Sndstream Library Reference

PlayStation®Vita Programmer Tool Runtime Library Release 3.50

Table of Contents

Introduction

Introduction

Sndstream is a library for streaming audio files on the PlayStation®Vita and PlayStation®4 platforms. Sndstream manages tasks such as reading audio files from disk, decoding compressed audio data, and filling audio buffers for playback. Once audio file playback has commenced, it continues in a background thread, using minimal CPU resources.

Sndstream is an add-on component to Scream. Sndstream requires Scream and its underlying synthesizer (NGS on the PlayStation®Vita platform and NGS2 on the PlayStation®4 platform) for audio playback, allocating synthesizer voices for each channel of played-back audio, and assigning Streams to Scream volume groups. The handles returned by the `sceScreamStartStream()`, `sceScreamStartStreamByFileToken()`, and `sceScreamStartStreamFromTransition()` functions can also be applied to Scream functions calls. This allows them to be manipulated along with other Scream Sounds.

This manual, the *Sndstream Library Reference*, documents the Sndstream library API used in conjunction with the NGS and NGS2 synthesizers, running on the PlayStation®Vita and PlayStation®4 platforms respectively. It documents all constants, type definitions, enumerations, structures, and functions required for successful operation. The *Sndstream Library Overview* is a counterpart to this document, and guides programmers through the functionality of Sndstream library

Parameter Checking

In the interests of efficiency, many functions do not check parameter values. For example, functions that require valid pointers do not always check when `NULL` pointers are passed. This is true of most functions that do not return error codes. For details, see the "Description" and "Return Values" sections for each function.

In general, it is the responsibility of the application to:

- Check for valid parameters before calling Scream and Sndstream functions.
- Check all return values for expected results.

Such checks can easily be removed in release versions of applications.

Function Return Values

In general, Sndstream functions return zero if execution was successful, and non-zero for unsuccessful execution. There are some exceptions to this, however. Notable exceptions are the Stream initialization functions `sceScreamStartStream()`, `sceScreamStartStreamByFileToken()`, and `sceScreamStartStreamFromTransition()`. These functions, upon successful execution, return a Scream Sound handle, by which the Stream can be referenced and manipulated using Scream and Sndstream function calls. Another group of exceptions are the status functions, which return values according to the data they retrieve – for example, current timing, looping, or buffer information.

Constants

Initialization Flags

Initialization flags provide a way to initialize Sndstream with certain behaviors. You apply the initialization flags to the `SceScreamSndStreamPlatformInit` *`flags`* member.

Define	Value	Description
`SCE_SCREAM_SND_SS_INIT_NO_PATH_COPY`	`(1L<<2)`	Initializes Sndstream to not allocate memory for file path storage. Instead, Sndstream stores a filename pointer in application memory.
`SCE_SCREAM_SND_SS_INIT_NO_MULTISTREAMS`	`(1L<<3)`	Initializes Sndstream without the capability to play multi-Layer Stream files. This reduces memory consumption slightly.

System Constants

System constants impose limits for various system resources and frequently-used parameters.

Define	Value	Description
`SCE_SCREAM_SND_STREAM_MAX_CHANNELS`	(8)	The maximum number of audio channels available for a single Stream.
`SCE_SCREAM_SND_FILE_QUEUE_MAX`	(7)	The maximum number of files that can be queued on a single Stream handle. **Note:** This number does not include the currently playing file, even if it is paused.
`SCE_SCREAM_SND_STREAM_MAX_PATH`	(256)	The maximum length of a file path string, including the terminating zero. See the `SceScreamSndFileParams` *`file`* member.
`SCE_SCREAM_SND_SYNC_CLOCKS_PER_QUARTER`	(24)	The number of sync clocks in a quarter-note.
`SCE_SCREAM_SND_STREAM_MAX_BITSTREAMS`	(16)	The maximum number of individual Bitstreams that can be associated with a multi-Layer Stream file. This number may be further constrained by bandwidth limitations. **Note:** Do not confuse this constant with the `sceScreamInitStreaming()` function's *`handleCount`* parameter, which sets the system-wide maximum number of simultaneously active Bitstreams that can occur in your game.
`SCE_SCREAM_SND_STREAM_INVALID_FILE_TOKEN`	(NULL)	An invalid `SceScreamSndStreamFileToken` value. Potentially returned by the `sceScreamParseStreamFile()` or `sceScreamGetFileTokenFromStorage()` functions.
`SCE_SCREAM_SND_STREAM_DEFAULT_MIDI_BUFFER_SIZE`	(2048)	Default size of the buffer used for Stream-associated MIDI files. See the `SceScreamSndStreamPlatformInit` *`midiBufferSize`* member.

Queuing Constants

Queuing constants define a range of values for the `sceScreamQueueToStream()` or `sceScreamQueueToStreamByFileToken()` functions' *`queueIndex`* parameter, when queuing Stream files to play on an active Stream.

Define	Value	Description
`SCE_SCREAM_SND_QUEUE_INDEX_HEAD`	`0`	Specifies the index position at the start of a Stream queue, that is, the location of the next file to play after the currently playing file.
`SCE_SCREAM_SND_QUEUE_INDEX_TAIL`	`SCE_SCREAM_SND_FILE_QUEUE_MAX`	Specifies the index position at the end of a Stream queue, that is, the location of the last file to play in a queue of up to `SCE_SCREAM_SND_FILE_QUEUE_MAX` files.

Memory Allocation Constants

Memory allocation constants are passed to the Scream Library memory allocation prototype `SceScreamExternSndMemAlloc` *`use`* parameter to indicate subsystem memory consumption.

Define	Value	Description
`SCE_SCREAM_SND_STREAM_MEM_USE_ALL`	`0x200`	Indicates that allocated memory is to be used for Sndstream.

File I/O Constants

File I/O constants define optional values used by Sndstream when calling the file I/O custom functions.

Define	Value	Description
`SCE_SCREAM_SND_STREAM_FILE_INVALID_HANDLE`	`((void *)-1)`	Indicates an invalid file handle. See the `SceScreamSndStreamFileOpenFunction()` or `SceScreamSndStreamFileAsyncOpenFunction()` prototypes.
`SCE_SCREAM_SND_STREAM_FILE_SEEK_SET`	`0`	A seek operation begins from the beginning of the file. See the `SceScreamSndStreamFileSeekFunction()` prototype *whence* parameter.
`SCE_SCREAM_SND_STREAM_FILE_SEEK_CUR`	`1`	A seek operation begins from the current offset in the file. See the `SceScreamSndStreamFileSeekFunction()` prototype *whence* parameter.
`SCE_SCREAM_SND_STREAM_FILE_SEEK_END`	`2`	A seek operation begins from the end of the file. See the `SceScreamSndStreamFileSeekFunction()` prototype *whence* parameter.
`SCE_SCREAM_SND_STREAM_FILE_PRIORITY_OPEN_START`	`(-2)`	The file being opened will start a new stream. See the `SceScreamSndStreamFileOpenFunction()` and `SceScreamSndStreamFileAsyncOpenFunction()` prototypes *priority* parameter.
`SCE_SCREAM_SND_STREAM_FILE_PRIORITY_OPEN_APPEND`	`(-3)`	The file being opened will be appended onto an existing stream. See the `SceScreamSndStreamFileOpenFunction()` and `SceScreamSndStreamFileAsyncOpenFunction()` prototypes *priority* parameter.
`SCE_SCREAM_SND_STREAM_FILE_PRIORITY_READ_PARSE`	`(-4)`	The purpose of the file read is for parsing header information. See the `SceScreamSndStreamFileReadFunction()` and `SceScreamSndStreamFileAsyncReadFunction()` prototypes *priority* parameter.
`SCE_SCREAM_SND_STREAM_FILE_PRIORITY_READ_DATA_FILL`	`(-5)`	The purpose of the file read is for initial filling of the playback buffer. See the `SceScreamSndStreamFileReadFunction()` and `SceScreamSndStreamFileAsyncReadFunction()` prototypes *priority* parameter.
`SCE_SCREAM_SND_STREAM_FILE_PRIORITY_READ_DATA_APPEND`	`(-6)`	The purpose of the file read is to add data to the playback buffer. See the `SceScreamSndStreamFileReadFunction()` and `SceScreamSndStreamFileAsyncReadFunction()` prototypes *priority* parameter.
`SCE_SCREAM_SND_STREAM_FILE_PRIORITY_OPEN_MIDI`	`(-7)`	The file being opened is a MIDI file associated with a stream file. See the `SceScreamSndStreamFileOpenFunction()` or `SceScreamSndStreamFileAsyncOpenFunction()` prototypes *priority* parameter.
`SCE_SCREAM_SND_STREAM_FILE_PRIORITY_READ_MIDI`	`(-8)`	The purpose of the file read is to load into memory a MIDI file associated with a stream file. See the `SceScreamSndStreamFileReadFunction()` and `SceScreamSndStreamFileAsyncReadFunction()` prototypes *priority* parameter.

Define	Value	Description
`SCE_SCREAM_SND_STREAM_FILE_MAX_BUFFER_IDS`	`(16)`	The maximum number of buffer IDs per Stream available to the custom asynchronous file I/O functions. See the `SceScreamSndStreamPlatformInit` *`subBufferCount`* member.
`SCE_SCREAM_SND_STREAM_FILE_DEFAULT_BUFFER_IDS`	`(8)`	The default number of buffer IDs per Stream available to the custom asynchronous file I/O functions. See the `SceScreamSndStreamPlatformInit` *`subBufferCount`* member.
`SCE_SCREAM_SND_STREAM_FILE_OPEN_PENDING`	`(0)`	A file open operation is pending. See `SceScreamSndStreamFileAsyncOpenFunction()` or `SceScreamSndStreamFileAsyncIsOpenCompleteFunction()`.
`SCE_SCREAM_SND_STREAM_FILE_OPEN_COMPLETE`	`(1)`	A file open operation is complete. See `SceScreamSndStreamFileAsyncOpenFunction()` or `SceScreamSndStreamFileAsyncIsOpenCompleteFunction()`.

