libaudiodec Reference

PlayStation®Vita Programmer Tool Runtime Library Release 3.50

Table of Contents

Datatypes

SceAudiodecInitParam

Union for libaudiodec initialization

Definition

```
#include <audiodec.h>
typedef union SceAudiodecInitParam {
        SceUInt32 size;
        SceAudiodecInitChParam at9;
        SceAudiodecInitStreamParam mp3;
        SceAudiodecInitStreamParam aac;
        SceAudiodecInitStreamParam celp;
} SceAudiodecInitParam;
```

Members

size	Size of the structure corresponding to the type of audio decoder to be used
at9	ATRAC9™ initialization structure
mp3	MP3 initialization structure
aac	AAC initialization structure
celp	CELP initialization structure

Description

This is the union for libaudiodec initialization.

This union is used to initialize libaudiodec using `sceAudiodecInitLibrary()`.

To *size*, do not specify `sizeof(SceAudiodecInitParam)`. Instead, specify the size of the structure corresponding to the type of audio decoder to be used.

See Also

`SceAudiodecInitChParam`, `SceAudiodecInitStreamParam`, `sceAudiodecInitLibrary()`

SceAudiodecInitChParam

Structure for libaudiodec channel initialization

Definition

```
#include <audiodec.h>
typedef struct SceAudiodecInitChParam {
        SceUInt32 size;
        SceUInt32 totalCh;
} SceAudiodecInitChParam;
```

Members

size	Size of the structure
totalCh	Total number of channels available for libaudiodec

Description

This is the structure for libaudiodec channel initialization.

This structure is used to initialize the libaudiodec ATRAC9™ decoder.

For example, to decode 1 monaural channel and 2 stereo channels, 1 x 1 channel + 2 x 2 channels = 5, therefore specify 5 to `totalCh`.

Note that `totalCh` has an upper limit. `totalCh` should not be set higher than `SCE_AUDIODEC_AT9_MAX_CH_IN_LIBRARY`, the maximum value for the total number of channels available for libaudiodec.

See Also

`SceAudiodecInitParam`, `sceAudiodecInitLibrary()`, Maximum Value for the Total Number of Channels Available for libaudiodec

SceAudiodecInitStreamParam

Structure for libaudiodec stream initialization

Definition

```
#include <audiodec.h>
typedef struct SceAudiodecInitStreamParam {
        SceUInt32 size;
        SceUInt32 totalStreams;
} SceAudiodecInitStreamParam;
```

Members

size	Size of the structure
totalStreams	Number of streams available for libaudiodec

Description

This is the structure for libaudiodec stream initialization.

This structure is used to initialize libaudiodec MP3/AAC/CELP decoders.

For example, to decode 1 monaural stream and 2 stereo streams, 1 stream + 2 streams = 3 streams, therefore specify 3 to *totalStreams*.

Note that *totalStreams* has an upper limit. *totalStreams* should not be set higher than `SCE_AUDIODEC_{MP3,AAC,CELP}_MAX_STREAMS`, the maximum value for the total number of streams available.

See Also

`SceAudiodecInitParam`, `sceAudiodecInitLibrary()`,
Maximum Value for the Number of Streams Available for libaudiodec

SceAudiodecCtrl

Audio decoder control structure

Definition

```
#include <audiodec.h>
typedef struct SceAudiodecCtrl {
        SceUInt32 size;
        SceInt32 handle;
        SceUInt8 *pEs;
        SceUInt32 inputEsSize;
        SceUInt32 maxEsSize;
        void *pPcm;
        SceUInt32 outputPcmSize;
        SceUInt32 maxPcmSize;
        SceUInt32 wordLength;
        SceAudiodecInfo *pInfo;
} SceAudiodecCtrl;
```

Members

size	Size of the structure
handle	Decoder handle
pEs	Pointer to elementary stream buffer
inputEsSize	Size of elementary stream used (in Bytes)
maxEsSize	Maximum size of elementary stream being used (in Bytes)
pPcm	Pointer to PCM buffer
outputPcmSize	Size of output PCM (in Bytes)
maxPcmSize	Maximum size of PCM to be output (in Bytes)
wordLength	Number of PCM quantization bits
pInfo	Pointer to audio decoder information structure

Description

This structure is used to control audio decoders.

By calling `sceAudiodecCreateDecoder()` using this structure, the decoder handle will be set and the structure and audio decoder will be associated. Thereafter, associated audio decoders can be used by calling various functions through this structure. At the end, release the association between this structure and audio decoders by calling `sceAudiodecDeleteDecoder()` through this structure.

Refer to each function regarding parameters that need to be set when calling these functions.

See Also

`SceAudiodecInfo`, `sceAudiodecCreateDecoder()`, `sceAudiodecDeleteDecoder()`, `sceAudiodecDecode()`, Number of PCM Quantization Bits

SceAudiodecInfo

Audio decoder information union

Definition

```
#include <audiodec.h>
typedef union SceAudiodecInfo {
        SceUInt32 size;
        SceAudiodecInfoAt9 at9;
        SceAudiodecInfoMp3 mp3;
        SceAudiodecInfoAac aac;
        SceAudiodecInfoCelp celp;
} SceAudiodecInfo;
```

Members

size	Size of the structure corresponding to the type of audio decoder to be used
at9	ATRAC9™ information structure
mp3	MP3 information structure
aac	AAC information structure
celp	CELP information structure

Description

This union is used to set and obtain audio decoder information.

Refer to each function regarding parameters that need to be set when calling these functions.

To *size*, do not specify `sizeof(SceAudiodecInfo)`. Instead, specify the size of the structure corresponding to the type of audio decoder to be used.

See Also

`SceAudiodecInfoAt9`, `SceAudiodecInfoMp3`, `SceAudiodecInfoAac`, `SceAudiodecInfoCelp`

SceAudiodecInfoAt9

ATRAC9™ information structure

Definition

```
#include <audiodec.h>
typedef struct SceAudiodecInfoAt9 {
        SceUInt32 size;
        SceUInt8 configData[4];
        SceUInt32 ch;
        SceUInt32 bitRate;
        SceUInt32 samplingRate;
        SceUInt32 superFrameSize;
        SceUInt32 framesInSuperFrame;
} SceAudiodecInfoAt9;
```

Members

size	Size of the structure
configData	ATRAC9™ settings information
ch	Number of channels
bitRate	Bit rate (in kbps)
samplingRate	Sampling frequency (in Hz)
superFrameSize	Superframe size (in Bytes)
framesInSuperFrame	Number of frames in Superframe

Description

This structure is for ATRAC9™ information.

See Also

`SceAudiodecCtrl`, `sceAudiodecCreateDecoder()`, `sceAudiodecDeleteDecoder()`, `sceAudiodecDecode()`

SceAudiodecInfoMp3

MP3 information structure

Definition

```
#include <audiodec.h>
typedef struct SceAudiodecInfoMp3 {
        SceUInt32 size;
        SceUInt32 ch;
        SceUInt32 version;
} SceAudiodecInfoMp3;
```

Members

size	Size of the structure
ch	Number of channels
version	MPEG version

Description

This structure is for MP3 information.

See Also

`SceAudiodecCtrl`, `sceAudiodecCreateDecoder()`, `sceAudiodecDeleteDecoder()`, `sceAudiodecDecode()`

SceAudiodecInfoAac

AAC information structure

Definition

```
#include <audiodec.h>
typedef struct SceAudiodecInfoAac {
        SceUInt32 size;
        SceUInt32 isAdts;
        SceUInt32 ch;
        SceUInt32 samplingRate;
        SceUInt32 isSbr;
} SceAudiodecInfoAac;
```

Members

size	Size of the structure
isAdts	Flag indicating the presence of ADTS headers
ch	Number of channels
samplingRate	Sampling frequency (in Hz)
isSbr	Flag indicating the presence of Spectral Band Replication (SBR) to be added to HE-AAC

Description

This structure is for AAC information.

