Mac OS Sound

Including Sound Manager 3.3

Apple Computer, Inc.
Technical Publications
April, 1998

Figure 2-4 illustrates the fields of a format 2 `'snd '` resource. The reference count field is for your application's use and is not used by the Sound Manager. The number of sound commands field and the sound command fields are the same as described in a format 1 resource. The last field of this resource contains the sampled sound. The first command should be either a `soundCmd` command or `bufferCmd` command with the data offset bit set in the command to specify the location of this sampled sound header.

Listing 2-41 shows a resource specification that illustrates the structure of a format 2 `'snd '` resource.

Listing 2-41 A format 2 `'snd '` resource

```
data 'snd ' (9003, "Pig Squeal", purgeable) {
   /*the sound resource header*/
   $"0002"        /*format type*/
   $"0000"        /*reference count for application's use*/
   /*the sound command*/
   $"0001"        /*number of sound commands that follow (1)*/
   $"8051"        /*command 1--bufferCmd*/
   $"0000"        /*param1 = 0*/
   $"0000000E"    /*param2 = offset to sound header (14 bytes)*/
   /*the sampled sound header*/
   $"00000000"    /*pointer to data (it follows immediately)*/
   $"00000BB8"    /*number of bytes in sample (3000 bytes)*/
   $"56EE8BA3"    /*sampling rate of this sound (22 kHz)*/
   $"000007D0"    /*starting of the sample's loop point*/
   $"00000898"    /*ending of the sample's loop point*/
   $"00"          /*standard sample encoding*/
   $"3C"          /*baseFrequency at which sample was taken*/
   $"80 80 81 82 84 87 93 84"    /*the sampled-sound data*/
   $"6F 68 6D 65 72 7B 82 88"
   $"91 8E 8D 8F 86 7E 7C 79"
   $"6F 6D 71 70 70 79 7F 81"
   $"89 8F 8D 8B" /*rest of data omitted in this example*/
};
```

Note
Remember that format 2 `'snd '` resources are obsolete. You should create only format 1 `'snd '` resources. ◆

Sound Files

This section describes in detail the structure of AIFF and AIFF-C files. Both of these types of sound files are collections of **chunks** that define characteristics of the sampled sound or other relevant data about the sound.

Note
Most applications only need to read AIFF and AIFF-C files or to record sampled-sound data directly to them. You can both play and record AIFF and AIFF-C files without knowing the details of the AIFF and AIFF-C file formats, as explained in the chapter “Introduction to Sound on the Macintosh” in this book. Thus, the information in this section is for advanced programmers only. ◆

Currently, the AIFF and AIFF-C specifications include the following chunk types.

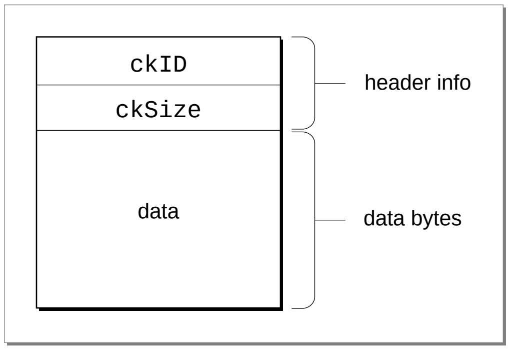
Chunk type	Description
Form Chunk	Contains information about the format of an AIFF or AIFF-C file and contains all the other chunks of such a file.
Format Version Chunk	Contains an indication of the version of the AIFF-C specification according to which this file is structured (AIFF-C only).
Common Chunk	Contains information about the sampled sound such as the sampling rate and sample size.
Sound Data Chunk	Contains the sample frames that comprise the sampled sound.
Marker Chunk	Contains markers that point to positions in the sound data.
Comments Chunk	Contains comments about markers in the file.
Sound Accelerator Chunk	Contains information intended to allow applications to accelerate the decompression of compressed audio data.
Instrument Chunk	Defines basic parameters that an instrument (such as a sampling keyboard) can use to play back the sound data.
MIDI Data Chunk	Contains MIDI data.
Audio Recording Chunk	Contains information pertaining to audio recording devices.
Application Specific Chunk	Contains application-specific information.
Name Chunk	Contains the name of the sampled sound.
Author Chunk	Contains one or more names of the authors (or creators) of the sampled sound.
Copyright Chunk	Contains a copyright notice for the sampled sound.
Annotation Chunk	Contains a comment.