File Parameters Constants

File parameters constants define optional Stream file behaviors.

Define	Value	Description
`SCE_SCREAM_SND_SS_FILE_HAS_MIDI_FILE`	`(1L<< 0)`	Indicates that a Stream file has an associated MIDI file of the same name. Apply to the `SceScreamSndFileParams` *`flags`* member. **Note:** The naming convention for associated MIDI files is to match the name of the Stream file, replacing the file's extension with a .mid extension. For example, an associated MIDI file for a Stream file named `musicClip1.vag` would be named `musicClip1.mid`.
`SCE_SCREAM_SND_SS_FILE_ALLOCATION_OK`	`(1L<< 1)`	Allows the `sceScreamParseStreamFile()` function to allocate memory for storing parsed file information, that is, if sufficient memory is allocated at initialization time. Apply to the `SceScreamSndFileParams` *`flags`* member.
`SCE_SCREAM_SND_SS_LOOP_INFINITE`	`(-1)`	Indicates that a Stream file should loop indefinitely. Apply to functions and structures with a loop count parameter, such as the `SceScreamSndFileParams` *`loopCount`* member or to the `sceScreamSetStreamFileLoopingCount()` *`loopCount`* parameter.
`SCE_SCREAM_SND_SS_LOOP_TILL_QUEUED`	`(-2)`	Indicates that a Stream file should loop until a new file is added to the – currently empty – queue on the same handle. Apply to functions and structures with a loop count parameter, such as the `SceScreamSndFileParams` *`loopCount`* member or to the `sceScreamSetStreamFileLoopingCount()` *`loopCount`* parameter.

Stream Initialization Constants

Stream initialization constants specify optional behaviors with which you can initialize a Stream.

Define	Value	Description
`SCE_SCREAM_SND_SS_START_PAUSED`	`(1L<< 0)`	Specifies that a Stream should start in a paused state. This can be useful for prebuffering a number of files to be started simultaneously. To begin playing a paused Stream, use the Scream Library functions `sceScreamContinueSound()` or `sceScreamContinueAllSoundsInGroup()`. Initialization option for starting a Stream that applies to the `SceScreamSndStartParams` *`flags`* member. Not valid when queuing a Stream to an active handle.
`SCE_SCREAM_SND_SS_START_VOICE_NO_STEAL`	`(1L<< 1)`	Specifies that voices allocated to a Stream handle cannot be stolen for other Scream or Sndstream voice requests. Initialization option for starting a Stream that applies to the `SceScreamSndStartParams` *`flags`* member.
`SCE_SCREAM_SND_SS_START_SMART_PAN`	`(1L<< 2)`	Initialization option for starting a Stream. Apply to the `SceScreamSndStartParams` *`flags`* member. For further information, see “Smart Pan” in the “Starting a Stream” chapter of the *Sndstream Library Overview*.
`SCE_SCREAM_SND_SS_START_SYNC_CLOCK`	`(1L<< 3)`	Sets a Stream as the current sync clock. Apply to the `SceScreamSndStartParams` *`flags`* member. The sync clock is used for Stream transitions using the `sceScreamStartStreamFromTransition()` function or for playing synchronized Scream Sounds using the `sceScreamPlaySoundSyncedByIndexEx()` and `sceScreamPlaySoundSyncedByNameEx()` functions. **Note:** A sync clock Stream must have an associated MIDI file specifying tempo and meter information.
`SCE_SCREAM_SND_SS_START_GET_VOICE_LEVEL`	`(1L<< 4)`	Specifies that a Stream's current voice level is available for retrieval by the `sceScreamGetStreamLevel()` function. Applicable to mono Streams only. Apply to the `SceScreamSndStartParams` *`flags`* member.
`SCE_SCREAM_SND_SS_START_ADSR1_DEFAULT`	`0x80FF`	Specifies the note-on (attack, decay, sustain) portion of a default Stream-specific envelope. Apply to the `SceScreamSndStartParams` *`adsr1`* member.
`SCE_SCREAM_SND_SS_START_ADSR2_DEFAULT`	`0xDFE0`	Specifies the note-off (release) portion of a default Stream-specific envelope. Apply to the `SceScreamSndStartParams` *`adsr2`* member.

Automated Parameter Change Flags

Optional behavior flags for the automated parameter change functions. For Scream details, see “Applying Automated Changes to Parameter Values” in the “Working with Sounds” chapter of the *Scream Library Overview*. For Sndstream details, see “Automated Incremental Settings” in the “Manipulating an Active Stream” chapter of the *Sndstream Library Overview*.

Define	Value	Description
`SCE_SCREAM_SND_AUTO_STOP_AT_DESTINATION`	`(1L << 0)`	Specifies that, upon reaching its target value, a Sound should stop. See the Scream functions `sceScreamAutoPan()`, `sceScreamAutoPitchTranspose()`, `sceScreamAutoPitchBend()`, `sceScreamAutoGain()`, and the Sndstream function `sceScreamAutoStreamLayerParams()`.
`SCE_SCREAM_SND_AUTO_REVERT_IF_ACTIVE`	`(1L << 1)`	Specifies that an automated parameter change that is still active (has not yet reached its target value) should return to its original value at the same rate of change as it set out. See the Scream functions `sceScreamAutoPan()`, `sceScreamAutoPitchTranspose()`, `sceScreamAutoPitchBend()`, `sceScreamAutoGain()`, and the Sndstream function `sceScreamAutoStreamLayerParams()`.
`SCE_SCREAM_SND_AUTO_COUNTER_CLOCKWISE`	`(1L << 2)`	Specifies that a panning parameter change should go in reverse direction producing a counter-clockwise panning motion. See the Scream function `sceScreamAutoPan()` and the Sndstream function `sceScreamAutoStreamLayerParams()`.
`SCE_SCREAM_SND_AUTO_TAKE_SHORTEST_PATH`	`(1L << 3)`	Specifies that a panning parameter change should go in whichever direction, clockwise or counter-clockwise, that provides the shortest path to the target. See the Scream function `sceScreamAutoPan()` and the Sndstream function `sceScreamAutoStreamLayerParams()`.
`SCE_SCREAM_SND_AUTO_USE_SEPARATE_FACTOR`	`(1L << 5)`	Specifies that an automated parameter change uses an automation-specific parameter factor rather than the default API parameter factor. See the Scream functions `sceScreamAutoPan()`, `sceScreamAutoPitchTranspose()`, `sceScreamAutoPitchBend()`, `sceScreamAutoGain()`, and the Sndstream function `sceScreamAutoStreamLayerParams()`.

Synchronization Constants

Synchronization constants specify optional behaviors for the synchronized Stream transition and playback functions.

Define	Value	Description
SCE_SCREAM_SND_SYNC_FLAG_ START_IF_NO_CLOCK	(1L << 0)	Synchronization behavior flag. Indicates that synchronized content should still play if there is no sync clock Stream to synchronize to. Used in the SceScreamSndSyncParams *syncFlags* member. **Note:** If there is no sync clock Stream and you do not set this flag, pending synchronized events do not play.
SCE_SCREAM_SND_SYNC_FLAG_ START_IF_CLOCK_ENDS	(1L << 1)	Synchronization behavior flag. Indicates that synchronized content should still play if the sync clock Stream terminates before a legal sync point is reached. Used in the SceScreamSndSyncParams *syncFlags* member. **Note:** If the sync clock Stream terminates and you do not set this flag, pending synchronized events do not play.
SCE_SCREAM_SND_SYNC_UNIT_ CONTENT	0	Specifies that the synchronization points are as defined in the content, that is, in accordance with marker(s) in the sync clock Stream's associated MIDI file. Used in the SceScreamSndSyncParams *syncUnit* member.
SCE_SCREAM_SND_SYNC_UNIT_ CLOCK	1	Specifies that the basic synchronization unit is a sync clock; that is, the synchronization point falls on a sync clock boundary. Used in the SceScreamSndSyncParams *syncUnit* member. See SCE_SCREAM_SND_SYNC_CLOCKS_PER_QUARTER.
SCE_SCREAM_SND_SYNC_UNIT_ BEAT	2	Specifies that the basic synchronization unit is a beat, that is, the synchronization point falls on a beat boundary. Used in the SceScreamSndSyncParams *syncUnit* member.
SCE_SCREAM_SND_SYNC_UNIT_ MEASURE	3	Specifies that the basic synchronization unit is a measure, that is, the synchronization point falls on a measure boundary. Used in the SceScreamSndSyncParams *syncUnit* member.
SCE_SCREAM_SND_SYNC_UNIT_ MARKER	4	Specifies that the basic synchronization unit is a marker (in the sync clock Stream's associated MIDI file), that is, the synchronization point falls on a MIDI marker boundary. Used in the SceScreamSndSyncParams *syncUnit* member.
SCE_SCREAM_SND_UNIT_CLOCK_ MULTIPLE_QUARTER_NOTE	(m*SCE_ SCREAM_SND_ SYNC_CLOCKS_ PER_QUARTER)	A macro for calculating the number of sync clocks in a given number of quarter-notes. See the SceScreamSndSyncParams *unitMultiple* member and SCE_SCREAM_SND_SYNC_CLOCKS_PER_QUARTER.
SCE_SCREAM_SND_UNIT_CLOCK_ MULTIPLE_EIGHTH_NOTE	(m*SCE_ SCREAM_SND_ SYNC_CLOCKS_ PER_ QUARTER/2)	A macro for calculating the number of sync clocks in a given number of eighth-notes. See the SceScreamSndSyncParams *unitMultiple* member and SCE_SCREAM_SND_SYNC_CLOCKS_PER_QUARTER.