See Also

`SceAudiodecCtrl`, `sceAudiodecCreateDecoder()`, `sceAudiodecDeleteDecoder()`, `sceAudiodecDecode()`

SceAudiodecInfoCelp

CELP information structure

Definition

```
#include <audiodec.h>
typedef struct SceAudiodecInfoCelp {
        SceUInt32 size;
        SceUInt32 excitationMode;
        SceUInt32 samplingRate;
        SceUInt32 bitRate;
        SceUInt32 lostCount;
} SceAudiodecInfoCelp;
```

Members

size	Size of the structure
excitationMode	Excitation mode
samplingRate	Sampling frequency (in Hz)
bitRate	Bit rate (in bps)
lostCount	Lost input data count (0 when normal, 1 when data is lost)

Description

This structure is for CELP information.

See Also

`SceAudiodecCtrl`, `sceAudiodecCreateDecoder()`, `sceAudiodecDeleteDecoder()`, `sceAudiodecDecode()`

Initializing/Terminating the Library

sceAudiodecInitLibrary

Initialize libaudiodec

Definition

```
#include <audiodec.h>
SceInt32 sceAudiodecInitLibrary (
        SceUInt32 codecType,
        SceAudiodecInitParam *pInitParam
)
```

Arguments

codecType	Type of audio decoder
pInitParam	Pointer to libaudiodec initialization structure

Return Values

Value	Description
0 (SCE_OK)	Success
<0	Error SCE_AUDIODEC_ERROR_API_FAIL SCE_AUDIODEC_ERROR_INVALID_TYPE SCE_AUDIODEC_ERROR_INVALID_INIT_PARAM SCE_AUDIODEC_ERROR_ALREADY_INITILIZED SCE_AUDIODEC_ERROR_OUT_OF_MEMORY SCE_AUDIODEC_ERROR_INVALID_SIZE

Description

This function is used to initialize libaudiodec.

To *pInitParam*, specify the pointer to the libaudiodec initialization structure with initialization parameters set for each corresponding type of audio decoder. By calling this function, the required amount of memory will be allocated from the Codec Engine memory, and libaudiodec will be initialized. To release the allocated memory, call `sceAudiodecTermLibrary()`.

Notes

This function is multi-thread safe.

Examples

```
SceAudiodecInitParam audiodecInitParam;

// Sets audio decoder initialization parameters concerning ATRAC9(TM)
memset(&audiodecInitParam, 0, sizeof(audiodecInitParam));
audiodecInitParam.size = sizeof(audiodecInitParam.at9);
audiodecInitParam.at9.totalCh = 2;

// Initializes the ATRAC9(TM) library of libaudiodec
res = sceAudiodecInitLibrary(SCE_AUDIODEC_TYPE_AT9, &audiodecInitParam);
if (res < 0) {
        //Error handling
}
```

See Also

`SceAudiodecInitParam`, `sceAudiodecTermLibrary()`

sceAudiodecTermLibrary

Terminate libaudiodec

Definition

```
#include <audiodec.h>
SceInt32 sceAudiodecTermLibrary (
        SceUInt32 codecType
)
```

Arguments

codecType	Type of audio decoder

Return Values

Value	Description
0 (SCE_OK)	Success
<0	Error SCE_AUDIODEC_ERROR_INVALID_TYPE SCE_AUDIODEC_ERROR_NOT_INITIALIZED SCE_AUDIODEC_ERROR_A_HANDLE_IN_USE

Description

This function is used to terminate libaudiodec.

Call this function to delete all generated audio decoders and terminate libaudiodec. By calling this function, the memory area allocated by `sceAudiodecInitLibrary()` will be released. Note that when this function is called, all audio decoders corresponding to the specified type of audio decoder will need to be deleted.

Notes

This function is multi-thread safe.

Examples

```
// Terminates the ATRAC9(TM) library of libaudiodec
res = sceAudiodecTermLibrary(SCE_AUDIODEC_TYPE_AT9);
if (res < 0) {
        //Error handling
}
```

See Also

`sceAudiodecInitLibrary()`

Generating/Deleting Audio Decoders

sceAudiodecCreateDecoder

Generate audio decoders

Definition

```
#include <audiodec.h>
SceInt32 sceAudiodecCreateDecoder (
        SceAudiodecCtrl *pCtrl,
        SceUInt32 codecType
)
```

Arguments

pCtrl	Pointer to audio decoder control structure
codecType	Type of audio decoder

Return Values

Value	Description
0(SCE_OK)	Success
<0	Error SCE_AUDIODEC_ERROR_API_FAIL SCE_AUDIODEC_ERROR_INVALID_TYPE SCE_AUDIODEC_ERROR_NOT_INITIALIZED SCE_AUDIODEC_ERROR_ALL_HANDLES_IN_USE SCE_AUDIODEC_ERROR_INVALID_PTR SCE_AUDIODEC_ERROR_CH_SHORTAGE SCE_AUDIODEC_ERROR_INVALID_WORD_LENGTH SCE_AUDIODEC_ERROR_INVALID_SIZE SCE_AUDIODEC_AT9_ERROR_INVALID_CONFIG SCE_AUDIODEC_MP3_ERROR_INVALID_CH

Description

This function generates audio decoders.

By calling this function, the memory secured with `sceAudiodecInitLibrary()` will be allocated to the generated audio decoders.

Parameters set in `SceAudiodecCtrl` will depend on the type of audio decoder. Refer to Table 1 and Table 2 for parameter settings when calling this function.

Notes

This function is multi-thread safe.

Table 1 SceAudiodecCtrl Structure When Calling sceAudiodecCreateDecoder()

Member variable in `SceAudiodecCtrl` structure	ATRAC9™		MP3		AAC		CELP	
	in	out	in	out	in	out	in	out
size	○		○		○		○	
handle		○		○		○		○
pEs								
inputEsSize								
maxEsSize		○		○		○		○
pPcm								
outputPcmSize								
maxPcmSize		○		○		○		○
wordLength	○		○		○		○	
pInfo	○		○		○		○	

Table 2 SceAudiodecInfo Structure When Calling sceAudiodecCreateDecoder()

Member variable in **`SceAudiodecInfo`** structure	ATRAC9™		MP3		AAC		CELP	
	in	out	in	out	in	out	in	out
size	○							
configData	○							
ch		○						
bitRate		○						
samplingRate		○						
superFrameSize		○						
framesInSuperFrame		○						
size			○					
ch			○					
version			○					
size					○			
isAdts					○			
ch					○			
samplingRate					○			
isSbr					○			
size							○	
excitationMode							○	
samplingRate							○	
bitRate							○	
lostCount								

Examples

```
SceAudiodecCtrl audiodecCtrl;
SceAudiodecInfo audiodecInfo;

// Set SceAudiodecInfo
memset(&audiodecInfo, 0, sizeof(SceAudiodecInfo));
audiodecInfo.size = sizeof(audiodecInfo.at9);

// Set SceAudiodecCtrl
memset(&audiodecCtrl, 0, sizeof(SceAudiodecCtrl));
audiodecCtrl.size = sizeof(SceAudiodecCtrl);

// Set ATRAC9(TM)stream data
audiodecCtrl.wordLength = SCE_AUDIODEC_WORD_LENGTH_16BITS;
memcpy(audiodecInfo.at9.configData, header.fmtChunk.configData,
sizeof(audiodecInfo.at9.configData));
audiodecCtrl.pInfo = &audiodecInfo;

// Generate ATRAC9(TM) audio decoders
res = sceAudiodecCreateDecoder(&audiodecCtrl, SCE_AUDIODEC_TYPE_AT9);
if (res < 0) {
        //Error handling
}
```

See Also

`SceAudiodecCtrl,sceAudiodecDeleteDecoder()`

sceAudiodecDeleteDecoder

Delete audio decoders

Definition

```
#include <audiodec.h>
SceInt32 sceAudiodecDeleteDecoder (
        SceAudiodecCtrl *pCtrl
)
```

Arguments

pCtrl	Pointer to the audio decoder control structure

Return Values

Value	Description
0 (SCE_OK)	Success
<0	Error SCE_AUDIODEC_ERROR_API_FAIL SCE_AUDIODEC_ERROR_INVALID_TYPE SCE_AUDIODEC_ERROR_NOT_INITIALIZED SCE_AUDIODEC_ERROR_INVALID_PTR SCE_AUDIODEC_ERROR_INVALID_HANDLE SCE_AUDIODEC_ERROR_NOT_HANDLE_IN_USE SCE_AUDIODEC_ERROR_INVALID_WORD_LENGTH SCE_AUDIODEC_ERROR_INVALID_SIZE

Description

This function deletes audio decoders.