The following sections document the four principal kinds of chunks that can occur in AIFF and AIFF-C files.

Chunk Organization and Data Types

An AIFF or AIFF-C file contains several different types of chunks. For example, there is a Common Chunk that specifies important parameters of the sampled sound, such as its size and sample rate. There is also a Sound Data Chunk that contains the actual audio samples. A chunk consists of some header information followed by some data. The header information consists of a chunk ID number and a number that indicates the size of the chunk data. In general, therefore, a chunk has the structure shown in Figure 2-6.

Figure 2-6 The general structure of a chunk



The header information of a chunk has this structure:

```
TYPE ChunkHeader =
RECORD
   ckID:    ID;         {chunk type ID}
   ckSize:  LongInt;    {number of bytes of data}
END;
```

The `ckID` field specifies the chunk type. An ID is a 32-bit concatenation of any four printable ASCII characters in the range `' '` (space character, ASCII value $20) through `'~'` (ASCII value $7E). Spaces cannot precede printing characters, but trailing spaces are allowed. Control characters are not allowed. You can specify values for the four types of chunks described later by using these constants:

```
CONST
   FormID            = 'FORM';   {ID for Form Chunk}
   FormatVersionID   = 'FVER';   {ID for Format Version Chunk}
   CommonID          = 'COMM';   {ID for Common Chunk}
   SoundDataID       = 'SSND';   {ID for Sound Data Chunk}
```

The `ckSize` field specifies the size of the data portion of a chunk and does not include the length of the chunk header information.

The Form Chunk

The chunks that define the characteristics of a sampled sound and that contain the actual sound data are grouped together into a container chunk, known as the Form Chunk. The Form Chunk defines the type and size of the file and holds all remaining chunks in the file. The chunk ID for this container chunk is `'FORM'`.

A chunk of type 'FORM' has this structure:

```
TYPE ContainerChunk =
RECORD
   ckID:       ID;         {'FORM'}
   ckSize:     LongInt;    {number of bytes of data}
   formType:   ID;         {type of file}
END;
```

For a Form Chunk, the `ckSize` field contains the size of the data portion of this chunk. Note that the data portion of a Form Chunk is divided into two parts, `formType` and the rest of the chunks of the file, which follow the `formType` field. These chunks are called *local chunks* because their chunk IDs are local to the Form Chunk.

The local chunks can occur in any order in a sound file. As a result, your application should be designed to get a local chunk, identify it, and then process it without making any assumptions about what kind of chunk it is based on its order in the Form Chunk.

The `formType` field of the Form Chunk specifies the format of the file. For AIFF files, `formType` is 'AIFF'. For AIFF-C files, `formType` is 'AIFC'. Note that this type might not be the same as the operating-system type with which the File Manager identifies the file. In particular, a file of operating-system type 'AIFC' might be formatted as an AIFF file.

The Format Version Chunk

One difference between the AIFF and AIFF-C file formats is that files of type AIFF-C contain a Format Version Chunk and files of type AIFF do not. The Format Version Chunk contains a `timestamp` field that indicates when the format version of this AIFF-C file was defined. This in turn indicates what format rules this file conforms to and allows you to ensure that your application can handle a particular AIFF-C file. Every AIFF-C file must contain one and only one Format Version Chunk.

In AIFF-C files, a Format Version Chunk has this structure:

```
TYPE FormatVersionChunk =
RECORD
   ckID:       ID;         {'FVER'}
   ckSize:     LongInt;    {4}
   timestamp:  LongInt;    {date of format version}
END;
```

Note
In AIFF files, there is no Format Version Chunk. ◆

The `timestamp` field indicates when the format version for this kind of file was created. The value indicates the number of seconds since January 1, 1904, following the normal time conventions used by the Macintosh Operating System. (See the chapter on date and time utilities in *Inside Macintosh: Operating System Utilities* for several routines that allow you to manipulate time stamps.)