Define	Value	Description
`SCE_SCREAM_SND_UNIT_CLOCK_MULTIPLE_SIXTEENTH_NOTE`	`(m*SCE_SCREAM_SND_SYNC_CLOCKS_PER_QUARTER/4)`	A macro for calculating the number of sync clocks in a given number of sixteenth-notes. See the `SceScreamSndSyncParams` *`unitMultiple`* member and `SCE_SCREAM_SND_SYNC_CLOCKS_PER_QUARTER`.

Transition Mode Constants

Transition mode constants specify optional behaviors for the `SceScreamSndTransitionParams` *`transitionMode`* member.

Define	Value	Description
`SCE_SCREAM_SND_TRANSITION_MODE_PLAY_WITH_MASTER`	0	Specifies that a transitioned (new) Stream actually plays along with a master (existing) Stream as the latter continues.
`SCE_SCREAM_SND_TRANSITION_MODE_FADEOUT_MASTER`	1	Specifies that the master (existing) Stream fades out in accordance with the `SceScreamSndTransitionParams` *`fadeOutTime`* and *`fadeOutGain`* members.
`SCE_SCREAM_SND_TRANSITION_MODE_KEYOFF_MASTER`	2	Specifies that the master (existing) Stream keys-off, that is, it enters the Release stage of an ADSR setting, rather than fading out at the transition point.

Associated Scream System Constants

The following Scream library System constants also apply to the Sndstream library.

Define	Value	Description
`SCE_SCREAM_SND_MAX_GAIN`	`1.0f`	The maximum gain level for Scream and Sndstream API calls that set gain. See the Scream `SceScreamSoundParams`, `SceScreamSndDistortionParams`, `SceScreamSndIIRFilterParams`, `SceScreamSynthParams`, `sceScreamSetMasterVolume()`, `sceScreamAutoGain()`, and the Sndstream `SceScreamSndBitstreamParams` and `SceScreamSndTransitionParams`.
`SCE_SCREAM_SND_MIN_GAIN`	`0.0f`	The minimum gain level for Scream and Sndstream API calls that set gain. See the Scream `SceScreamSoundParams`, `SceScreamSndDistortionParams`, `SceScreamSndIIRFilterParams`, `SceScreamSynthParams`, `sceScreamSetMasterVolume()`, `sceScreamAutoGain()`, and the Sndstream `SceScreamSndBitstreamParams` and `SceScreamSndTransitionParams`.
`SCE_SCREAM_SND_MAX_PREMASTER_SUBMIXES`	`4`	The maximum number of premaster submix voices. See the Scream `SceScreamSystemParams` structure's *`numPremasterCompSubmixes`* and *`numPremasterScCompSubmixes`* members, as well as the Sndstream functions `sceScreamStartStream()` and `sceScreamStartStreamByFileToken()` *`outputDest`* parameter.
`SCE_SCREAM_SND_DEFAULT_THREAD_PRIORITY`	`128`	Default thread priority. See the Scream `SceScreamSystemParams` *`tickThreadPriority`* member and the Sndstream `sceScreamFillDefaultPlatformInitArgs()` function.

Associated Scream Sound Constants

The following Scream library Sound constants also apply to the Sndstream library.

Define	Value	Description
`SCE_SCREAM_SND_MASK_GAIN`	`(1L << 2)`	The *`gain`* member has been set.
`SCE_SCREAM_SND_MASK_PAN_AZIMUTH`	`(1L << 3)`	The *`azimuth`* member has been set.
`SCE_SCREAM_SND_MASK_PAN_FOCUS`	`(1L << 4)`	The *`focus`* member has been set.

Associated Scream Sound Output Destinations

You use Sound output destinations when specifying a value for the *outputDest* parameter in the functions sceScreamStartStream() and sceScreamStartStreamByFileToken().

Define	Value	Description
SCE_SCREAM_SND_OUTPUT_DEST_PREMASTER_0	0	Specifies that a Sound's output destination is a pre-master submix. This constant expresses the first pre-master submix index. Add 1 to this value for each additional pre-master submix index. The number of available pre-master submixes is determined at initialization time using the Scream SceScreamSystemParams structure's *numPremasterCompSubmixes* and *numPremasterScCompSubmixes* members.
SCE_SCREAM_SND_OUTPUT_DEST_MASTER	(-1)	Specifies that a Sound's output destination is the master output. The default output destination in the *outputDest* parameter in Stream starting functions sceScreamStartStream() and sceScreamStartStreamByFileToken().
SCE_SCREAM_SND_OUTPUT_DEST_BY_GROUP	(-2)	Specifies that a Sound's output destination is inherited from that specified for the Group the Sound belongs to.

Data Structures

Summary

Sndstream data structures store data referenced in Sndstream functions.

Item	Description
SceScreamSndBitstreamParams	Stores gain, azimuth, and focus parameter values for one or more Stream Layers.
SceScreamSndFileInterface	Stores the addresses of custom file I/O functions.
SceScreamSndFileParams	Stores Stream file parameter values.
SceScreamSndStartParams	Stores the parameter values required for starting a Stream.
SceScreamSndStreamParseParams	Stores parameter values used when parsing a Stream file.
SceScreamSndStreamPlatformInit	Stores the platform-specific parameter values required for initializing Sndstream.
SceScreamSndStreamQueueParams	Stores playback information used when starting a new Stream or queuing a file to an existing Stream.
SceScreamSndSyncParams	Stores synchronization properties for Stream transitions or synchronized play of Scream Sounds.
SceScreamSndTransitionParams	Stores properties for a Stream transition.

SceScreamSndBitstreamParams

Stores gain, azimuth, and focus parameter values for one or more Stream Layers.

Definition

```
struct SceScreamSndBitstreamParams {
    uint32_t mask;
    float gain[SCE_SCREAM_SND_STREAM_MAX_BITSTREAMS];
    uint32_t azimuth[SCE_SCREAM_SND_STREAM_MAX_BITSTREAMS];
    uint32_t focus[SCE_SCREAM_SND_STREAM_MAX_BITSTREAMS];
};
```

Members

mask	A mask indicating which of the subsequent members have active settings. One or more of the following Scream Sound parameter bitmask constants: SCE_SCREAM_SND_MASK_GAIN, SCE_SCREAM_SND_MASK_PAN_AZIMUTH, SCE_SCREAM_SND_MASK_PAN_FOCUS. Use the OR operator to specify multiple selections. Set to NULL to leave existing (default, Bank contents, or Layer-specific) settings unchanged.
gain	An array of gain values for each Layer. Range: SCE_SCREAM_SND_MIN_GAIN to SCE_SCREAM_SND_MAX_GAIN.
azimuth	An array of azimuth values for each Layer. Expressed in degrees clockwise relative to the screen. Range: 0 to 359. Alternatively, you can set specific Output Speaker Targets. For further details, see "Output Speaker Targets" in the *Scream Library Reference* documents.
focus	An array of pan focus values for each Layer. Range: 0 to 360. Ignored if focus is set to an Output Speaker Target.

Description

This structure stores gain, azimuth, and focus parameter values for one or more Stream Layers. Its members store parameter values in arrays, one value for each Layer. The maximum length of the arrays is SCE_SCREAM_SND_STREAM_MAX_BITSTREAMS, which is the maximum number of Layers that can be contained in a Stream.

You can set Stream Layers parameters collectively (using the sceScreamSetStreamLayerParams() and sceScreamAutoStreamLayerParams() functions) or individually (using the Scream sceScreamSetSoundParamsEx(), sceScreamAutoGain(), and sceScreamAutoPan() functions). These functions allow you to manipulate Stream Layers as if live-mixing a multi-track recording.

Notes

Setting parameter values with this structure overrides any corresponding default settings or settings arising from Bank contents. For example, setting azimuth values for a stereo file would override the default channel speaker assignments (that is, to front left/right speakers).

See Also

SceScreamSndStartParams, sceScreamStartStream(), sceScreamSetStreamLayerParams(), sceScreamGetStreamLayerParams(), sceScreamAutoStreamLayerParams()

SceScreamSndFileInterface

Stores the addresses of custom file I/O functions.

Definition

```
struct SceScreamSndFileInterface {
    SceScreamSndStreamFileOpenFunction *m_pFileOpen;
    SceScreamSndStreamFileInfoCBFunction *m_pFileInfoCB;
    SceScreamSndStreamFileSeekFunction *m_pFileSeek;
    SceScreamSndStreamFileReadFunction *m_pFileRead;
    SceScreamSndStreamFileCloseFunction *m_pFileClose;
    SceScreamSndStreamFileAsyncOpenFunction *m_pFileAsyncOpen;
    SceScreamSndStreamFileAsyncIsOpenCompleteFunction
    *m_pFileAsyncIsOpenComplete;
    SceScreamSndStreamFileAsyncOpenBitstreamFunction
    *m_pFileAsyncOpenBitstream;
    SceScreamSndStreamFileAsyncReadFunction *m_pFileAsyncRead;
    SceScreamSndStreamFileAsyncIsReadCompleteFunction
    *m_pFileAsyncIsReadComplete;
    SceScreamSndStreamFileAsyncCloseBitstreamFunction
    *m_pFileAsyncCloseBitstream;
    SceScreamSndStreamFileAsyncCloseFunction *m_pFileAsyncClose;
};
```

Members

m_pFileOpen	Initialize with the address of a custom `SceScreamSndStreamFileOpenFunction()` function.
m_pFileInfoCB	Initialize with the address of a custom `SceScreamSndStreamFileInfoCBFunction()` function. Optional.
m_pFileSeek	Initialize with the address of a custom `SceScreamSndStreamFileSeekFunction()` function.
m_pFileRead	Initialize with the address of a custom `SceScreamSndStreamFileReadFunction()` function.
m_pFileClose	Initialize with the address of a custom `SceScreamSndStreamFileCloseFunction()` function.
m_pFileAsyncOpen	Initialize with the address of a custom `SceScreamSndStreamFileAsyncOpenFunction()` function.
m_pFileAsyncIsOpenComplete	Initialize with the address of a custom `SceScreamSndStreamFileAsyncIsOpenCompleteFunction()` function.
m_pFileAsyncOpenBitstream	Initialize with the address of a custom `SceScreamSndStreamFileAsyncOpenBitstreamFunction()` function.
m_pFileAsyncRead	Initialize with the address of a custom `SceScreamSndStreamFileAsyncReadFunction()` function.
m_pFileAsyncIsReadComplete	Initialize with the address of a custom `SceScreamSndStreamFileAsyncIsReadCompleteFunction()` function.
m_pFileAsyncCloseBitstream	Initialize with the address of a custom `SceScreamSndStreamFileAsyncCloseBitstreamFunction()` function.
m_pFileAsyncClose	Initialize with the address of a custom `SceScreamSndStreamFileAsyncCloseFunction()` function.