By calling this function, the memory allocated for the audio decoders using `sceAudiodecCreateDecoder()` will be released. When terminating libaudiodec by using `sceAudiodecTermLibrary()`, all audio decoders corresponding to the type of audio decoder need to be deleted by using this function.

Parameters set in `SceAudiodecCtrl` will depend on the type of audio decoder. Refer to Table 3 and Table 4 for parameter settings when calling this function.

Notes

This function is multi-thread safe. However, two separate function calls among the `sceAudiodecDeleteDecoder()`, `sceAudiodecDeleteDecoderExternal()`, `sceAudiodecDecode()`, `sceAudiodecDecodeNFrames()`, `sceAudiodecDecodeNStreams()`, and `sceAudiodecClearContext()` functions for the same instance are not multi-thread safe. When making a call that is not multi-thread safe, the API that is called later will return `SCE_AUDIODEC_ERROR_BUSY`.

Table 3 SceAudiodecCtrl Structure When Calling sceAudiodecDeleteDecoder()

Member variable in `SceAudiodecCtrl` structure	ATRAC9™		MP3		AAC		CELP	
	in	out	in	out	in	out	in	out
`size`	○		○		○		○	
`handle`	○		○		○		○	
`pEs`								
`inputEsSize`								
`maxEsSize`								
`pPcm`								
`outputPcmSize`								
`maxPcmSize`								
`wordLength`	○		○		○		○	
`pInfo`	○		○		○		○	

Table 4 SceAudiodecInfo Structure When Calling sceAudiodecDeleteDecoder()

Member variable in `SceAudiodecInfo` structure	ATRAC9™		MP3		AAC		CELP	
	in	out	in	out	in	out	in	out
`size`	○							
`configData`								
`ch`								
`bitRate`								
`samplingRate`								
`superFrameSize`								
`framesInSuperFrame`								
`size`			○					
`ch`								
`version`								
`size`					○			
`isAdts`								
`ch`								
`samplingRate`								
`isSbr`								
`size`							○	
`excitationMode`								
`samplingRate`								
`bitRate`								
`lostCount`								

Examples

```
SceAudiodecCtrl audiodecCtrl;

// Generate audio decoders

// Delete audio decoders
res = sceAudiodecDeleteDecoder(&audiodecCtrl);
if (res < 0) {
        //Error handling
}
```

See Also

`SceAudiodecCtrl`, `sceAudiodecCreateDecoder()`

sceAudiodecGetContextSize

Get memory size allocated to audio decoder

Definition

```
#include <audiodec.h>
SceInt32 sceAudiodecGetContextSize (
        SceAudiodecCtrl *pCtrl,
        SceUInt32 codecType
)
```

Arguments

pCtrl	Pointer to the audio decoder control structure
codecType	Type of audio decoder

Return Values

Value	Description
0(SCE_OK)	Success
<0	Error

Description

This function obtains the memory size allocated to the audio decoder.

The obtained value is used with `sceAudiodecCreateDecoderExternal()`.

Parameters set in `SceAudiodecCtrl` depend on the type of audio decoder. The parameter settings when this function is called conform to `sceAudiodecCreateDecoder()`.

Notes

This function is multi-thread safe.

Table 5 SceAudiodecCtrl Structure When Calling sceAudiodecGetContextSize()

Member variable in `SceAudiodecCtrl` structure	ATRAC9™		MP3		AAC		CELP	
	in	out	in	out	in	out	in	out
size	○		○		○		○	
handle								
pEs								
inputEsSize								
maxEsSize								
pPcm								
outputPcmSize								
maxPcmSize								
wordLength	○		○		○		○	
pInfo	○		○		○		○	

Table 6 SceAudiodecInfo Structure When Calling sceAudiodecGetContextSize()

Member variable in `SceAudiodecInfo` structure	ATRAC9™		MP3		AAC		CELP	
	in	out	in	out	in	out	in	out
size	○							
configData	○							
ch								
bitRate								
samplingRate								
superFrameSize								
framesInSuperFrame								
size			○					
ch			○					
version								
size					○			
isAdts								
ch					○			
samplingRate								
isSbr								
size							○	
excitationMode								
samplingRate								
bitRate								
lostCount								

Examples

```
SceAudiodecCtrl audiodecCtrl;
SceAudiodecInfo audiodecInfo;

uint32_t contextSize = 0;

// Set SceAudiodecInfo
memset(&audiodecInfo, 0, sizeof(SceAudiodecInfo));
audiodecInfo.size = sizeof(audiodecInfo.aac);
audiodecInfo.aac.isAdts = 1;
audiodecInfo.aac.ch = 2;
audiodecInfo.aac.samplingRate = 48000;
audiodecInfo.aac.isSbr = 0;
// Set SceAudiodecCtrl
memset(&audiodecCtrl, 0, sizeof(SceAudiodecCtrl));
audiodecCtrl.size = sizeof(SceAudiodecCtrl);
audiodecCtrl.wordLength = SCE_AUDIODEC_WORD_LENGTH_16BITS;
audiodecCtrl.pInfo = &audiodecInfo;

// Get required memory size
contextSize = sceAudiodecGetContextSize(&audiodecCtrl,
SCE_AUDIODEC_TYPE_AAC);
if (contextSize <= 0) {
        //Error handling
}
```

See Also

SceAudiodecCtrl,sceAudiodecCreateDecoderExternal()

sceAudiodecCreateDecoderExternal

Generate audio decoder with specified memory

Definition

```
#include <audiodec.h>
SceInt32 sceAudiodecCreateDecoderExternal (
        SceAudiodecCtrl *pCtrl,
        SceUInt32 codecType,
        SceUIntVAddr vaContext,
        SceUInt32 contextSize
)
```

Arguments

pCtrl	Pointer to audio decoder control structure
codecType	Type of audio decoder
vaContext	Starting address of context memory
contextSize	Size of context memory

Return Values

Value	Description
0(SCE_OK)	Success
<0	Error

Description

This function generates an audio decoder with specified memory.

The memory specified with this function is allocated to the generated audio decoder. `sceAudiodecInitLibrary()` does not need to be executed before executing this function.

However, memory can only be specified to this function for cache-disabled and physical continuous memory that is enabled for reading and writing by the Codec Engine but not by the user and for which `sceCodecEngineOpenUnmapMemBlock()` has been applied. In addition, allocate that memory only with the context size obtained with `sceAudiodecGetContextSize()`, and specify that size to the *contextSize* argument.

Parameters set in `SceAudiodecCtrl` depend on the type of audio decoder. The parameter settings when this function is called conform to `sceAudiodecCreateDecoder()`.

Notes

This function is multi-thread safe.