You should not confuse the format version time stamp with the creation date of the file. The format version time stamp indicates the time of creation of the version of the format according to which this file is structured. Because Apple defines the formats of AIFF-C files, only Apple can change this value. The current version is defined by a constant:

```
CONST
   AIFCVersion1   = $A2805140;   {May 23, 1990, 2:40 p.m.}
```

The Common Chunk

Every AIFF and AIFF-C file must contain a Common Chunk that defines some fundamental characteristics of the sampled sound contained in the file. Note that the format of the Common Chunk is different for AIFF and AIFF-C files. As a result, you need to determine the type of file format (by inspecting the `formType` field of the Form Chunk) before reading the Common Chunk.

For AIFF files, the Common Chunk has this structure:

```
TYPE CommonChunk =
RECORD
   ckID:             ID;         {'COMM'}
   ckSize:           LongInt;    {size of chunk data}
   numChannels:      Integer;    {number of channels}
   numSampleFrames:  LongInt;    {number of sample frames}
   sampleSize:       Integer;    {number of bits per sample}
   sampleRate:       Extended;   {number of frames per second}
END;
```

For AIFF-C files, the Common Chunk has this structure:

```
TYPE ExtCommonChunk =
RECORD
   ckID:             ID;         {'COMM'}
   ckSize:           LongInt;    {size of chunk data}
   numChannels:      Integer;    {number of channels}
   numSampleFrames:  LongInt;    {number of sample frames}
   sampleSize:       Integer;    {number of bits per sample}
   sampleRate:       Extended;   {number of frames per second}
   compressionType:  ID;         {compression type ID}
   compressionName:  PACKED ARRAY[0..0] OF Byte;
                                 {compression type name}
END;
```

The fields that exist in both types of Common Chunk have the following meanings:

The numChannels field of both types of Common Chunk indicate the number of audio channels contained in the sampled sound. A value of 1 indicates monophonic sound, a value of 2 indicates stereo sound, a value of 4 indicates four-channel sound, and so forth. Any number of audio channels may be specified. The actual sound data is stored elsewhere, in the Sound Data Chunk.

The numSampleFrames field indicates the number of sample frames in the Sound Data Chunk. Note that this field contains the number of sample frames, not the number of bytes of data and not the number of sample points. For noncompressed sound data, the total number of sample points in the file is numChannels * numSampleFrames. (For more information on sample points, see "Sampled-Sound Data" on page 64.)

The sampleSize field indicates the number of bits in each sample point of noncompressed sound. Although the field can contain any integer from 1 to 32, the Sound Manager currently supports only 8- and 16-bit sound. For compressed sound data, this field indicates the number of bits per sample in the original sound data, before compression.

The sampleRate field contains the sample rate at which the sound is to be played back, in sample frames per second. For a list of common sample rates, see Table 2-1 on page 71.

An AIFF-C Common Chunk includes two fields that describe the type of compression (if any) used on the audio data. The compressionType field contains the type of the compression algorithm, if any, used on the sound data. Here are the currently available compression types and their associated compression names:

```
CONST
   {compression types}
   NoneType                = 'NONE';
   ACE2Type                = 'ACE2';
   ACE8Type                = 'ACE8';
   MACE3Type               = 'MAC3';
   MACE6Type               = 'MAC6';
```

You can define your own compression types, but you should register them with Apple.

Finally, the compressionName field contains a human-readable name for the compression algorithm ID specified in the compressionType field. Compression names for Apple-supplied codecs are defined by constants:

```
CONST
   {compression names}
   NoneName                = 'not compressed';
   ACE2to1Name             = 'ACE 2-to-1';
   ACE8to3Name             = 'ACE 8-to-3';
   MACE3to1Name            = 'MACE 3-to-1';
   MACE6to1Name            = 'MACE 6-to-1';
```

This string is useful when putting up alert boxes (perhaps because a necessary decompression routine is missing). Pad the end of this array with a byte having the value

0 if the length of this array is not an even number (but do not include the pad byte in the count).