Description

The SceScreamSndFileInterface structure stores the addresses of custom file I/O functions. The structure stores addresses for both synchronous functions (used for parsing the file headers), and asynchronous functions (used for asynchronous reading of Stream data).

By default, SceScreamSndStream uses the FIOS file I/O functions on the PlayStation®Vita and PlayStation®4 platforms. To override the defaults and use custom file I/O functions:

(1) Ensure that your custom file I/O functions match the file I/O type definitions. See SceScreamSndStreamFileOpenFunction, and so on.

(2) Store their addresses as the corresponding members of the SceScreamSndFileInterface data structure.

(3) After initializing Sndstream, and before calling any other Sndstream functions, call sceScreamSetDefaultFileInterface() with the initialized SceScreamSndFileInterface data structure as its argument.

For further details, see "Using Custom File I/O Functions" in the "Working with System Globals" chapter of the *Sndstream Library Overview*.

Notes

You can use this structure as a value for the SceScreamSndFileParams *fileInterface* member and then use that SceScreamSndFileParams in sceScreamStartStream() to start a stream for a file, allowing file I/O specification on a per file basis.

See Also

sceScreamSetDefaultFileInterface(), SceScreamSndFileParams, SceScreamSndStreamFileOpenFunction(), SceScreamSndStreamFileInfoCBFunction(), SceScreamSndStreamFileSeekFunction(), SceScreamSndStreamFileReadFunction(), SceScreamSndStreamFileCloseFunction()

SceScreamSndFileParams

Stores Stream file parameter values.

Definition

```
struct SceScreamSndFileParams {
    const char *file;
    SceScreamSndFileInterface *fileInterface;
    uint32_t flags;
    int32_t loopCount;
    uint64_t seekOffset;
    float startSecond;
    SceScreamSndStreamUserContext userContext;
};
```

Members

file	ASCIIZ string pointer to the fully qualified path of the Stream file. For example, `mydir/mysubdir/mysound.wav`. Maximum length: `SCE_SCREAM_SND_STREAM_MAX_PATH`.
fileInterface	A pointer to a `SceScreamSndFileInterface` structure specifying the file interface to use for a Stream file. If `NULL`, the default file interface is used.
flags	Optional Stream file behaviors. Us the `OR` operator to combine multiple values: `SCE_SCREAM_SND_SS_FILE_HAS_MIDI_FILE`, `SCE_SCREAM_SND_SS_FILE_ALLOCATION_OK`.
loopCount	Specifies a number of additional loops to play. That is, 0 to play once without looping, 1 to play twice, 2 to play 3 times, and so on. To loop indefinitely, use the `SCE_SCREAM_SND_SS_LOOP_INFINITE` constant; to loop until another file has been queued on the handle, use `SCE_SCREAM_SND_SS_LOOP_TILL_QUEUED`. See "Notes" below.
seekOffset	Offset number of bytes into the file at which point to start reading. Used in cases where the streamed audio file is embedded in a larger container file. Points to the location of the beginning of the header of an embedded audio file. If the audio file is not embedded, set to zero.
startSecond	Offset number of seconds into the file at which point to start playback. Used in cases where the desired audio data starts other than at the beginning of the file (including where *seekOffset* > zero). Expressed in seconds. **Note:** applies to WAV and VAG file formats only.
userContext	User defined data that is passed to the file I/O interfaces.

Description

This structure stores parameter values related to a Stream file. It is used to provide information for starting streams from files. The `SceScreamSndStreamUserContext` value in *userContext* is returned in various status functions, such as `sceScreamGetStreamFileLocationInSeconds()`.

Notes

The *fileInterface* member allows file I/O specification on a per Stream basis.

Sndstream recognizes loop points embedded in ATRAC9™ Stream files. And if found, instead of looping around the entire file from start to end, looping playback takes place between the loop points. For further details, see "Working with Embedded Loop Points in Stream Files" in the "Starting a Stream" chapter of the *Sndstream Library Overview*.

See Also

`SceScreamSndFileInterface`, `sceScreamStartStream()`, `sceScreamStartStreamFromTransition()`, `sceScreamQueueToStream()`

SceScreamSndStartParams

Stores the parameter values required for starting a Stream.

Definition

```
struct SceScreamSndStartParams {
    uint32_t flags;
    int8_t volumeGroup;
    int8_t priority;
    int8_t reserved[2];
    float priorityReductionScale;
    uint16_t adsr1;
    uint16_t adsr2;
    SceScreamSndBitstreamParams bitstreamParams;
    SceScreamSndBitstreamParams layerParams;
    SceScreamSoundParams soundParams;
};
```

Members

flags	One or more of the following Stream Initialization constants (use the OR operator to make multiple selections): `SCE_SCREAM_SND_SS_START_PAUSED`, `SCE_SCREAM_SND_SS_START_VOICE_NO_STEAL`, `SCE_SCREAM_SND_SS_START_SMART_PAN`, `SCE_SCREAM_SND_SS_START_SYNC_CLOCK`.
volumeGroup	One of the Scream Library Volume Group constants. See “Volume Groups” in the *Scream Library Reference* documents for details.
priority	Voice allocation priority. Range: 0 to 127. Higher values indicate higher priorities, making allocated voices less likely to be stolen for new voice requests. If there are no free voices when the Stream is initialized, active voices with lower priority values are more likely to be stolen. A value of 127 can only be set from the API, and is therefore guaranteed to be of a higher priority than *Stream* Grain settings made in Scream Tool.
reserved	For internal use only.
priorityReductionScale	Determines the extent to which Scream can reduce voice allocation priority based on gain. Range: 0.0 to 1.0. A value of 1.0 specifies maximum gain-based voice priority reduction. A value of 0.0 specifies zero gain-based voice priority reduction. For more information, see “Voice Prioritization” in “Scream Sounds and Synthesizer Voices” in the “System Overview” chapter of the *Scream Library Overview*.
adsr1	Note-on portion of a Stream-specific gain envelope. Currently, must be set to `SCE_SCREAM_SND_SS_START_ADSR1_DEFAULT`.
adsr2	Note-off portion of a Stream-specific gain envelope. Currently, must be set to `SCE_SCREAM_SND_SS_START_ADSR2_DEFAULT`.
bitstreamParams	A `SceScreamSndBitstreamParams` structure specifying Bitstream parameter values.
layerParams	A `SceScreamSndBitstreamParams` structure specifying Layer parameter values.
soundParams	A Scream Library `SceScreamSoundParams` structure specifying Sound-specific parameter values. See the *Scream Library Reference* documents for details on this structure.

Description

A data structure used to store parameter values used when initializing a Stream. It is used by all the Stream starting functions: `sceScreamStartStream()`, `sceScreamStartStreamByFileToken()`, and `sceScreamStartStreamFromTransition()`.

Notes

In Scream, voice allocation priority is scaled according to gain. That is, if two sounds share the same priority value, but one is louder than the other, the louder voice is given a higher priority, making it less susceptible to voice stealing. The *`priorityReductionScale`* member scales the extent to which Scream can reduce voice allocation priority based on gain.

Setting the `SCE_SCREAM_SND_SS_START_VOICE_NO_STEAL` *`flags`* option has no bearing on the allocation of voices for the Stream. This option simply prevents the Stream's voices from being stolen for subsequent Scream or Sndstream voice requests.

See Also

`sceScreamStartStream()`, `sceScreamStartStreamByFileToken()`, `sceScreamStartStreamFromTransition()`, `SceScreamSndBitstreamParams`

SceScreamSndStreamParseParams

Stores parameter values used when parsing a Stream file.

Definition

```
struct SceScreamSndStreamParseParams {
    const char *file;
    SceScreamSndFileInterface *fileInterface;
    uint32_t flags;
    int8_t reserved1[4];
    uint64_t seekOffset;
    SceScreamSndStreamUserContext userContext;
    int8_t reserved2[4];
};
```

Members

file	ASCIIZ string pointer to the fully qualified path of the Stream file. For example, `mydir/mysubdir/mysound.wav`. Maximum length: `SCE_SCREAM_SND_STREAM_MAX_PATH`.
fileInterface	A pointer to a `SceScreamSndFileInterface` structure specifying the file interface to use for a Stream file. If `NULL`, the default file interface is used.
flags	If the file has an associated MIDI file, use `SCE_SCREAM_SND_SS_FILE_HAS_MIDI_FILE`. This is currently the only valid flag.
reserved1	For internal use only.
seekOffset	Offset number of bytes into the file at which point to start reading. Used in cases where the streamed audio file is embedded in a larger container file. Points to the location of the beginning of the header of an embedded audio file. If the audio file is not embedded, set to zero.
userContext	User defined data that is passed to the file I/O interfaces.
reserved2	For internal use only.