Examples

```
SceUID uidMemBlock, uidUnmap;
const uint32_t memBlockSize = 0x100000U;
void *pMemBlock = NULL;

SceAudiodecCtrl audiodecCtrl;
SceAudiodecInfo audiodecInfo;

uint32_t vaContext = 0;
uint32_t contextSize = 0;
```

```
        // Allocate a cache-disabled and physical continuous memory that is enabled for
        // reading and writing by the user
        uidMemBlock = sceKernelAllocMemBlock("PhysicallyContiguousMemoryLpddr",
        SCE_KERNEL_MEMBLOCK_TYPE_USER_MAIN_PHYCONT_NC_RW, memBlockSize, NULL);
        if (uidMemBlock < 0) {
                //Error handling
        }
        // Obtain starting address of allocated memory
        res = sceKernelGetMemBlockBase(uidMemBlock, &pMemBlock);
        if (res < 0) {
                //Error handling
        }
        // Remap as a cache-disabled and physical continuous memory that is enabled for
        // reading and writing by the Codec Engine but not by the user
        uidUnmap = sceCodecEngineOpenUnmapMemBlock(pMemBlock, memBlockSize);
        if (uidUnmap < 0) {
                //Error handling
        }

        // Set SceAudiodecInfo
        memset(&audiodecInfo, 0, sizeof(SceAudiodecInfo));
        audiodecInfo.size = sizeof(audiodecInfo.aac);
        audiodecInfo.aac.isAdts = 1;
        audiodecInfo.aac.ch = 2;
        audiodecInfo.aac.samplingRate = 48000;
        audiodecInfo.aac.isSbr = 0;
        // Set SceAudiodecCtrl
        memset(&audiodecCtrl, 0, sizeof(SceAudiodecCtrl));
        audiodecCtrl.size = sizeof(SceAudiodecCtrl);
        audiodecCtrl.wordLength = SCE_AUDIODEC_WORD_LENGTH_16BITS;
        audiodecCtrl.pInfo = &audiodecInfo;

        // Obtain required memory size
        contextSize = sceAudiodecGetContextSize(&audiodecCtrl,
        SCE_AUDIODEC_TYPE_AAC);
        if (contextSize <= 0) {
                //Error handling
        }
        // Allocate memory from remapped area
        vaContext = sceCodecEngineAllocMemoryFromUnmapMemBlock(uidUnmap, contextSize,
        SCE_AUDIODEC_ALIGNMENT_SIZE);
        if (vaContext == 0) {
                //Error handling
        }
        // Generate AAC audio decoders
        res = sceAudiodecCreateDecoderExternal(&audiodecCtrl, SCE_AUDIODEC_TYPE_AAC,
        vaContext, contextSize);
        if (res < 0) {
                //Error handling
        }
```

See Also

`SceAudiodecCtrl`, `sceAudiodecGetContextSize()`, `sceAudiodecDeleteDecoderExternal()`

sceAudiodecDeleteDecoderExternal

Delete audio decoder generated with specified memory

Definition

```
#include <audiodec.h>
SceInt32 sceAudiodecDeleteDecoderExternal (
        SceAudiodecCtrl *pCtrl,
        SceUIntVAddr *pvaContext
)
```

Arguments

pCtrl	Pointer to audio decoder control structure
pvaContext	Pointer to starting address of context memory

Return Values

Value	Description
0(SCE_OK)	Success
<0	Error

Description

This function deletes the audio decoder generated with specified memory.

Memory allocated to the audio decoder is not released. Memory must be released after executing this function. The address value that can be obtained with the *pvaContext* argument can be used at this time. This value is the value specified to `sceAudiodecCreateDecoderExternal()`.

In addition, `sceAudiodecTermLibrary()` does not need to be executed after executing this function.

Parameters set in `SceAudiodecCtrl` depend on the type of audio decoder. The parameter settings when this function is called conform to `sceAudiodecDeleteDecoder()`.

Notes

This function is multi-thread safe. However, two separate function calls among the `sceAudiodecDeleteDecoder()`, `sceAudiodecDeleteDecoderExternal()`, `sceAudiodecDecode()`, `sceAudiodecDecodeNFrames()`, `sceAudiodecDecodeNStreams()`, and `sceAudiodecClearContext()` functions for the same instance are not multi-thread safe. When making a call that is not multi-thread safe, the API that is called later will return `SCE_AUDIODEC_ERROR_BUSY`.

Examples

```
SceUID uidUnmap;
SceAudiodecCtrl audiodecCtrl;
uint32_t vaContext = 0;

// Generate audio decoders

// Delete audio decoders
res = sceAudiodecDeleteDecoderExternal(&audiodecCtrl, &vaContext);
if (res < 0) {
        //Error handling
}
// Release memory from remapped area
res = sceCodecEngineFreeMemoryFromUnmapMemBlock(uidUnmap, vaContext);
if (res < 0) {
        //Error handling
}
```

See Also

SceAudiodecCtrl,sceAudiodecCreateDecoderExternal()

Decoding Audio Data

sceAudiodecDecode

Decode audio data

Definition

```
#include <audiodec.h>
SceInt32 sceAudiodecDecode (
        SceAudiodecCtrl *pCtrl
)
```

Arguments

pCtrl	Pointer to the audio decoder control structure

Return Values

Value	Description
0 (`SCE_OK`)	Success

<0	Error
	`SCE_AUDIODEC_ERROR_API_FAIL`
	`SCE_AUDIODEC_ERROR_INVALID_TYPE`
	`SCE_AUDIODEC_ERROR_NOT_INITIALIZED`
	`SCE_AUDIODEC_ERROR_INVALID_PTR`
	`SCE_AUDIODEC_ERROR_INVALID_HANDLE`
	`SCE_AUDIODEC_ERROR_NOT_HANDLE_IN_USE`
	`SCE_AUDIODEC_ERROR_INVALID_WORD_LENGTH`
	`SCE_AUDIODEC_ERROR_INVALID_SIZE`

Description

This function decodes audio data.

By calling this function, elementary streams loaded to *pEs* will be decoded, and decoded PCM data will be overwritten in *pPcm*. At this time, the elementary stream size used for decoding and the output PCM data size are stored to *inputEsSize* and *outputPcmSize*.

Parameters set in `SceAudiodecCtrl` will depend on the type of audio decoder. Refer to Table 7 and Table 8 for parameter settings when calling this function.

Notes

- The maximum value of the elementary stream size per frame will be set in *maxEsSize* when `sceAudiodecCreateDecoder()` is called. The buffer set to *pEs* should be equal to or larger than *maxEsSize*.

- The *pEs* buffer area will be accessed by both the ARM and the Codec Engine. At this time, cache coherency must be secured between the ARM and the Codec Engine.
  In order to secure this cache coherency, **memory which is 256 bytes aligned and whose size is a multiple of 256 bytes must be allocated for the *pEs* buffer. However, *pEs*, the starting address of the elementary stream, does not require 256 bytes of alignment. Do not specify the same buffer area at the same time for several decoders.**

- The maximum value of the PCM size per frame will be set in *maxPcmSize* when `sceAudiodecCreateDecoder()` is called. For the buffer set in *pPcm*, set aside an area equal to or greater than *maxPcmSize*.

- The `pPcm` buffer area will be accessed by both the ARM and the Codec Engine. At this time, cache coherency must be secured between the ARM and the Codec Engine.
  In order to secure this cache coherency, **memory which is 256 bytes aligned and whose size is a multiple of 256 bytes must be allocated for the *`pPcm`* buffer. However, *`pPcm`*, the PCM starting address, does not require 256 bytes of alignment.**

Notes

Although this function is multi-thread safe when called for differing instances, two separate function calls among the `sceAudiodecDeleteDecoder()`, `sceAudiodecDeleteDecoderExternal()`, `sceAudiodecDecode()`, `sceAudiodecDecodeNFrames()`, `sceAudiodecDecodeNStreams()`, and `sceAudiodecClearContext()` functions for the same instance are not multi-thread safe. When making a call that is not multi-thread safe, the API that is called later will return `SCE_AUDIODEC_ERROR_BUSY`.

