## Getting and Setting Sound Input Device Information

You can get information about a specific sound input device and alter that information by calling the functions **SPBGetDeviceInfo** and **SPBSetDeviceInfo**. These functions accept selectors that determine which information you need or want to change. The currently defined selectors are defined by constants of type OSType:

| | |
|---|---|
| siActiveChannels | channels active |
| siActiveLevels | levels active |
| siAGCOnOff | automatic gain control state |
| siAsync | asynchronous capability |
| siNumberChannels | current number of channels |
| siChannelAvailable | number of channels available |
| siCompressionAvailable | compression types available |
| siCompressionFactor | current compression factor |
| siCompressionHeader | return compression header |
| siCompressionType | current compression type |
| siContinuous | continuous recording |
| siDeviceBufferInfo | size of interrupt buffer |
| siDeviceConnected | input device connection status |
| siDeviceIcon | input device icon |
| siLevelMeterOnOff | level meter state |
| siDeviceName | input device name |
| siOptionsDialog | display options dialog box |
| siPlayThruOnOff | play-through state |
| siRecordingQuality | recording quality |
| siSampleRate | current sample rate |
| siSampleRateAvailable | sample rates available |
| siSampleSizeAvailable | sample sizes available |
| siSampleSize | current sample size |
| siTwosComplementOnOff | two's complement state |
| siVoxRecordInfo | VOX record parameters |
| siVoxStopInfo | VOX stop parameters |

The format of the relevant data (either returned by the **Sound Manager** or provided by you) depends on the selector you provide. For example, if you want to determine the name of some sound input device, you can pass the siDeviceName selector and a pointer to a 256-byte buffer to the **SPBGetDeviceInfo** function. If **SPBGetDeviceInfo** can get the information, it fills that buffer with the name of the specified sound input device. The code example below illustrates one way you can determine the name of a particular sound input device.

```
// Determining the name of a sound input device

// Assuming inclusion of MacHeaders
#include <Sound.h>
#include <SoundInput.h>

// Prototype routine like this prior to calling it
OSErr DeviceName (long, Str255);

OSErr  DeviceName (long myInRefNum, Str255 dName)
{
    OSErr myErr;  // Error returned

    // GetDevice Information, storing error return value
    myErr = SPBGetDeviceInfo(myInRefNum, siDeviceName, (Ptr)&dName);
    return myErr;
}
```

Some selectors cause **SPBGetDeviceInfo** to return data of other types. The next code example illustrates how to determine the number of channels, the sample rate, the sample size, and the compression type currently in use by a given sound input device. The procedure defined in the Listing below is called in the procedure defined in the code example listed in the section **Recording Sounds Directly From a Device**

```
// Determining some sound input device settings

// Assuming inclusion of MacHeaders
#include <SoundInput.h>

// Prototype routine like this prior to calling it
void GetDeviceSettings (long, short *, Fixed *, short *, OSType *);

void GetDeviceSettings (long myInRefNum, short *numChannels,
    Fixed *sampleRate, short *sampleSize,
    OSType *compressionType)
{
    OSErr myErr;

    // get number of active channels
    myErr = SPBGetDeviceInfo(myInRefNum, siNumberChannels,
            (Ptr) &numChannels);

    // get sample rate
    myErr = SPBGetDeviceInfo(myInRefNum, siSampleRate, (Ptr)
&sampleRate);

    // get sample size
    myErr = SPBGetDeviceInfo(myInRefNum, siSampleSize,
            (Ptr) &sampleSize);

    // get compression type
    myErr = SPBGetDeviceInfo(myInRefNum, siCompressionType,
            (Ptr) &compressionType);
}
```

Some other selectors return a list of items, which your application must interpret. The following Table lists the available selectors together with the size and meaning of the associated information whose address is passed or returned in the *infoData* parameter. Note that all of the selectors returning a handle will allocate the memory for that handle in the current heap zone; you are responsible for disposing of that handle when you are done with it.

**Sound input device information selectors**

| Selector | Description | | |
|---|---|---|---|
| 'agc ' | Gets or sets the current state of the automatic gain control feature. | | |
| | *infoData* | short | 0 if off, 1 if on |
| 'asyn' | Determines if driver supports asynchronous recording functions. Some sound input drivers might support synchronous recording only. | | |
| | *infoData* | short | 0 if synchronous calls only, 1 otherwise |
| 'chac' | Gets or sets the channels to record from. When setting the active channels, the data passed in is a long integer that is interpreted as a bitmap describing the channels to record from. For example, if bit 0 is set, then the first channel is made active. The samples for each active channel are stored one after another in the application's buffer. When reading the active channels, the data returned is a bitmap of the active channels. | | |
| | *infoData* | long | bitmap of active channels |
| 'chan' | Gets or sets the number of channels this device is to record. | | |
| | *infoData* | short | number of channels |
| 'chav' | Gets the maximum number of channels this device can record. | | |
| | *infoData* | short | number of available channels |
| 'cmav' | Gets the number and list of compression types this device can produce. | | |
| | *infoData* | short | number of compression types supported |
| | | Handle | list of compression types (each is OSType, 4 bytes) |
| 'cmfa' | Gets the number of samples per byte at the current compression setting. | | |
| | *infoData* | short | compression factor |
| 'cmhd' | Gets a compression header for the current recording settings. Your application passes in a pointer to a compressed sound header and the driver fills it in. Before calling **SPBGetDeviceInfo** with this selector, you should set the numFrames field of the compressed sound header to the number of bytes in the sound. When **SPBGetDeviceInfo** returns successfully, that field contains the number | | |
| 'cmhd' | of sample frames in the sound. This selector is needed only by drivers that use compression types that are not directly supported by Apple. If you call this selector after recording a sound, your application can get enough information about the sound to play it or save it in a file. | | |
| | *infoData* | Pointer | pointer to a compressed sound header |
| 'comp' | Gets or sets the compression type. Some devices allow the incoming samples to be compressed before being placed in your application's input buffer. | | |
| | *infoData* | OSType | compression type |
| 'cont' | Gets or sets the state of continuous recording from this device. If continuous recording is being turned on, the driver records samples into an internal buffer between calls to **SPBRecord**. This allows a | | |