Description

This structure stores parameter values related to parsing the header of an audio file. You initialize this structure with appropriate values for the `sceScreamParseStreamFile()` function's *parseParams* parameter. Note that this structure has some of the same members as `SceScreamSndFileParams` and `SceScreamSndFileParams`.

Notes

The *fileInterface* member allows file I/O specification on a per file basis.

See Also

`sceScreamParseStreamFile()`, `sceScreamGetFileTokenFromStorage()`

SceScreamSndStreamPlatformInit

Stores the platform-specific parameter values required for initializing Sndstream.

Definition

```
struct SceScreamSndStreamPlatformInit {
    int32_t size;
    int32_t streaming_thread_priority;
    int32_t streaming_thread_affinity;
    int32_t parsing_thread_priority;
    int32_t parsing_thread_affinity;
    uint32_t midiBufferCount;
    uint32_t midiBufferSize;
    uint32_t flags;
    uint32_t parsedFileCount;
    uint32_t subBufferCount;
    uint32_t parsingThreadStackSize;
    uint32_t extraStreamsForStealing;
};
```

Members

size	Size of the structure in bytes. Must be correctly initialized by the application. For guidance on setting this member, see "Setting `SceScreamSndStreamPlatformInit` Structure Members" in the "Configuration, Initialization, and Shutdown" chapter of the *Sndstream Library Overview*.
streaming_thread_priority	Priority to use when creating a streaming thread. The higher the specified value, the higher the thread priority. Defaults to the synthesizer-specific Scream constant `SCE_SCREAM_SND_DEFAULT_THREAD_PRIORITY` + 1. See "Setting `SceScreamSndStreamPlatformInit` Structure Members" in the "Configuration, Initialization, and Shutdown" chapter of the *Sndstream Library Overview* for a discussion of priority.
streaming_thread_affinity	Processor affinity to use when creating a streaming thread. To specify a particular CPU core, use a zero-based index. To specify all available CPU cores, use -1. Defaults to the synthesizer-specific Scream constant `SCE_SCREAM_SND_DEFAULT_THREAD_AFFINITY`.
parsing_thread_priority	Priority to use when creating a stream parsing thread. The higher the specified value, the higher the thread priority. Defaults to the synthesizer-specific Scream constant `SCE_SCREAM_SND_DEFAULT_THREAD_PRIORITY` + 1. See "Setting `SceScreamSndStreamPlatformInit` Structure Members" in the "Configuration, Initialization, and Shutdown" chapter of the *Sndstream Library Overview* for a discussion of priority.
parsing_thread_affinity	Processor affinity to use when creating a stream parsing thread. To specify a particular CPU core use a zero-based index. To specify all available CPU cores use -1. Defaults to the synthesizer-specific Scream constant `SCE_SCREAM_SND_DEFAULT_THREAD_AFFINITY`.
midiBufferCount	A count of the number of Streams that have an associated MIDI file. Defaults to zero.

midiBufferSize	The size in bytes to allocate for each MIDI buffer. Defaults to 2048 (2K).
flags	Use this member to initialize Sndstream with alternative behaviors. Defaults to zero. Use one or more of the `Initialization Flags`.
parsedFileCount	A count of the number of pre-parsed files that have memory pre-allocated for them. Defaults to zero. For more information on tokens, see the "Working with File Tokens" chapter in the *Sndstream Library Overview*.
subBufferCount	The number of sub-buffers into which to divide each stream's buffer. Defaults to `SCE_SCREAM_SND_STREAM_FILE_DEFAULT_BUFFER_IDS`. Can be zero, or in the range: 2 to `SCE_SCREAM_SND_STREAM_FILE_MAX_BUFFER_IDS`. If you specify zero, an optimum value is chosen internally. File reads try to fill a sub-buffer, subject to alignment restrictions.
parsingThreadStackSize	The number of bytes to use for the file header parsing thread. Defaults to zero, in which case an optimal value is chosen internally.
extraStreamsForStealing	The number of additional internal structures to allocate for Stream stealing. Range: zero to the value passed to the `sceScreamInitStreaming()` function's *handleCount* parameter. Defaults to zero. See "Notes" below.

Description

This data structure contains values for all parameters required to initialize Sndstream. It is used when calling `sceScreamInitStreaming()`, and can be initialized with default values by calling `sceScreamFillDefaultPlatformInitArgs()`.

For a detailed discussion of starting a Stream, see the "Configuration, Initialization, and Shutdown" chapter of the *Sndstream Library Overview*.

Notes

Stream stealing is based on priorities specified in Bank contents or in the `SceScreamSndStartParams.priority` member when starting a Stream. Stream stealing involves starting a new Stream while a stolen Stream finishes, and therefore requires buffer management as well as voice management. For this reason, an additional internal structure is needed for every Bitstream being stolen. The memory cost per additional internal structure is only a few hundred bytes.

See Also

`sceScreamFillDefaultPlatformInitArgs()`, `sceScreamInitStreaming()`

SceScreamSndStreamQueueParams

Stores playback information used when starting a new Stream or queuing a file to an existing Stream.

Definition

```
struct SceScreamSndStreamQueueParams {
    int32_t loopCount;
    float startSecond;
};
```

Members

loopCount	Specifies a number of additional loops to play. That is, 0 to play once without looping, 1 to play twice, 2 to play 3 times, and so on. To loop indefinitely, use the `SCE_SCREAM_SND_SS_LOOP_INFINITE` constant; to loop until another file has been queued on the handle, use `SCE_SCREAM_SND_SS_LOOP_TILL_QUEUED`. See "Notes" below.
startSecond	Offset number of seconds into the file at which point to start playback. Used in cases where the desired audio data starts other than at the beginning of the specified file. Expressed in seconds. **Note:** applies to WAV and VAG file formats only.

Description

This structure stores playback information used when starting a new Stream by reference to a file token with the `sceScreamStartStreamByFileToken()` function or queuing a file to an existing Stream by reference to a file token with the `sceScreamQueueToStreamByFileToken()` function. You initialize this structure with appropriate values and use it in `sceScreamStartStreamByFileToken()` or `sceScreamQueueToStreamByFileToken()` function's *queueParams* parameter.

Notes

Alternatively, you can queue a Stream, referencing the file by path, using the `sceScreamQueueToStream()` function. And you can start a Stream from a file, referencing the file by a token, using the `sceScreamStartStreamByFileToken()` function.

Sndstream recognizes loop points embedded in ATRAC9™ Stream files. And if found, instead of looping around the entire file from start to end, looping playback takes place between the loop points. For further details, see "Working with Embedded Loop Points in Stream Files" in the "Starting a Stream" chapter of the *Sndstream Library Overview*.

See Also

`sceScreamStartStreamByFileToken()`, `sceScreamQueueToStreamByFileToken()`

SceScreamSndSyncParams

Stores synchronization properties for Stream transitions or synchronized play of Scream Sounds.

Definition

```
struct SceScreamSndSyncParams {
    uint32_t syncFlags;
    uint32_t syncUnit;
    uint32_t unitMultiple;
};
```

Members

syncFlags	One or both of the synchronization behavior flags: SCE_SCREAM_SND_SYNC_FLAG_START_IF_NO_CLOCK SCE_SCREAM_SND_SYNC_FLAG_START_IF_CLOCK_ENDS Alternatively, set to 0 if, in the absence of a sync clock Stream, you do not want a synchronized event to play out of synchronization.
syncUnit	One of the synchronization unit constants: SCE_SCREAM_SND_SYNC_UNIT_CONTENT SCE_SCREAM_SND_SYNC_UNIT_CLOCK SCE_SCREAM_SND_SYNC_UNIT_BEAT SCE_SCREAM_SND_SYNC_UNIT_MEASURE SCE_SCREAM_SND_SYNC_UNIT_MARKER.
unitMultiple	Defines synchronization point boundaries. A multiple of the *syncUnit* value. See “Notes” below.

Description

The `sceScreamStartStreamFromTransition()` function allows you to start a new Stream as a coordinated transition from a master Stream. You can also overlay a new Stream in synchronization with a master Stream. The `sceScreamPlaySoundSyncedByIndexEx()` and `sceScreamPlaySoundSyncedByNameEx()` functions allow you to trigger Scream Sounds in synchronization with a master Stream. This structure stores synchronization points and behaviors for use with these functions.

To use synchronization points defined in the sync clock Stream's associated MIDI file, set *syncUnit* to SCE_SCREAM_SND_SYNC_UNIT_CONTENT. Setting *syncUnit* to any of the other synchronization unit constants overrides any synchronization points defined in the sync clock Stream's associated MIDI file.

To set synchronization points from the API, you specify them in terms of a number of multiples (*unitMultiple*) of a basic synchronization unit (*syncUnit*). For example, to specify synchronization points occurring every two beats, set *syncUnit* to SCE_SCREAM_SND_SYNC_UNIT_BEAT, and *unitMultiple* to 2.

For *syncUnit* values smaller than a quarter-note, the resolution is in sync clocks (1/24th subdivisions of a quarter-note; see SCE_SCREAM_SND_SYNC_CLOCKS_PER_QUARTER). For example, an eighth-note is 12 sync clocks, a triplet-eighth-note is 8 sync clocks, and a sixteenth-note is 6 sync clocks, and so on. The sync clock macros may be of help in calculating multiples of quarter-notes and quarter-note subdivisions in terms of sync clocks. See SCE_SCREAM_SND_UNIT_CLOCK_MULTIPLE_QUARTER_NOTE, and so on.

Notes

If *syncUnit* is set to SCE_SCREAM_SND_SYNC_UNIT_CONTENT, *unitMultiple* is ignored.