Table 7 SceAudiodecCtrl Structure When Calling sceAudiodecDecode()

Member variable in `SceAudiodecCtrl` structure	ATRAC9™		MP3		AAC		CELP	
	in	out	in	out	in	out	in	out
size	○		○		○		○	
handle	○		○		○		○	
pEs	○		○		○		○	
inputEsSize		○		○		○		○
maxEsSize								
pPcm	○		○		○		○	
outputPcmSize		○		○		○		○
maxPcmSize								
wordLength	○		○		○		○	
pInfo	○		○		○		○	

Table 8 SceAudiodecInfo Structure When Calling sceAudiodecDecode()

Member variable in **`SceAudiodecInfo`** structure	ATRAC9™		MP3		AAC		CELP	
	in	out	in	out	in	out	in	out
size	○							
configData								
ch								
bitRate								
samplingRate								
superFrameSize								
framesInSuperFrame								
size			○					
ch								
version								
size					○			
isAdts								
ch								
samplingRate								
isSbr								
size							○	
excitationMode								
samplingRate								
bitRate								
lostCount							○	

Examples

```
SceAudiodecCtrl audiodecCtrl;

// Generate audio decoders

// Set input/output buffer
audiodecCtrl.pEs  = esBuffer;
audiodecCtrl.pPcm = pcmBuffer;

// Decode audio data
res = sceAudiodecDecode(&audiodecCtrl);
if (res < 0) {
        //Error handling
}
```

See Also

SceAudiodecCtrl

sceAudiodecDecodeNFrames

Collectively decode multiple frames

Definition

```
#include <audiodec.h>
SceInt32 sceAudiodecDecodeNFrames (
        SceAudiodecCtrl *pCtrl,
        SceUInt32 nFrames
)
```

Arguments

pCtrl	Pointer to the audio decoder control structure
nFrames	Number of frames to be decoded collectively

Return Values

Value	Description
0(SCE_OK)	Success
<0	Error SCE_AUDIODEC_ERROR_API_FAIL SCE_AUDIODEC_ERROR_INVALID_TYPE SCE_AUDIODEC_ERROR_NOT_INITIALIZED SCE_AUDIODEC_ERROR_INVALID_PTR SCE_AUDIODEC_ERROR_INVALID_HANDLE SCE_AUDIODEC_ERROR_NOT_HANDLE_IN_USE SCE_AUDIODEC_ERROR_INVALID_WORD_LENGTH SCE_AUDIODEC_ERROR_INVALID_SIZE SCE_AUDIODEC_ERROR_INVALID_NFRAMES

Description

This function collectively decodes multiple frames.

By replacing multiple calls of `sceAudiodecDecode()` with this function, the ARM load can be reduced. However, each buffer size must be proportional to the number of frames.

By calling this function, elementary streams loaded to *pEs* will be decoded, and decoded PCM data will be overwritten in *pPcm*. At this time, the elementary stream size used for decoding and the output PCM data size are stored to *inputEsSize* and *outputPcmSize*.

Parameters set in `SceAudiodecCtrl` will depend on the type of audio decoder. Refer to Table 9 and Table 10 for parameter settings when calling this function.

There is a limit to the number of frames that can be batch decoded, and in cases other than ATRAC9™, this is 1. For details, refer to the "Maximum Value for the Number of Frames Available for Collective Decoding" section.

Notes

- The maximum value of the elementary stream size per frame will be set in `maxEsSize` when `sceAudiodecCreateDecoder()` is called. The buffer set to `pEs` should be equal to or larger than `maxEsSize` x `nFrames`.
- The `pEs` buffer area will be accessed by both the ARM and the Codec Engine. At this time, cache coherency must be secured between the ARM and the Codec Engine.
  In order to secure this cache coherency, **memory which is 256 bytes aligned and whose size is a multiple of 256 bytes must be allocated for the `pEs` buffer. However, `pEs`, the starting address of the elementary stream, does not require 256 bytes of alignment. Do not specify the same buffer area at the same time for several decoders.**
- The maximum value of the PCM size per frame will be set in `maxPcmSize` when `sceAudiodecCreateDecoder()` is called. For the buffer set in `pPcm`, set aside an area equal to or greater than `maxPcmSize` x `nFrames`.
- The `pPcm` buffer area will be accessed by both the ARM and the Codec Engine. At this time, cache coherency must be secured between the ARM and the Codec Engine.
  In order to secure this cache coherency, **memory which is 256 bytes aligned and whose size is a multiple of 256 bytes must be allocated for the `pPcm` buffer. However, `pPcm`, the PCM starting address, does not require 256 bytes of alignment.**

Notes

Although this function is multi-thread safe when called for differing instances, two separate function calls among the `sceAudiodecDeleteDecoder()`, `sceAudiodecDeleteDecoderExternal()`, `sceAudiodecDecode()`, `sceAudiodecDecodeNFrames()`, `sceAudiodecDecodeNStreams()`, and `sceAudiodecClearContext()` functions for the same instance are not multi-thread safe. When making a call that is not multi-thread safe, the API that is called later will return `SCE_AUDIODEC_ERROR_BUSY`.

Table 9 SceAudiodecCtrl Structure When Calling sceAudiodecDecodeNFrames()

Member variable in `SceAudiodecCtrl` structure	ATRAC9™		MP3		AAC		CELP	
	in	out	in	out	in	out	in	out
size	○		○		○		○	
handle	○		○		○		○	
pEs	○		○		○		○	
inputEsSize		○		○		○		○
maxEsSize								
pPcm	○		○		○		○	
outputPcmSize		○		○		○		○
maxPcmSize								
wordLength	○		○		○		○	
pInfo	○		○		○		○	

Table 10 SceAudiodecInfo Structure When Calling sceAudiodecDecodeNFrames()

Member variable in `SceAudiodecInfo` structure	ATRAC9™		MP3		AAC		CELP	
	in	out	in	out	in	out	in	out
size	○							
configData								
ch								
bitRate								
samplingRate								
superFrameSize								
framesInSuperFrame								
size			○					
ch								
version								
size					○			
isAdts								
ch								
samplingRate								
isSbr								
size							○	
excitationMode								
samplingRate								
bitRate								
lostCount							○	

Examples

```
SceAudiodecCtrl audiodecCtrl;

// Generate audio decoders

// Set input/output buffer
audiodecCtrl.pEs  = esBuffer;
audiodecCtrl.pPcm = pcmBuffer;

// Decode audio data
res = sceAudiodecDecodeNFrames(&audiodecCtrl, SCE_AUDIODEC_AT9_MAX_NFRAMES);
if (res < 0) {
        // Error handling
}
```

See Also

`SceAudiodecCtrl`, Maximum Value for the Number of Frames Available for Collective Decoding

sceAudiodecDecodeNStreams

Collectively decode multiple streams

Definition

```
#include <audiodec.h>
SceInt32 sceAudiodecDecodeNStreams (
        SceAudiodecCtrl *pCtrls[],
        SceUInt32 nStreams
)
```

Arguments

pCtrls	Array of pointer to the audio decoder control structure
nStreams	Number of streams to be decoded collectively

Return Values

Value	Description
0(`SCE_OK`)	Success

<0	Error
	`SCE_AUDIODEC_ERROR_API_FAIL`
	`SCE_AUDIODEC_ERROR_INVALID_TYPE`
	`SCE_AUDIODEC_ERROR_NOT_INITIALIZED`
	`SCE_AUDIODEC_ERROR_INVALID_PTR`
	`SCE_AUDIODEC_ERROR_INVALID_HANDLE`
	`SCE_AUDIODEC_ERROR_NOT_HANDLE_IN_USE`
	`SCE_AUDIODEC_ERROR_INVALID_WORD_LENGTH`
	`SCE_AUDIODEC_ERROR_INVALID_SIZE`
	`SCE_AUDIODEC_ERROR_INVALID_NSTREAMS`
	`SCE_AUDIODEC_ERROR_DIFFERENT_TYPES`
	`SCE_AUDIODEC_ERROR_SAME_HANDLES`

Description

This function decodes multiple streams one frame at a time. All specified streams must have the same audio decoder type.

By replacing multiple calls of `sceAudiodecDecode()` with this function, the ARM load can be reduced. However, the user should provide the system for combining multiple decoding requests. Note that the combining of multiple decoding requests involves latency and may not be suitable for decoding that requires immediacy.

