Recording Sounds Through the Sound Input Dialog Box

You can record sounds from the current input device by using the **SndRecord** or **SndRecordToFile** function. You can use the **SndRecord** function to present a standard user interface for recording sounds. When calling **SndRecord**, you need to provide a handle to a block of memory where the incoming data should be stored. If you pass the address of a NULL handle, however, the **Sound Manager** allocates a large block of space and resizes it when the recording stops. The Listing below illustrates how to call **SndRecord**.

```
//Listing: Recording through the sound input dialog box


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

// Prototype routine like this prior to calling it
void RecordThruDialog(Handle *);

void  RecordThruDialog (Handle *mySndH)
{
    OSErr myErr;
    Point myCorner;

    // ToolBox should be initialized prior to calling this routine
    //  or dialog may not work properly

    // Set upper left corner for sound input dialog
    SetPt(&myCorner, 50, 50);

    // Initialize handle, and start recording.
    // SndRecord will return a handle to the recorded sound
    *mySndH = nil;
    myErr = SndRecord(nil, myCorner, siBetterQuality, mySndH);
    // handle any error that may occur here
}
```

If you pass a sound handle that is not NULL, the time of recording is derived from the amount of space reserved by that handle. The handle is resized on completion of the recording.

The first parameter in the call to **SndRecord** is the address of a filter procedure that determines how user actions in the dialog box are filtered. In the Listing above, no filter procedure is desired, so the parameter is specified as NULL. The third parameter specifies the quality of the recording. Currently three values are supported:

```
siBestQuality       //the best quality available
siBetterQuality     //a quality better than good
siGoodQuality       //a good quality
```

The precise meanings of these constants are driver-specific. The constant siBestQuality indicates that you want the highest quality recorded sound, usually at the expense of increased storage space (probably because no

compression is performed on the sound data). The constant siGoodQuality indicates that you are willing to sacrifice audio quality if necessary to minimize the amount of storage space required (typically this means that MACE 6:1 compression is performed on the sound data). For most voice recording, you should specify siGoodQuality. The constant siBetterQuality defines a quality and storage space combination that is between those provided by the other two constants.

After the procedure in the Listing above has executed successfully, you could play the recorded sound by calling **SndPlay** and passing it mySndH. Note that mySndH is a handle to some data in memory that has the structure of an 'snd ' resource, not a handle to an existing resource. To save the recorded data as a resource, you can call **AddResource**.

To record a sound directly into a file, you can call **SndRecordToFile**. The **SndRecordToFile** function works exactly like **SndRecord**, except that you must pass it the file reference number of an open file instead of a handle to some memory. When **SndRecordToFile** exits successfully, that file contains the recorded audio data in AIFF or AIFF-C format. You can then play the recorded sound by passing that file reference number to the **SndStartFilePlay** function.