See Also

sceScreamStartStreamFromTransition(), sceScreamPlaySoundSyncedByIndexEx(), sceScreamPlaySoundSyncedByNameEx()

SceScreamSndTransitionParams

Stores properties for a Stream transition.

Definition

```
struct SceScreamSndTransitionParams {
    uint32_t transitionMode;
    float fadeInTime;
    float fadeInGain;
    float fadeOutTime;
    float fadeOutGain;
};
```

Members

transitionMode	One of the Transition Mode Constants: SCE_SCREAM_SND_TRANSITION_MODE_PLAY_WITH_MASTER SCE_SCREAM_SND_TRANSITION_MODE_FADEOUT_MASTER SCE_SCREAM_SND_TRANSITION_MODE_KEYOFF_MASTER.
fadeInTime	Fade-in time of the transitioned (new) Stream. Expressed in seconds.
fadeInGain	Target gain of the transitioned (new) Stream on completion of its fade-in. Range: SCE_SCREAM_SND_MIN_GAIN to SCE_SCREAM_SND_MAX_GAIN.
fadeOutTime	Fade-out time of the existing Stream. Expressed in seconds. See "Notes" below.
fadeOutGain	Target gain of the existing Stream on completion of its fade-out. Range: SCE_SCREAM_SND_MIN_GAIN to SCE_SCREAM_SND_MAX_GAIN. See "Notes" below.

Description

The structure is used to store transition mode and fade in/out properties for a Stream transition. Used in conjunction with the sceScreamStartStreamFromTransition() function.

Notes

If *transitionMode* is *not* set to SCE_SCREAM_SND_TRANSITION_MODE_FADEOUT_MASTER, *fadeOutTime* and *fadeOutGain* are ignored.

See Also

sceScreamStartStreamFromTransition()

Function Prototypes

Summary

Prototypes for custom file I/O functions.

Member	Description
SceScreamSndStreamFileOpenFunction	Prototype for creating a custom file open function.
SceScreamSndStreamFileInfoCBFunction	Prototype for creating an optional custom file information callback function.
SceScreamSndStreamFileSeekFunction	Prototype for creating a custom file seek function.
SceScreamSndStreamFileReadFunction	Prototype for creating a custom file read function.
SceScreamSndStreamFileCloseFunction	Prototype for creating a custom file close function.
SceScreamSndStreamFileAsyncOpenFunction	Prototype for creating a custom function to open a file for subsequent asynchronous reading.
SceScreamSndStreamFileAsyncIsOpenCompleteFunction	Prototype for creating a custom function for polling asynchronous file open completion.
SceScreamSndStreamFileAsyncOpenBitstreamFunction	Prototype for creating a custom function to initialize Bitstream data for asynchronous reading.
SceScreamSndStreamFileAsyncReadFunction	Prototype for creating a custom asynchronous file read function.
SceScreamSndStreamFileAsyncIsReadCompleteFunction	Prototype for creating a custom function for polling asynchronous file read completion.
SceScreamSndStreamFileAsyncCloseBitstreamFunction	Prototype for creating a custom function to reset Bitstream data following asynchronous reading.
SceScreamSndStreamFileAsyncCloseFunction	Prototype for creating a custom function to close a file that was opened for asynchronous reading.

SceScreamSndStreamFileOpenFunction

Prototype for creating a custom file open function.

Definition

```
typedef int32_t SceScreamSndStreamFileOpenFunction(
    const char *filePath,
    SceScreamSndStreamUserFileHandle *pReturnedHandle,
    int32_t millisecondDeadline,
    int8_t priority,
    SceScreamSndStreamUserContext userContext
);
```

Arguments

filePath	(Input) Pointer to a zero terminated string containing the fully qualified file path to open.
pReturnedHandle	(Output) Pointer to a `SceScreamSndStreamUserFileHandle` variable in which to store the file handle. Used to reference the opened file.
millisecondDeadline	(Input) Remaining time in milliseconds until the valid data in the playback buffer is played out. May be zero if no data is currently playing.
priority	(Input) One of the following defined priority values. - `SCE_SCREAM_SND_STREAM_FILE_PRIORITY_OPEN_START`: The file being opened is to initiate a new playing stream, but playback has not yet begun. - `SCE_SCREAM_SND_STREAM_FILE_PRIORITY_OPEN_APPEND`: The file being opened is to be appended to a currently playing stream. - `SCE_SCREAM_SND_STREAM_FILE_PRIORITY_OPEN_MIDI`: The file being opened is a MIDI file associated with a stream file.
userContext	(Input) The *userContext* member of the `SceScreamSndFileParams` data structure, user defined data passed to the `sceScreamStartStream()`, `sceScreamQueueToStream()`, or `sceScreamStartStreamFromTransition()` functions.

Return Values

If the file open operation is successful, the custom function should return `SCE_SCREAM_SND_STREAM_FILE_ERROR_OK`. If an error occurred, the custom function should return `SCE_SCREAM_SND_STREAM_FILE_ERROR_OPEN`.

Description

`SceScreamSndStreamFileOpenFunction()` is a prototype for creating a custom file open function. The custom function should synchronously open a file and return a valid file handle stored at the address specified by *pReturnedHandle*. The file handle can then be used with other file I/O functions.

Notes

If an error occurred, the value of `SCE_SCREAM_SND_STREAM_FILE_INVALID_HANDLE` should be stored in the *pReturnedHandle* parameter.

See Also

SceScreamSndFileInterface, sceScreamSetDefaultFileInterface(), SceScreamSndStreamFileInfoCBFunction(), SceScreamSndStreamFileSeekFunction(), SceScreamSndStreamFileReadFunction(), SceScreamSndStreamFileCloseFunction(), SceScreamSndStreamFileAsyncOpenFunction()

SceScreamSndStreamFileInfoCBFunction

Prototype for creating an optional custom file information callback function.

Definition

```
typedef void SceScreamSndStreamFileInfoCBFunction(
    SceScreamSndStreamUserFileHandle fileHandle,
    uint32_t dataRate,
    int32_t loopCount
);
```

Arguments

fileHandle	(Output) Handle reference to the Stream file. Stored in an output variable specified by the `SceScreamSndStreamFileOpenFunction()` function's *pReturnedHandle* parameter.
dataRate	(Output) A `uint32_t` variable in which to store the data rate of the file, in bits-per-second. This value is zero for VBR-encoded MP3 streams.
loopCount	(Output) A `uint32_t` variable in which to store looping information, as set in the `SceScreamSndFileParams` *loopCount* member.

Return Values

None

Description

`SceScreamSndStreamFileInfoCBFunction()` is a prototype for creating an optional custom file information callback function. If specified in the `SceScreamSndFileInterface` *SceScreamSndStreamFileInfoCBFunction* member, the custom file information callback function is called once per Stream file, immediately after it is opened.

See Also

`SceScreamSndFileInterface`, `sceScreamSetDefaultFileInterface()`, `SceScreamSndStreamFileOpenFunction()`, `SceScreamSndStreamFileSeekFunction()`, `SceScreamSndStreamFileReadFunction()`, `SceScreamSndStreamFileCloseFunction()`

SceScreamSndStreamFileSeekFunction

Prototype for creating a custom file seek function.

Definition

```
typedef int64_t SceScreamSndStreamFileSeekFunction(
    SceScreamSndStreamUserFileHandle fileHandle,
    int64_t offset,
    uint32_t whence,
    SceScreamSndStreamUserContext userContext
);
```

Arguments

fileHandle	(Input) Handle reference to the Stream file. Stored in an output variable specified by the SceScreamSndStreamFileOpenFunction() function's *pReturnedHandle* parameter.
offset	(Input) Count of bytes to seek over.
whence	(Input) Starting point of the seek operation. Must be one of the following constants: SCE_SCREAM_SND_STREAM_FILE_SEEK_SET, SCE_SCREAM_SND_STREAM_FILE_SEEK_CUR, or SCE_SCREAM_SND_STREAM_FILE_SEEK_END.
userContext	(Input) The *userContext* member of the SceScreamSndFileParams data structure (user defined data) passed to the sceScreamStartStream(), sceScreamQueueToStream(), or sceScreamStartStreamFromTransition() functions.

Return Values

If the file seek operation is successful, the custom function should return the current position, in bytes from the beginning of the file. If an error occurred, the function should return SCE_SCREAM_SND_STREAM_FILE_ERROR_SEEK.

Description

SceScreamSndStreamFileSeekFunction() is a prototype for creating a custom file seek function. The custom function should synchronously seek in an open a file, and almost immediately return either the new current location in the file or, if an error occurred, a value less than zero.

See Also

SceScreamSndFileInterface, sceScreamSetDefaultFileInterface(), SceScreamSndStreamFileOpenFunction(), SceScreamSndStreamFileInfoCBFunction(), SceScreamSndStreamFileReadFunction(), SceScreamSndStreamFileCloseFunction()

SceScreamSndStreamFileReadFunction

Prototype for creating a custom file read function.