The Sound Data Chunk

The Sound Data Chunk contains the actual sample frames that make up the sampled sound. The Sound Data Chunk has this structure:

```
TYPE SoundDataChunk =
RECORD
   ckID:       ID;         {’SSND’}
   ckSize:     LongInt;    {size of chunk data}
   offset:     LongInt;    {offset to sound data}
   blockSize:  LongInt;    {size of alignment blocks}
END;
```

The `offset` field indicates an offset (in bytes) to the beginning of the first sample frame in the chunk data. Most applications do not need to use the `offset` field and should set it to 0.

The `blockSize` field contains the size (in bytes) of the blocks to which the sound data is aligned. This field is used in conjunction with the `offset` field for aligning sound data to blocks. As with the `offset` field, most applications do not need to use the `blockSize` field and should set it to 0.

The sampled-sound data follows the `blockSize` field. For information on the format of sampled-sound data, see “Sampled-Sound Data” on page 64.

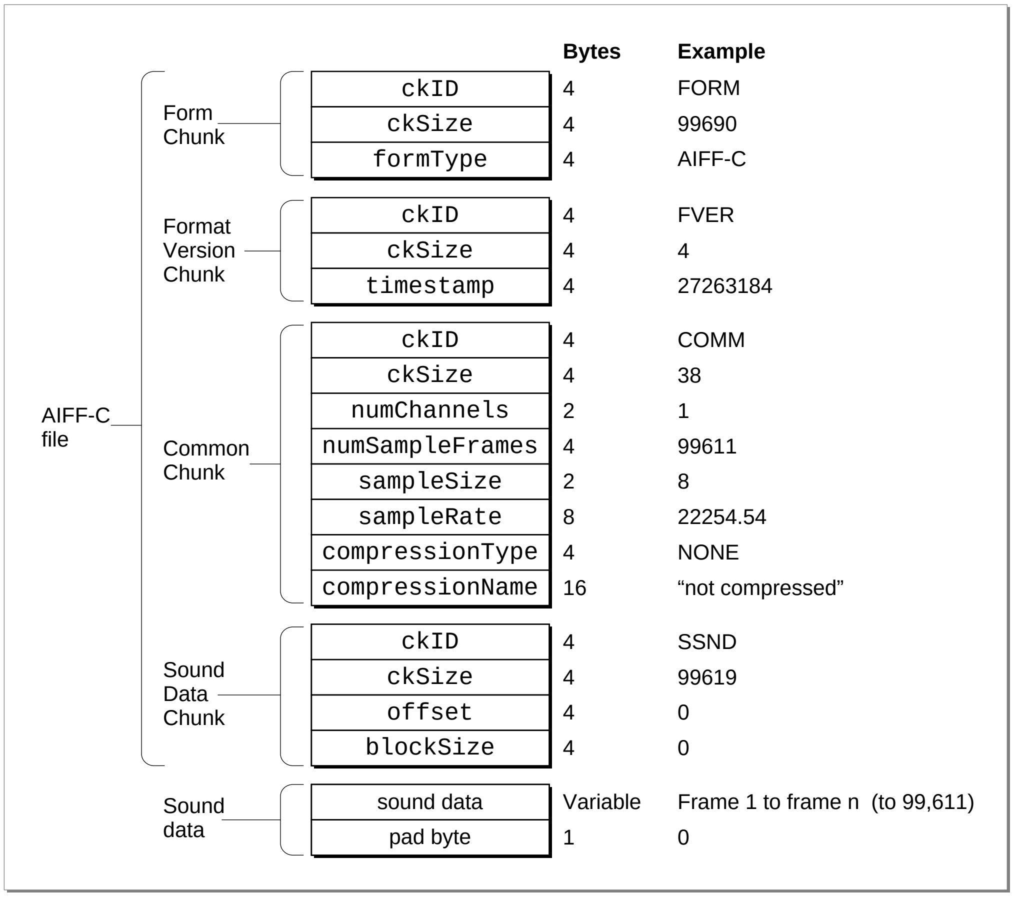
Note
The Sound Data Chunk is required unless the `numSampleFrames` field in the Common Chunk is 0. A maximum of one Sound Data Chunk can appear in an AIFF or AIFF-C file. ◆

Format of Entire Sound Files

Figure 2-7 illustrates an AIFF-C file that contains approximately 4.476 seconds of 8-bit monophonic sound data sampled at 22 kHz. The sound data is not compressed. Note that the number of sample frames in this example is odd, forcing a pad byte to be inserted after the sound data. This pad byte is not reflected in the `ckSize` field of the Sound Data Chunk, which means that special processing is required to correctly determine the actual chunk size.