By calling this function, elementary streams loaded to *pEs* will be decoded, and decoded PCM data will be overwritten in *pPcm*. At this time, the elementary stream size used for decoding and the output PCM data size are stored to *inputEsSize* and *outputPcmSize*.

Parameters set in `SceAudiodecCtrl` will depend on the type of audio decoder. Refer to Table 11 and Table 12 for parameter settings when calling this function.

The number of streams that can be decoded collectively is limited. For details, refer to the "Maximum Value for the Number of Streams Available for Collective Decoding" section.

Notes

- The maximum value of the elementary stream size per frame will be set in *`maxEsSize`* when `sceAudiodecCreateDecoder()` is called. The buffer set to *`pEs`* should be equal to or larger than *`maxEsSize`*.
- The *`pEs`* buffer area will be accessed by both the ARM and the Codec Engine. At this time, cache coherency must be secured between the ARM and the Codec Engine.
  In order to secure this cache coherency, **memory which is 256 bytes aligned and whose size is a multiple of 256 bytes must be allocated for the *`pEs`* buffer. However, *`pEs`*, the starting address of the elementary stream, does not require 256 bytes of alignment. Do not specify the same buffer area at the same time for several decoders.**
- The maximum value of the PCM size per frame will be set in *`maxPcmSize`* when `sceAudiodecCreateDecoder()` is called. For the buffer set in *`pPcm`*, set aside an area equal to or greater than *`maxPcmSize`*.
- The *`pPcm`* buffer area will be accessed by both the ARM and the Codec Engine. At this time, cache coherency must be secured between the ARM and the Codec Engine.
  In order to secure this cache coherency, **memory which is 256 bytes aligned and whose size is a multiple of 256 bytes must be allocated for the *`pPcm`* buffer. However, *`pPcm`*, the PCM starting address, does not require 256 bytes of alignment.**

Notes

Although this function is multi-thread safe when called for differing instances, two separate function calls among the `sceAudiodecDeleteDecoder()`, `sceAudiodecDeleteDecoderExternal()`, `sceAudiodecDecode()`, `sceAudiodecDecodeNFrames()`, `sceAudiodecDecodeNStreams()`, and `sceAudiodecClearContext()` functions for the same instance are not multi-thread safe. When making a call that is not multi-thread safe, the API that is called later will return `SCE_AUDIODEC_ERROR_BUSY`.

Table 11 SceAudiodecCtrl Structure When Calling sceAudiodecDecodeNStreams()

Member variable in `SceAudiodecCtrl` structure	ATRAC9™		MP3		AAC		CELP	
	in	out	in	out	in	out	in	out
size	○		○		○		○	
handle	○		○		○		○	
pEs	○		○		○		○	
inputEsSize		○		○		○		○
maxEsSize								
pPcm	○		○		○		○	
outputPcmSize		○		○		○		○
maxPcmSize								
wordLength	○		○		○		○	
pInfo	○		○		○		○	

Table 12 SceAudiodecInfo Structure When Calling sceAudiodecDecodeNStreams()

Member variable in `SceAudiodecInfo` structure	ATRAC9™		MP3		AAC		CELP	
	in	out	in	out	in	out	in	out
size	○							
configData								
ch								
bitRate								
samplingRate								
superFrameSize								
framesInSuperFrame								
size			○					
ch								
version								
size					○			
isAdts								
ch								
samplingRate								
isSbr								
size							○	
excitationMode								
samplingRate								
bitRate								
lostCount							○	

Examples

```
SceAudiodecCtrl audiodecCtrl[SCE_AUDIODEC_AT9_MAX_NSTREAMS];
SceAudiodecCtrl *pAudiodecCtrls[SCE_AUDIODEC_AT9_MAX_NSTREAMS];
int32_t i;

// Generate audio decoders

// Set input/output buffer
for (i = 0; i < SCE_AUDIODEC_AT9_MAX_NSTREAMS; i++) {
        audiodecCtrl[i].pEs  = esBuffer[i];
        audiodecCtrl[i].pPcm = pcmBuffer[i];
}
// pAudiodecCtrls setting
for (i = 0; i < SCE_AUDIODEC_AT9_MAX_NSTREAMS; i++) {
        pAudiodecCtrls[i] = &audiodecCtrl[i];
}

// Collectively decode multiple streams
res = sceAudiodecDecodeNStreams(pAudiodecCtrls,
                                SCE_AUDIODEC_AT9_MAX_NSTREAMS);
if (res < 0) {
        //Error handling
}
```

See Also

`SceAudiodecCtrl`, Maximum Value for the Number of Streams Available for Collective Decoding

sceAudiodecClearContext

Reinitialize audio decoders

Definition

```
#include <audiodec.h>
SceInt32 sceAudiodecClearContext (
        SceAudiodecCtrl *pCtrl
)
```

Arguments

pCtrl Pointer to the audio decoder control structure

Return Values

Value	Description
0 (`SCE_OK`)	Success

<0	Error
	`SCE_AUDIODEC_ERROR_API_FAIL`
	`SCE_AUDIODEC_ERROR_INVALID_TYPE`
	`SCE_AUDIODEC_ERROR_NOT_INITIALIZED`
	`SCE_AUDIODEC_ERROR_INVALID_PTR`
	`SCE_AUDIODEC_ERROR_INVALID_HANDLE`
	`SCE_AUDIODEC_ERROR_NOT_HANDLE_IN_USE`
	`SCE_AUDIODEC_ERROR_INVALID_WORD_LENGTH`
	`SCE_AUDIODEC_ERROR_INVALID_SIZE`

Description

This function reinitializes audio decoders.

By calling this function, the context memory is cleared and audio decoders are reinitialized.

Parameters set in `SceAudiodecCtrl` depend on the type of audio decoder. In terms of parameter settings when this function is called refer to Table 13 and Table 14.

This function is used for decoding non-continuous audio data as shown below.

- Seek Play
  Call this function immediately before redecoding the segment found using seek.
- Loop Play
  Call this function immediately before redecoding from the loop start.

Notes

Although this function is multi-thread safe when called for differing instances, two separate function calls among the `sceAudiodecDeleteDecoder()`, `sceAudiodecDeleteDecoderExternal()`, `sceAudiodecDecode()`, `sceAudiodecDecodeNFrames()`, `sceAudiodecDecodeNStreams()`, and `sceAudiodecClearContext()` functions for the same instance are not multi-thread safe. When making a call that is not multi-thread safe, the API that is called later will return `SCE_AUDIODEC_ERROR_BUSY`.

Table 13 SceAudiodecCtrl Structure When Calling sceAudiodecClearContext()

Member variable in `SceAudiodecCtrl` structure	ATRAC9™		MP3		AAC		CELP	
	in	out	in	out	in	out	in	out
size	○		○		○		○	
handle	○		○		○		○	
pEs								
inputEsSize								
maxEsSize								
pPcm								
outputPcmSize								
maxPcmSize								
wordLength	○		○		○		○	
pInfo	○		○		○		○	

Table 14 SceAudiodecInfo Structure When Calling sceAudiodecClearContext()

Member variable in `SceAudiodecInfo` structure	ATRAC9™		MP3		AAC		CELP	
	in	out	in	out	in	out	in	out
size	○							
configData								
ch								
bitRate								
samplingRate								
superFrameSize								
framesInSuperFrame								
size			○					
ch								
version								
size					○			
isAdts								
ch								
samplingRate								
isSbr								
size							○	
excitationMode							○	
samplingRate							○	
bitRate							○	
lostCount								

Examples

```
SceAudiodecCtrl audiodecCtrl;

// Generate audio decoders

// Decode audio data

// Reinitialize audio decoders
res = sceAudiodecClearContext(&audiodecCtrl);
if (res < 0) {
        //Error handling
}
```

See Also

SceAudiodecCtrl

Obtaining Information

sceAudiodecGetInternalError

Obtain internal errors

Definition

```
#include <audiodec.h>
SceInt32 sceAudiodecGetInternalError (
        SceAudiodecCtrl *pCtrl,
        SceInt32 *pInternalError
)
```

Arguments

pCtrl	Pointer to the audio decoder control structure
pInternalError	Pointer to internal error variables

Return Values

Value	Description
0 (SCE_OK)	Success
<0	Error SCE_AUDIODEC_ERROR_INVALID_TYPE SCE_AUDIODEC_ERROR_NOT_INITIALIZED SCE_AUDIODEC_ERROR_INVALID_PTR SCE_AUDIODEC_ERROR_INVALID_HANDLE SCE_AUDIODEC_ERROR_NOT_HANDLE_IN_USE SCE_AUDIODEC_ERROR_INVALID_WORD_LENGTH SCE_AUDIODEC_ERROR_INVALID_SIZE

Description

This function obtains internal errors from audio decoders.