Definition

```
typedef int32_t SceScreamSndStreamFileReadFunction(
    SceScreamSndStreamUserFileHandle fileHandle,
    void *pDestBuffer,
    uint32_t sizeToRead,
    uint32_t *pReturnedSizeRead,
    int32_t millisecondDeadline,
    int8_t priority,
    SceScreamSndStreamUserContext userContext
);
```

Arguments

fileHandle	(Input) Handle reference to the Stream file. Stored in an output variable specified by the `SceScreamSndStreamFileOpenFunction()` function's *pReturnedHandle* parameter.
pDestBuffer	(Output) Pointer to a buffer in which to receive the data.
sizeToRead	(Input) Number of bytes to read from the file.
pReturnedSizeRead	(Output) Pointer to a `uint32_t` variable in which to receive the actual count of bytes read. May be `NULL`.
millisecondDeadline	(Input) Remaining time in milliseconds until the valid data in the playback buffer is played out. May be zero if no data is currently playing.
priority	(Input) One of the following defined constants values. - `SCE_SCREAM_SND_STREAM_FILE_PRIORITY_READ_PARSE`: The purpose of the file read is to parse the header information of the file. **Note:** The *millisecondDeadline* parameter is not valid when this priority is used, and defaults to zero. - `SCE_SCREAM_SND_STREAM_FILE_PRIORITY_READ_DATA_FILL`: The purpose of the file read is to initially fill the playback buffer. **Note:** The *millisecondDeadline* parameter is not valid when this priority is used, and defaults to zero. - `SCE_SCREAM_SND_STREAM_FILE_PRIORITY_READ_DATA_APPEND`: The purpose of the file read is to add data to the playback buffer. - `SCE_SCREAM_SND_STREAM_FILE_PRIORITY_READ_MIDI`: The purpose of the file read is to load into memory a MIDI file associated with a stream file. **Note**: The *millisecondDeadline* parameter is not valid when this priority is used, and defaults to zero.
userContext	(Input) The *userContext* member of the `SceScreamSndFileParams` data structure, user defined data passed to the `sceScreamStartStream()`, `sceScreamQueueToStream()`, or `sceScreamStartStreamFromTransition()` functions.

Return Values

If the file read operation is successful, the custom function should return `SCE_SCREAM_SND_STREAM_FILE_ERROR_OK`. If an error occurred, the custom function should return `SCE_SCREAM_SND_STREAM_FILE_ERROR_READ`. If the file I/O system decided to pass on (that is, to omit) the read request, the custom function should return `SCE_SCREAM_SND_STREAM_FILE_ERROR_PASS`. In this case, Sndstream reiterates the read request in the next processing pass. **Note:** This return code can only be used if the `priority` parameter is set to `SCE_SCREAM_SND_STREAM_FILE_PRIORITY_READ_DATA_APPEND`.

Description

`SceScreamSndStreamFileReadFunction()` is a prototype for creating a custom file read function. The custom function should synchronously read from an open file, and return the size of the data actually read into the buffer referenced by the supplied pointer.

See Also

`SceScreamSndFileInterface`, `sceScreamSetDefaultFileInterface()`, `SceScreamSndStreamFileOpenFunction()`, `SceScreamSndStreamFileInfoCBFunction()`, `SceScreamSndStreamFileSeekFunction()`, `SceScreamSndStreamFileCloseFunction()`

SceScreamSndStreamFileCloseFunction

Prototype for creating a custom file close function.

Definition

```
typedef int32_t SceScreamSndStreamFileCloseFunction(
    SceScreamSndStreamUserFileHandle fileHandle,
    SceScreamSndStreamUserContext userContext
);
```

Arguments

fileHandle	(Input) Handle reference to the Stream file. Stored in an output variable specified by the `SceScreamSndStreamFileOpenFunction()` function's *pReturnedHandle* parameter.
userContext	(Input) The *userContext* member of the `SceScreamSndFileParams` data structure, user defined data passed to the `sceScreamStartStream()`, `sceScreamQueueToStream()`, or `sceScreamStartStreamFromTransition()` functions.

Return Values

If the file close operation is successful, the custom function should return `SCE_SCREAM_SND_STREAM_FILE_ERROR_OK`. If an error occurred, the custom function should return `SCE_SCREAM_SND_STREAM_FILE_ERROR_CLOSE`.

Description

`SceScreamSndStreamFileCloseFunction()` is a prototype for creating a custom file close function. The custom function should synchronously close a file, and return almost instantly.

See Also

`SceScreamSndFileInterface`, `sceScreamSetDefaultFileInterface()`, `SceScreamSndStreamFileOpenFunction()`, `SceScreamSndStreamFileInfoCBFunction()`, `SceScreamSndStreamFileSeekFunction()`, `SceScreamSndStreamFileReadFunction()`

SceScreamSndStreamFileAsyncOpenFunction

Prototype for creating a custom function to open a file for subsequent asynchronous reading.

Definition

```
typedef int32_t SceScreamSndStreamFileAsyncOpenFunction(
    const char *filePath,
    SceScreamSndStreamUserFileAsyncHandle *pReturnedAsyncHandle,
    int32_t millisecondDeadline,
    int8_t priority,
    SceScreamSndStreamUserContext userContext,
    int32_t bitstreamId
);
```

Arguments

filePath	(Input) Pointer to a zero terminated string containing the fully qualified file path to open.
pReturnedAsyncHandle	(Output) Pointer to a `SceScreamSndStreamUserFileAsyncHandle` variable in which to store the asynchronous file handle. Used to reference the opened file (see "Notes" below). The value stored by your custom function can either be a real handle or just a pointer to a structure being used to track files opened for asynchronous reading. If an error occurred, your custom function should store `SCE_SCREAM_SND_STREAM_FILE_INVALID_HANDLE` in the *pReturnedAsyncHandle* output variable.
millisecondDeadline	(Input) Remaining time in milliseconds until the valid data in the playback buffer is played out. May be zero if no data is currently playing.
priority	(Input) One of the following defined priority values. - `SCE_SCREAM_SND_STREAM_FILE_PRIORITY_OPEN_START`: The file being opened is to initiate a new playing stream, but playback has not yet begun. - `SCE_SCREAM_SND_STREAM_FILE_PRIORITY_OPEN_APPEND`: The file being opened is to be appended to a currently playing stream. - `SCE_SCREAM_SND_STREAM_FILE_PRIORITY_OPEN_MIDI`: The file being opened is a MIDI file associated with a stream file.
userContext	(Input) The *userContext* member of the `SceScreamSndFileParams` data structure, user defined data passed to the `sceScreamStartStream()`, `sceScreamQueueToStream()`, or `sceScreamStartStreamFromTransition()` functions.
bitstreamId	(Input) Zero-based index of the target Bitstream. Range: 0 to (*handleCount* – 1), where *handleCount* is a value set when initializing Sndstream, as an argument to the `sceScreamInitStreaming()` function. See "Notes" below.

Return Values

Value	Description
`SCE_SCREAM_SND_STREAM_FILE_OPEN_COMPLETE`	The file open operation is complete
`SCE_SCREAM_SND_STREAM_FILE_OPEN_PENDING`	The file open operation is still pending
`SCE_SCREAM_SND_STREAM_FILE_ERROR_OPEN`	An error occurred

Description

`SceScreamSndStreamFileAsyncOpenFunction()` is a prototype for creating a custom function to open a file for subsequent asynchronous reading. The custom function itself should operate synchronously, opening a file for asynchronous I/O, and upon return storing a valid `SceScreamSndStreamUserFileAsyncHandle` at the address specified by *`pReturnedAsyncHandle`*. The `SceScreamSndStreamUserFileAsyncHandle` is then used with other asynchronous file I/O functions.

Notes

For efficiency purposes when opening or closing files containing multiple Bitstreams, Sndstream selects one *`bitstreamId`* to serve as a reference. For further details, see "Multi-Layer/Bitstream Files and the Asynchronous File I/O Functions" in the "Working with Multi-Layer Streams" chapter of the *Sndstream Library Overview* document.

The `SceScreamSndStreamUserFileAsyncHandle` returned in *`pReturnedAsyncHandle`* applies to all Bitstreams contained within the same file.

See Also

`SceScreamSndFileInterface`, `sceScreamSetDefaultFileInterface()`, `SceScreamSndStreamFileAsyncIsOpenCompleteFunction()`, `SceScreamSndStreamFileAsyncReadFunction()`, `SceScreamSndStreamFileAsyncIsReadCompleteFunction()`, `SceScreamSndStreamFileAsyncCloseFunction()`, `SceScreamSndStreamFileAsyncOpenBitstreamFunction()`

SceScreamSndStreamFileAsyncIsOpenComplete Function

Prototype for creating a custom function for polling asynchronous file open completion.

Definition

```
typedef int32_t SceScreamSndStreamFileAsyncIsOpenCompleteFunction(
    SceScreamSndStreamUserFileAsyncHandle asyncFileHandle,
    SceScreamSndStreamUserContext userContext,
    int32_t bitstreamId
);
```

Arguments

asyncFileHandle	(Input) Asynchronous file handle to query. Stored by the `SceScreamSndStreamFileAsyncOpenFunction()` function in an output variable specified by the *pReturnedAsyncHandle* argument.
userContext	(Input) The *userContext* member of the `SceScreamSndStreamParseParams` structure, which is user supplied data passed to the `sceScreamStartStream()`, `sceScreamQueueToStream()`, or `sceScreamStartStreamFromTransition()` functions.
bitstreamId	(Input) Zero-based index of the target Bitstream. Range: 0 to (*handleCount* – 1), where *handleCount* is a value set when initializing Sndstream, as an argument to the `sceScreamInitStreaming()` function. See “Notes” below.

Return Values

Value	Description
`SCE_SCREAM_SND_STREAM_FILE_OPEN_COMPLETE`	The file open operation is complete
`SCE_SCREAM_SND_STREAM_FILE_OPEN_PENDING`	The file open operation is still pending
`SCE_SCREAM_SND_STREAM_FILE_ERROR_OPEN`	An error occurred

Description

`SceScreamSndStreamFileAsyncIsOpenCompleteFunction()` is a prototype for creating a custom function for polling asynchronous file open completion. If the custom `SceScreamSndStreamFileAsyncOpenFunction()` function returns `SCE_SCREAM_SND_STREAM_FILE_OPEN_PENDING`, Sndstream polls for completion of the asynchronous file open operation by calling this custom function.

Notes

For efficiency purposes when opening or closing files containing multiple Bitstreams, Sndstream selects one *bitstreamId* to serve as a reference. For further details, see “Multi-Layer/Bitstream Files and the Asynchronous File I/O Functions” in the “Working with Multi-Layer Streams” chapter of the *Sndstream Library Overview*.