On a Macintosh computer, the Form Chunk (and hence all the other chunks in an AIFF or AIFF-C file) is stored in the data fork of the file. The file type of an AIFF format file is `’AIFF’`, and the file type of an AIFF-C format file is `’AIFC’`. Macintosh applications should not store any information in the resource fork of an AIFF or AIFF-C file because that information might not be preserved by other applications that edit sound files.

Figure 2-7 A sample AIFF-C file



	Bytes	Example
`ckID`	4	FORM
`ckSize`	4	99690
`formType`	4	AIFF-C
`ckID`	4	FVER
`ckSize`	4	4
`timestamp`	4	27263184
`ckID`	4	COMM
`ckSize`	4	38
`numChannels`	2	1
`numSampleFrames`	4	99611
`sampleSize`	2	8
`sampleRate`	8	22254.54
`compressionType`	4	NONE
`compressionName`	16	“not compressed”
`ckID`	4	SSND
`ckSize`	4	99619
`offset`	4	0
`blockSize`	4	0
sound data	Variable	Frame 1 to frame n (to 99,611)
pad byte	1	0

Every Form Chunk must contain a Common Chunk, and every AIFF-C file must contain a Format Version Chunk. In addition, if the sampled sound has a length greater than 0, there must be a Sound Data Chunk in the Form Chunk. All other chunk types are optional. Your application should be able to read all the required chunks if it uses AIFF or AIFF-C files, but it can choose to ignore any of the optional chunks.

When reading AIFF or AIFF-C files, you should keep the following points in mind:

- Remember that the local chunks in an AIFF or AIFF-C file can occur in any order. An application that reads these types of files should be designed to get a chunk, identify it, and then process it without making any assumptions about what kind of chunk it is based on its order.
- If your application allows modification of a chunk, then it must also update other chunks that might be based on the modified chunk. However, if there are chunks in the file that your application does not recognize, you must discard those unrecognized chunks. Of course, if your application is simply copying the AIFF or AIFF-C file without any modification, you should copy the unrecognized chunks, too.

- You can get the clearest indication of the number of sample frames contained in an AIFF or AIFF-C file from the `numSampleFrames` parameter in the Common Chunk, not from the `ckSize` parameter in the Sound Data Chunk. The `ckSize` parameter is padded to include the fields that follow it, but it does not include the byte with a value of 0 at the end if the total number of sound data bytes is odd.
- Remember that each chunk must contain an even number of bytes. Chunks whose total contents would yield an odd number of bytes must have a pad byte with a value of 0 added at the end of the chunk. This pad byte is not included in the `ckSize` field.
- Remember that the `ckSize` field of any chunk does not include the first 8 bytes of the chunk (which specify the chunk type).

Sound Manager Reference

This section describes the constants, data structures, and routines provided by the Sound Manager. It also describes the format of data stored in sound resources and files that the Sound Manager can play.

The section "Constants" describes the constants defined by the Sound Manager that you can use to specify channel initialization parameters and sound commands. It also lists the sound attributes selector for the `Gestalt` function and the returned bit numbers.

The section "Data Structures" beginning on page 154 describes the Pascal data structures for all of the Sound Manager records that applications can use, including sound commands, sound channels, and sound headers.

The section "Sound Manager Routines" beginning on page 174 describes the routines that allow you to play sounds, manage sound channels, and obtain sound-related information. That section also includes information on routines that give you low-level control over sound output.

The section "Application-Defined Routines" beginning on page 206 describes callback procedures and completion routines that your application might need to define.

The section "Resources" beginning on page 209 describes the organization of format 1 and format 2 `'snd '` resources.

Constants

This section describes the constants that you can use to specify channel initialization parameters, sound commands, and chunk IDs. It also lists the `Gestalt` function sound attributes selector and the returned bit numbers. All other constants defined by the Sound Manager are described at the appropriate location in this chapter. (For example, the constants that you can use to specify sound data types are described in connection with the `SndNewChannel` function beginning on page 182.)