subsequent recording to begin where the previous one stopped. If continuous recording is being turned off, the driver stops recording samples to its internal buffer.
*infoData* short state of continuous recording (0 is off, 1 is on)

'dbin' Gets the size of the device's internal buffer. This information can be useful when you want to modify sound input data at interrupt time.
*infoData* long size of device's internal buffer

'dcon' Gets the state of the device connection.
*infoData* short one of the following values:

siDeviceIsConnected //device is connected and ready
siDeviceNotConnected //device is not connected
siDontKnowIfConnected //can not tell if device connected

'icon' Gets the device's icon and icon mask.
*infoData* Handle icon and icon mask

'lmac' Gets the current signal level for each active channel. Each value returned is a short, and the number of values returned depends on the number of active channels. You can determine how many channels are active by calling **SPBGetDeviceInfo** with the 'chan' selector.
*infoData* short level meter setting

'lmet' Gets or sets the current state of the level meter. Once the level meter has been turned on, calling **SPBGetDeviceInfo** returns a value in the level meter setting that ranges from 0 (no volume) to 255 (full volume).
*infoData* short state of level meter (0 is off, 1 is on)
short level meter setting

'name' Gets the name of the sound input device. Your application must pass a pointer to a buffer that will be filled in with the device's name. The buffer needs to be large enough to hold a Str255 data type.
*infoData* Pointer pointer to buffer for name of device

'optd' Gets or sets the **Options** dialog box feature. The **Options** dialog box is designed to allow the user to configure device-specific features of the sound input hardware. Note that no argument should be supplied when you pass this selector to **SPBSetDeviceInfo**.
*infoData* short 1 if device supports options, 0 otherwise

'plth' Gets or sets the current play-through volume.
*infoData* short volume (0 is off, 1-7 otherwise)

'qual' Gets or sets the current quality of recorded sound. Currently three qualities are supported: 'good', 'betr', and 'best'.
*infoData* OSType recording quality

'srat' Gets or sets the sample rate to be produced by this device. The sample rate must be in the range 0 to 65535.99998 Hz. Note that the sample rate is declared as a Fixed data type, but the most significant bit is not treated as a sign bit; instead, that bit is interpreted as having the value 32,768.
*infoData* Fixed sample rate

'srav' Gets the range of sample rates this device can produce. The first 2 bytes of the information returned specify how many different sample rates the device supports. If that number is 0, then the next two sample rates define a continuous range of sample rates. Otherwise, a list is returned that contains the sample rates supported. Each sample rate is of type Fixed and occupies 4 bytes. Note that the sample rates are declared as Fixed data types, but their most significant bit is not treated as a sign bit; instead, that bit is interpreted as having the value 32,768.

| | | |
|---|---|---|
| *infoData* | short | number of sample rates |
| | Handle | list of sample rates |

'ssav' Gets the range of sample sizes supported by this device. The first 2 bytes of the information returned specify how many different sample sizes the device supports. A list is returned that contains the sample sizes supported. Each sample size is a short and occupies 2 bytes.

| | | |
|---|---|---|
| *infoData* | short | number of sample sizes |
| | Handle | list of sample sizes |

'ssiz' Gets or sets the sample size to be produced by this device. Because some compression formats require specific sample sizes, this selector may return an error when compression is used.

| | | |
|---|---|---|
| *infoData* | short | sample size |

'twos' Gets or sets the current state of the two's complement feature.

| | | |
|---|---|---|
| *infoData* | short | 1 if two's complement output desired, 0 otherwise |

'voxr' Gets or sets the current VOX record parameters. The first 2 bytes of the *infoData* parameter indicate whether VOX recording is on or off. The next 2 bytes contain the VOX record trigger value. Trigger values range from 0 to 255 (0 is trigger immediately, 255 is trigger only on full volume).

| | | |
|---|---|---|
| *infoData* | short | 0 if off, 1 if on |
| | short | record trigger value |

'voxs' Gets or sets the current VOX stop parameters. The first 2 bytes of the *infoData* parameter indicate whether VOX stopping is on or off. The next 2 bytes contain the VOX stop trigger value. Trigger values range from 0 to 255 (255 is stop immediately, 0 is stop only on total silence). The final 2 bytes indicate how many milliseconds the trigger value must be continuously valid for recording to be stopped. Delay values range from 0 to 65,535.

| | | |
|---|---|---|
| *infoData* | short | 0 if off, 1 if on |
| | short | record trigger value |
| | short | delay value |