By calling this function, details can be obtained regarding SCE_AUDIODEC_ERROR_API_FAIL internal errors within the Codec Engine.

Parameters set in SceAudiodecCtrl depend on the type of audio decoder. Refer to Table 15 and Table 16 for parameter settings when calling this function.

This function is provided for supporting debugging. Programming that uses data obtained with this function to modify controls is not recommended.

Notes

This function is multi-thread safe when called for differing instances.

Table 15 SceAudiodecCtrl Structure When Calling sceAudiodecGetInternalError()

Member variable in `SceAudiodecCtrl` **structure**	**ATRAC9™**		**MP3**		**AAC**		**CELP**	
	in	**out**	**in**	**out**	**in**	**out**	**in**	**out**
size	○		○		○		○	
handle	○		○		○		○	
pEs								
inputEsSize								
maxEsSize								
pPcm								
outputPcmSize								
maxPcmSize								
wordLength	○		○		○		○	
pInfo	○		○		○		○	

Table 16 SceAudiodecInfo Structure When Calling sceAudiodecGetInternalError()

Member variable in `SceAudiodecInfo` **structure**	**ATRAC9™**		**MP3**		**AAC**		**CELP**	
	in	**out**	**in**	**out**	**in**	**out**	**in**	**out**
size	○							
configData								
ch								
bitRate								
samplingRate								
superFrameSize								
framesInSuperFrame								
size			○					
ch								
version								
size					○			
isAdts								
ch								
samplingRate								
isSbr								
size							○	
excitationMode								
samplingRate								
bitRate								
lostCount								

Examples

```
SceAudiodecCtrl audiodecCtrl;
SceInt32 internalError;

// Generate audio decoders

// Obtain internal errors from audio decoders
res = sceAudiodecGetInternalError (&audiodecCtrl, &internalError);
if (res < 0) {
        //Error handling
}
```

See Also

`sceAudiodecCreateDecoder()`,`sceAudiodecDeleteDecoder()`,`sceAudiodecDecode()`, `sceAudiodecClearContext()`

Constants

Alignment Size

Alignment size

Definition

Value	(Number)	Description
`SCE_AUDIODEC_ALIGNMENT_SIZE`	0x100U	Alignment size

Description

This is the alignment size required for data accessed by the audio decoder.

Use this identifier when allocating an elementary stream buffer or PCM buffer provided to `sceAudiodecDecode()` or user memory for the audio decoder provided to `sceAudiodecCreateDecoderExternal()`.

Audio Decoder Types

Audio decoder types

Definition

Value	(Number)	Description
`SCE_AUDIODEC_TYPE_AT9`	0x1003U	ATRAC9™
`SCE_AUDIODEC_TYPE_MP3`	0x1004U	MP3
`SCE_AUDIODEC_TYPE_AAC`	0x1005U	MPEG4 AAC
`SCE_AUDIODEC_TYPE_CELP`	0x1006U	CELP

Description

This is an identifier that indicates the type of audio decoder.

When calling `sceAudiodecInitLibrary()`, `sceAudiodecTermLibrary()`, or `sceAudiodecCreateDecoder()`, specify this identifier.

Maximum Value for the Total Number of Channels Available for libaudiodec

Maximum value for the total number of channels available for libaudiodec

Definition

Value	(Number)	Description
`SCE_AUDIODEC_AT9_MAX_CH_IN_LIBRARY`	16	Maximum value for the total number of ATRAC9™ channels available for libaudiodec

Description

This identifier indicates the maximum value for the total number of channels available for libaudiodec.

When specifying the *`totalCh`* variable in the `SceAudiodecInitChParam` structure, ensure that it does not exceed this value.

Maximum Value for the Number of Streams Available for libaudiodec

Maximum value for the number of streams available for libaudiodec

Definition

Value	(Number)	Description
`SCE_AUDIODEC_MP3_MAX_STREAMS`	8	Maximum value for the number of MP3 streams available for libaudiodec
`SCE_AUDIODEC_AAC_MAX_STREAMS`	8	Maximum value for the number of AAC streams available for libaudiodec
`SCE_AUDIODEC_CELP_MAX_STREAMS`	8	Maximum value for the number of CELP streams available for libaudiodec

Description

This identifier indicates the maximum value for the number of streams available for libaudiodec.

When specifying the *`totalStreams`* variable in the `SceAudiodecInitStreamParam` structure, ensure that it does not exceed this value.

Number of PCM Quantization Bits

Number of PCM quantization bits

Definition

Value	(Number)	Description
`SCE_AUDIODEC_WORD_LENGTH_16BITS`	16	16 bits

Description

This identifier indicates the number of PCM quantization bits for audio decoders.

Set this identifier to the *`wordLength`* variable in the `SceAudiodecCtrl` structure.

Maximum Number of Channels

Maximum number of channels

Definition

Value	(Number)	Description
`SCE_AUDIODEC_AT9_MAX_CH_IN_DECODER`	2	Maximum number of channels for ATRAC9™ decoders
`SCE_AUDIODEC_MP3_MAX_CH_IN_DECODER`	2	Maximum number of channels for MP3 decoders
`SCE_AUDIODEC_AAC_MAX_CH_IN_DECODER`	2	Maximum number of channels for AAC decoders
`SCE_AUDIODEC_CELP_MAX_CH_IN_DECODER`	1	Maximum number of channels for CELP decoders

Description

This identifier indicates the maximum number of channels for audio decoders.

Maximum Number of Output Samples

Maximum number of output samples

Definition

Value	(Number)	Description
`SCE_AUDIODEC_AT9_MAX_SAMPLES`	256	Maximum number of output samples for ATRAC9™ decoders
`SCE_AUDIODEC_MP3_MAX_SAMPLES`	1152	Maximum number of output samples for MP3 decoders
`SCE_AUDIODEC_AAC_MAX_SAMPLES`	2048	Maximum number of output samples for AAC decoders
`SCE_AUDIODEC_CELP_MAX_SAMPLES`	320	Maximum number of output samples for CELP decoders

Description

This identifier indicates the maximum number of output samples for each audio decoder channel.

Each time `sceAudiodecDecode()` is called, the decoded PCM is output the same number of samples as the value of this identifier.

Maximum Size of Elementary Streams

Maximum size of elementary streams

Definition

Value	(Number)	Description
`SCE_AUDIODEC_AT9_MAX_ES_SIZE`	1024	Maximum size of elementary streams for ATRAC9™ decoders
`SCE_AUDIODEC_MP3_MAX_ES_SIZE`	1441	Maximum size of elementary streams for MP3 decoders
`SCE_AUDIODEC_AAC_MAX_ES_SIZE`	1792	Maximum size of elementary streams for AAC decoders
`SCE_AUDIODEC_CELP_MAX_ES_SIZE`	27	Maximum size of elementary streams for CELP decoders

Description

This identifier indicates the maximum size of elementary streams for audio decoders.

Each time `sceAudiodecDecode()` is called, elementary streams will be decoded up to a maximum of the value of this identifier.

