM  
SNDDRIVER\_RESUME\_STREAM

DESCRIPTION

These constants represent the controlling operations that are specified as an argument to **snddriver\_stream\_control()**.

### Sound Stream Null Time

DECLARED IN

soundkit/NXSoundStream.h

SYNOPSIS

NX\_SOUNDSTREAM\_TIME\_NULL

DESCRIPTION

Provides a **timeval** value (as defined in **sys/time.h**) that indicates the present time.

### Sound Stream Path Codes

DECLARED IN

sound/sounddriver.h

SYNOPSIS

SNDDRIVER\_STREAM\_FROM\_SNDIN  
SNDDRIVER\_STREAM\_TO\_SNDOUT\_22  
SNDDRIVER\_STREAM\_TO\_SNDOUT\_44  
SNDDRIVER\_STREAM\_FROM\_DSP  
SNDDRIVER\_STREAM\_TO\_DSP  
SNDDRIVER\_STREAM\_DSP\_TO\_SNDOUT\_22  
SNDDRIVER\_STREAM\_DSP\_TO\_SNDOUT\_44  
SNDDRIVER\_STREAM\_THROUGH\_DSP\_TO\_SNDOUT\_22

SNDDRIVER\_STREAM\_THROUGH\_DSP\_TO\_SNDOUT\_44  
SNDDRIVER\_DMA\_STREAM\_TO\_DSP  
SNDDRIVER\_DMA\_STREAM\_FROM\_DSP  
SNDDRIVER\_DMA\_STREAM\_THROUGH\_DSP\_TO\_SNDOUT\_22  
SNDDRIVER\_DMA\_STREAM\_THROUGH\_DSP\_TO\_SNDOUT\_44

**DESCRIPTION**      These constants represent the sound stream paths that can be specified as an argument to the **snddriver\_stream\_setup()** function.

**Sound Structure Formats**

**DECLARED IN**      sound/soundstruct.h

**SYNOPSIS**

SND\_FORMAT\_MULAW\_8  
SND\_FORMAT\_LINEAR\_8  
SND\_FORMAT\_LINEAR\_16  
SND\_FORMAT\_LINEAR\_24  
SND\_FORMAT\_LINEAR\_32  
SND\_FORMAT\_FLOAT  
SND\_FORMAT\_DOUBLE  
SND\_FORMAT\_INDIRECT  
SND\_FORMAT\_DSP\_CORE  
SND\_FORMAT\_DSP\_DATA\_8  
SND\_FORMAT\_DSP\_DATA\_16  
SND\_FORMAT\_DSP\_DATA\_24  
SND\_FORMAT\_DSP\_DATA\_32  
SND\_FORMAT\_DISPLAY  
SND\_FORMAT\_MULAW\_SQUELCH  
SND\_FORMAT\_EMPHASIZED  
SND\_FORMAT\_COMPRESSED  
SND\_FORMAT\_COMPRESSED\_EMPHASIZED  
SND\_FORMAT\_DSP\_COMMANDS

SND\_FORMAT\_UNSPECIFIED

**DESCRIPTION**      These constants represent the various sound formats in which sound data can be stored. Note that not all formats are playable without conversion.

**Sound Structure Magic Number**

**DECLARED IN**      sound/soundstruct.h

**SYNOPSIS**

SND\_MAGIC

**DESCRIPTION**      This constant is used to identify a sound structure. It's the value of the **magic** field of all valid SNDSoundStruct structures.

**SoundView Display Modes**

**DECLARED IN**      soundkit/SoundView.h

**SYNOPSIS**

NX\_SOUNDVIEW\_MINMAX  
NX\_SOUNDVIEW\_WAVE

NX\_SOUNDVIEW\_MAX

<b>DESCRIPTION</b>	These constants represent the two display modes offered by the SoundView class. See the SoundView class specification for details.
--------------------	--

# Global Variables

## NXSoundPboardType

**DECLARED IN**      `soundkit/Sound.h`

**SYNOPSIS** `extern NXAtom NXSoundPboardType;`

<b>DESCRIPTION</b>	This is the sound pasteboard type.
--------------------	------------------------------------