Sound Kit

Issues

Q: Why doesn't SoundView's **setReductionFactor:** work in Release 3?

Q: How can I determine a Sound's duration?

Q: How can I get Sounds to repeat one after the other, or the same one multiple times?

Q: I am porting my sound driver program from Release 2 to 3. But the kernel header file **snd_msgs.h** is no longer available in Release 3. What can I do to get a handle on the error codes returned by the SNDDRIVER functions?

Q: Why doesn't SoundView's *setReductionFactor:* work in Release 3?

A: There is a known bug in the SoundView's **setSound:** method which resets the reductionFactor even if you are explicitly setting it yourself. If you wish to use your own reductionFactor, you need to call **setReductionFactor:** after each call to **setSound:**.

The SoundEditor example released in 3.0 uses **setReductionFactor:**, but it doesn't work correctly because of this bug.

A point of confusion is that the reduction methodsĐsetReduction and reductionĐhave been removed from the API for 3.0, but the reductionFactor methodsĐsetReductionFactor and setReductionĐare still part of the API.

QA846

Valid for 3.0, 3.1

Q: How can I determine a Sound's duration?

A: Send the Sound the sampleCount and samplingRate methods (if necessary) and divide the former number by the latter.

Q: How can I get Sounds to repeat one after the other, or the same one multiple times?

A: There are several approaches; however, none are guaranteed to work in all possible situations.

1) You can compute the duration of each sound just before you play it (following the approach in the answer to the first question). Set up a timed entry (see documentation on DPSAddTimedEntry) that occurs at a time equal to the sound's duration plus any desired delay between playings. When you get this timed entry, you tell the next Sound to play, and so forth.

2) If you don't require an Application event loop, you can reliably chain sounds together using the Sound library C functions. See the example file /NextDeveloper/Examples/Sound/chaintest.c, and the README file in the same directory. You are advised to use the SNDNotificationFun approach, as in this example, and to avoid the SNDWait() function.

3) If you require an Application with an event loop (the normal case), you might have success using the SoundKit delegation method didPlay:. Create an object that is the Sound's delegate, or use an existing object. Provide it with these methods:

```
int sndCtr;

- init {
      [mySound setDelegate:self]; /* mySound is the Sound to be looped */
      sndCtr = 1;
}
```

```
- playIt {
       [mySound play];
}

- didPlay:sender {
       if (sndCtr++ < NUMTIMES)
        [self playIt]; /* play it NUMTIMES in a row.  Alternately, use
some other terminating condition, or trigger other Sounds           here */
}
```

This approach sometimes causes problems in the playback, however, particularly if the user interrupts the playing with an event such as a mouse-down. You'll have to try it for your specific case and see whether it works.

4) If none of these approaches is suitable, the workaround is to create a new Sound composed of multiple copies of the desired sound(s), using the SoundKit's copy/paste functionality. You can do this splicing programmatically with the insertSamples:at: method, or manually by using the SoundEditor example.

5) As a final option, you can combine the chaintest strategy with the SoundKit example above. Override Sound's "play" method and directly enqueue the soundstruct with SNDStartPlaying, giving it your own special SNDNotificationFun to do the chaining as in chaintest. The drawback of this approach is that you cannot

safely send the delegate a soundDidPlay: message when the repeat count runs out.

QA267

Valid for 1.0, 2.0, 3.0

Q: I am porting my sound driver program from Release 2 to 3. But the kernel header file snd_msgs.h is no longer available in Release 3. What can I do to get a handle on the error codes returned by the SNDDRIVER functions?

A: You can include the following code in your program to provide for error handling:

```
#import <sound/sounderror.h>    /* for SNDSoundError() */

#ifndef SNDDRIVER_NO_ERROR
#define SNDDRIVER_NO_ERROR      100     // non-error ack.
#define SNDDRIVER_BAD_PORT      101     // message sent to wrong port
#define SNDDRIVER_BAD_MSG       102     // unknown message id
#define SNDDRIVER_BAD_PARM      103     // bad parameter list in message
#define SNDDRIVER_NO_MEMORY     104     // can't allocate memory (record)
#define SNDDRIVER_PORT_BUSY     105     // access req'd to existing excl access port
#define SNDDRIVER_NOT_OWNER     106     // must be owner to do this
#define SNDDRIVER_BAD_CHAN      107     // dsp channel hasn't been inited
#define SNDDRIVER_SEARCH        108     // couldn't find requested resource
```

```
#define SNDDRIVER_NODATA        109     // can't send data commands to dsp in this mode
#define SNDDRIVER_NOPAGER       110     // can't allocate from external pager (record).
#define SNDDRIVER_NOTALIGNED    111     // bad data alignment.
#define SNDDRIVER_BAD_HOST_PRIV 112     // bad host privilege port passed.
#define SNDDRIVER_BAD_PROTO     113     // can't do requested operation in cur protocol
#define SNDDRIVER_MAX_ERROR     113
#endif

static char *snddriver_error_list[] = {
        "sound success",
        "sound message sent to wrong port",
        "unknown sound message id",
        "bad parameter list in sound message",
        "can't allocate memory for recording",
        "sound service in use",
        "sound service requires ownership",
        "DSP channel not initialized",
        "can't find requested sound resource",
        "bad DSP mode for sending data commands",
        "external pager support not implemented",
        "sound data not properly aligned",
        "bad host provilege port passed",
        "can't do requested operation in extant DSP protocol"
};

/*
```

```
 * Error code to string conversion
 */
extern char *mach_error_string();
char *snddriver_error_string(int error)
{
    if (error <= 0)
      return mach_error_string(error);
    else if (error >= SNDDRIVER_NO_ERROR && error <= SNDDRIVER_MAX_ERROR)
      return snddriver_error_list[error-SNDDRIVER_NO_ERROR];
    else
      return SNDSoundError(error);
}

void snddriver_error(char *msg,int error)
{
    char *errtype;
    if (error <= 0)
      errtype = "mach";
    else if (error >= SNDDRIVER_NO_ERROR && error <= SNDDRIVER_MAX_ERROR)
      errtype = "snddriver";
    else
      errtype = "sound";
    fprintf(stderr,"%s\n\t%s error:%s\n",msg,errtype,snddriver_error_string(error));
}
```

QA855

Valid for 3.0, 3.1