Maximum Value for the Number of Frames Available for Collective Decoding

Maximum value for the number of frames available for collective decoding

Definition

Value	(Number)	Description
`SCE_AUDIODEC_AT9_MAX_NFRAMES`	8	Number of frames that the ATRAC9™ decoder can decode collectively
`SCE_AUDIODEC_MP3_MAX_NFRAMES`	1	Number of frames that the MP3 decoder can decode collectively
`SCE_AUDIODEC_AAC_MAX_NFRAMES`	1	Number of frames that the AAC decoder can decode collectively
`SCE_AUDIODEC_CELP_MAX_NFRAMES`	1	Number of frames that the CELP decoder can decode collectively

Description

These are the maximum values for the number of frames available for collective decoding.

When calling `sceAudiodecDecodeNFrames()`, set the *`nFrames`* argument so as not to exceed the above values.

Maximum Value for the Number of Streams Available for Collective Decoding

Maximum value for the number of streams available for collective decoding

Definition

Value	(Number)	Description
`SCE_AUDIODEC_AT9_MAX_NSTREAMS`	6	Number of streams that the ATRAC9™ decoder can decode collectively
`SCE_AUDIODEC_MP3_MAX_NSTREAMS`	6	Number of streams that the MP3 decoder can decode collectively
`SCE_AUDIODEC_AAC_MAX_NSTREAMS`	6	Number of streams that the AAC decoder can decode collectively
`SCE_AUDIODEC_CELP_MAX_NSTREAMS`	7	Number of streams that the CELP decoder can decode collectively

Description

These are the maximum values for the number of streams available for collective decoding.

When calling `sceAudiodecDecodeNStreams()`, set the *`nStreams`* argument so as not to exceed the above values.

Surplus Access Size of Elementary Streams

Surplus access size of elementary streams

Definition

Value	(Number)	Description
`SCE_AUDIODEC_AT9_EXTRA_ACCESS_SIZE`	0	Surplus access size of elementary streams for ATRAC9™ decoders
`SCE_AUDIODEC_MP3_EXTRA_ACCESS_SIZE`	0	Surplus access size of elementary streams for MP3 decoders
`SCE_AUDIODEC_AAC_EXTRA_ACCESS_SIZE`	0	Surplus access size of elementary streams for AAC decoders
`SCE_AUDIODEC_CELP_EXTRA_ACCESS_SIZE`	0	Surplus access size of elementary streams for CELP decoders

Description

This identifier indicates the surplus access size of elementary streams for audio decoders.

No surplus access is generated in any of the audio decoders.

MPEG Version

MPEG version

Definition

Value	(Number)	Description
`SCE_AUDIODEC_MP3_MPEG_VERSION_2_5`	0	MPEG version 2.5
`SCE_AUDIODEC_MP3_MPEG_VERSION_RESERVED`	1	Reserved
`SCE_AUDIODEC_MP3_MPEG_VERSION_2`	2	MPEG version 2
`SCE_AUDIODEC_MP3_MPEG_VERSION_1`	3	MPEG version 1

Description

This identifier indicates the MPEG version of the MP3.

CELP Excitation Mode

CELP excitation mode

Definition

Value	(Number)	Description
`SCE_AUDIODEC_CELP_MPE`	0	Multi-pulse excitation

Description

This identifier indicates the CELP excitation mode.

CELP Sampling Rate

CELP sampling rate

Definition

Value	(Number)	Description
`SCE_AUDIODEC_CELP_SAMPLING_RATE_8KHZ`	8000	8 kHz

Description

This identifier indicates the CELP sampling rate.

CELP Bit Rate

CELP bit rate

Definition

Value	(Number)	Description
SCE_AUDIODEC_CELP_BIT_RATE_3850BPS	3850	3850 bps
SCE_AUDIODEC_CELP_BIT_RATE_4650BPS	4650	4650 bps
SCE_AUDIODEC_CELP_BIT_RATE_5700BPS	5700	5700 bps
SCE_AUDIODEC_CELP_BIT_RATE_6600BPS	6600	6600 bps
SCE_AUDIODEC_CELP_BIT_RATE_7300BPS	7300	7300 bps
SCE_AUDIODEC_CELP_BIT_RATE_8700BPS	8700	8700 bps
SCE_AUDIODEC_CELP_BIT_RATE_9900BPS	9900	9900 bps
SCE_AUDIODEC_CELP_BIT_RATE_10700BPS	10700	10700 bps
SCE_AUDIODEC_CELP_BIT_RATE_11800BPS	11800	11800 bps
SCE_AUDIODEC_CELP_BIT_RATE_12200BPS	12200	12200 bps

Description

This identifier indicates the CELP bit rate.

Error Codes

List of error codes returned by libaudiodec

Definition

Value	(Number)	Description
`SCE_AUDIODEC_ERROR_API_FAIL`	0x807F0000	An internal error has occurred in the Codec Engine
`SCE_AUDIODEC_ERROR_INVALID_TYPE`	0x807F0001	Audio decoder type is invalid
`SCE_AUDIODEC_ERROR_INVALID_INIT_PARAM`	0x807F0002	Initialization parameter of libaudiodec is invalid
`SCE_AUDIODEC_ERROR_ALREADY_INITIALIZED`	0x807F0003	libaudiodec has already been initialized
`SCE_AUDIODEC_ERROR_OUT_OF_MEMORY`	0x807F0004	Insufficient memory
`SCE_AUDIODEC_ERROR_NOT_INITIALIZED`	0x807F0005	libaudiodec has not been initialized
`SCE_AUDIODEC_ERROR_A_HANDLE_IN_USE`	0x807F0006	A decoder is currently being used
`SCE_AUDIODEC_ERROR_ALL_HANDLES_IN_USE`	0x807F0007	All handles are being used
`SCE_AUDIODEC_ERROR_INVALID_PTR`	0x807F0008	The specified pointer is invalid
`SCE_AUDIODEC_ERROR_INVALID_HANDLE`	0x807F0009	The `SceAudiodecCtrl` structure handle is invalid
`SCE_AUDIODEC_ERROR_NOT_HANDLE_IN_USE`	0x807F000A	The `SceAudiodecCtrl` structure handle has not been used
`SCE_AUDIODEC_ERROR_CH_SHORTAGE`	0x807F000B	Insufficient number of channels available for libaudiodec
`SCE_AUDIODEC_ERROR_INVALID_WORD_LENGTH`	0x807F000C	The number of PCM quantization bits in the `SceAudiodecCtrl` structure is invalid
`SCE_AUDIODEC_ERROR_INVALID_SIZE`	0x807F000D	The size of a structure is invalid
`SCE_AUDIODEC_ERROR_UNSUPPORTED`	0x807F000E	The executed feature is not supported
`SCE_AUDIODEC_ERROR_INVALID_NFRAMES`	0x807F000F	Invalid number of frames for collective decoding
`SCE_AUDIODEC_ERROR_INVALID_NSTREAMS`	0x807F0010	Invalid number of streams for collective decoding
`SCE_AUDIODEC_ERROR_DIFFERENT_TYPES`	0x807F0011	Different audio decoder types in multiple streams
`SCE_AUDIODEC_ERROR_SAME_HANDLES`	0x807F0012	Same handles are set in multiple streams
`SCE_AUDIODEC_ERROR_BUSY`	0x807F0013	An API that is not multi-thread safe is called at the same time from a different thread
`SCE_AUDIODEC_AT9_ERROR_INVALID_CONFIG`	0x807F2000	ATRAC9™ information structure settings are invalid
`SCE_AUDIODEC_MP3_ERROR_INVALID_CH`	0x807F2800	Number of channels for the MP3 information structure is invalid
`SCE_AUDIODEC_MP3_ERROR_INVALID_MPEG_VERSION`	0x807F2801	The MPEG version of the MP3 information structure is invalid
`SCE_AUDIODEC_AAC_ERROR_INVALID_CH`	0x807F3000	Number of channels for the AAC information structure is invalid
`SCE_AUDIODEC_CELP_ERROR_INVALID_CONFIG`	0x807F3800	CELP information structure settings are invalid