See Also

`SceScreamSndFileInterface`, `sceScreamSetDefaultFileInterface()`, `SceScreamSndStreamFileAsyncOpenFunction()`, `SceScreamSndStreamFileAsyncReadFunction()`, `SceScreamSndStreamFileAsyncCloseFunction()`

SceScreamSndStreamFileAsyncOpenBitstream Function

Prototype for creating a custom function to initialize Bitstream data for asynchronous reading.

Definition

```
typedef int32_t SceScreamSndStreamFileAsyncOpenBitstreamFunction(
    SceScreamSndStreamUserFileAsyncHandle asyncFileHandle,
    SceScreamSndStreamUserContext userContext,
    int32_t bitstreamId
);
```

Arguments

asyncFileHandle	(Input) Handle reference to the Stream file. Stored in an output variable specified by the `SceScreamSndStreamFileAsyncOpenFunction()` function's *pReturnedAsyncHandle* parameter.
userContext	(Input) The *userContext* member of the `SceScreamSndFileParams` data structure, user defined data passed to the `sceScreamStartStream()`, `sceScreamQueueToStream()`, or `sceScreamStartStreamFromTransition()` functions.
bitstreamId	(Input) Zero-based index of the target Bitstream. Range: 0 to (*handleCount* – 1), where *handleCount* is a value set when initializing Sndstream, as an argument to the `sceScreamInitStreaming()` function. See “Notes” below.

Return Values

If the open operation is successful, the custom function should return `SCE_SCREAM_SND_STREAM_FILE_ERROR_OK`. If an error occurred, the custom function should return `SCE_SCREAM_SND_STREAM_FILE_ERROR_OPEN`.

Description

`SceScreamSndStreamFileAsyncOpenBitstreamFunction()` is a prototype for creating a custom function to initialize Bitstream data for asynchronous reading. The custom function itself should operate synchronously.

Notes

For efficiency purposes when opening or closing files containing multiple Bitstreams, Sndstream selects one *bitstreamId* to serve as a reference. Sndstream calls `SceScreamSndStreamFileAsyncOpenBitstreamFunction()` however, in respect of all Bitstreams in a file, in preparation for reading Bitstream data. For further details, see “Multi-Layer/Bitstream Files and the Asynchronous File I/O Functions” in the “Working with Multi-Layer Streams” chapter of the *Sndstream Library Overview*.

See Also

`SceScreamSndFileInterface`, `sceScreamSetDefaultFileInterface()`, `SceScreamSndStreamFileAsyncOpenFunction()`, `SceScreamSndStreamFileAsyncIsOpenCompleteFunction()`, `SceScreamSndStreamFileAsyncReadFunction()`, `SceScreamSndStreamFileAsyncIsReadCompleteFunction()`,

SceScreamSndStreamFileAsyncCloseBitstreamFunction(),
SceScreamSndStreamFileAsyncCloseFunction()

SceScreamSndStreamFileAsyncReadFunction

Prototype for creating a custom asynchronous file read function.

Definition

```
typedef int32_t SceScreamSndStreamFileAsyncReadFunction(
    SceScreamSndStreamUserFileAsyncHandle asyncFileHandle,
    void *pDestBuffer,
    uint32_t sizeToRead,
    int64_t offset,
    int32_t millisecondDeadline,
    int8_t priority,
    SceScreamSndStreamUserContext userContext,
    int32_t bitstreamId,
    int32_t bufferId
);
```

Arguments

asyncFileHandle	(Input) Handle reference to the Stream file. Stored in an output variable specified by the `SceScreamSndStreamFileAsyncOpenFunction()` function's *pReturnedAsyncHandle* parameter.
pDestBuffer	(Output) Pointer to a buffer in which to receive the data.
sizeToRead	(Input) Data size, in bytes, to read from the Stream file. **Note:** While Sndstream should not attempt to read beyond the end of a Stream file, SCE recommends that a custom `SceScreamSndStreamFileAsyncReadFunction()` function validates the *sizeToRead* argument to ensure it remains within the bounds of available data. And if not, the function should return an error.
offset	(Input) Offset number of bytes into the Stream file at which point Sndstream begins reading data.
millisecondDeadline	(Input) Remaining time in milliseconds until the valid data in the playback buffer is played out. May be zero if no data is currently playing.
priority	(Input) One of the following defined constants values. - `SCE_SCREAM_SND_STREAM_FILE_PRIORITY_READ_PARSE`: The purpose of the file read is to parse the header information of the file. **Note:** The *millisecondDeadline* parameter is not valid when this priority is used, and defaults to zero. - `SCE_SCREAM_SND_STREAM_FILE_PRIORITY_READ_DATA_FILL`: The purpose of the file read is to initially fill the playback buffer. **Note:** The *millisecondDeadline* parameter is not valid when this priority is used, and defaults to zero. - `SCE_SCREAM_SND_STREAM_FILE_PRIORITY_READ_DATA_APPEND`: The purpose of the file read is to add data to the playback buffer. - `SCE_SCREAM_SND_STREAM_FILE_PRIORITY_READ_MIDI`: The purpose of the file read is to load into memory a MIDI file associated with a stream file. **Note**: The *millisecondDeadline* parameter is not valid when this priority is used, and defaults to zero.
userContext	(Input) The *userContext* member of the `SceScreamSndFileParams` data structure, user defined data passed to the `sceScreamStartStream()`, `sceScreamQueueToStream()`, or `sceScreamStartStreamFromTransition()` functions.

bitstreamId	(Input) Zero-based index of the target Bitstream. Range: 0 to (*handleCount* – 1), where *handleCount* is a value set when initializing Sndstream, as an argument to the sceScreamInitStreaming() function. See "Notes" below.
bufferId	(Input) Identifies an asynchronous read request for the specified *bitstreamId*. Sndstream can issue multiple read requests at the same time. Each request is identified by a combination of the *bitstreamId* and *bufferId* values. Range: 0 to (SCE_SCREAM_SND_STREAM_FILE_MAX_BUFFER_IDS – 1). See "Notes" below.

Return Values

If the file read operation is successful, the custom function should return SCE_SCREAM_SND_STREAM_FILE_ERROR_OK. If an error occurred, the custom function should return SCE_SCREAM_SND_STREAM_FILE_ERROR_READ. If the file I/O system decided to omit the read request, the custom function should return SCE_SCREAM_SND_STREAM_FILE_ERROR_PASS. In this case, Sndstream reiterates the read request in the next processing pass. **Note:** This return code can only be used if the *priority* parameter is set to SCE_SCREAM_SND_STREAM_FILE_PRIORITY_READ_DATA_APPEND.

Description

SceScreamSndStreamFileAsyncReadFunction() is a prototype for creating a custom asynchronous file read function. The custom function should asynchronously read from an open file.

Notes

When reading files containing multiple Bitstreams, Sndstream uses multiple *bitstreamId* values; only one of which is the same as the *bitstreamId* used for file open and close operations. For further details, see "Multi-Layer/Bitstream Files and the Asynchronous File I/O Functions" in the "Working with Multi-Layer Streams" chapter of the *Sndstream Library Overview*.

An application might use the *bufferId* value to index into an array of structures that are tracking asynchronous read requests for each Stream.

A read request is not issued for a specific *bufferId* unless a previous read operation for the same *bufferId* has completed. Sndstream polls for completion of an asynchronous read request by calling the custom SceScreamSndStreamFileAsyncIsReadCompleteFunction() function with the corresponding *bufferId* value.

See Also

SceScreamSndFileInterface, sceScreamSetDefaultFileInterface(), SceScreamSndStreamFileAsyncOpenFunction(), SceScreamSndStreamFileAsyncIsOpenCompleteFunction(), SceScreamSndStreamFileAsyncIsReadCompleteFunction(), SceScreamSndStreamFileAsyncCloseFunction()

SceScreamSndStreamFileAsyncIsReadComplete Function

Prototype for creating a custom function for polling asynchronous file read completion.

Definition

```
typedef int32_t SceScreamSndStreamFileAsyncIsReadCompleteFunction(
    SceScreamSndStreamUserFileAsyncHandle asyncFileHandle,
    SceScreamSndStreamUserContext userContext,
    int32_t bitstreamId,
    int32_t bufferId
);
```

Arguments

asyncFileHandle	(Input) Handle reference to the Stream file. Stored in an output variable specified by the `SceScreamSndStreamFileAsyncOpenFunction()` function's *pReturnedAsyncHandle* parameter.
userContext	(Input) The *userContext* member of the `SceScreamSndStreamParseParams` structure. User defined data passed to the `sceScreamStartStream()`, `sceScreamQueueToStream()`, or `sceScreamStartStreamFromTransition()` functions.
bitstreamId	(Input) Zero-based index of the target Bitstream. Range: 0 to (*handleCount* – 1), where *handleCount* is a value set when initializing Sndstream, as an argument to the `sceScreamInitStreaming()` function. See "Notes" below.
bufferId	(Input) Identifies an asynchronous read request for the specified *bufferId*. Range: 0 to (`SCE_SCREAM_SND_STREAM_FILE_MAX_BUFFER_IDS` – 1).

Return Values

Returns a positive value if the file read operation is complete. Returns 0 if the read operation is still pending. Otherwise, returns a negative value if an error occurred. For example, return `SCE_SCREAM_SND_STREAM_FILE_ERROR_READ` if the full amount of requested data cannot be read.

Description

`SceScreamSndStreamFileAsyncIsReadCompleteFunction()` is a prototype for creating a custom function for polling asynchronous file read completion. Sndstream polls for completion of an asynchronous read request by calling this custom function. A new read request is not issued for a given *bufferId* until any previous read operation for that *bufferId* is complete.

Notes

When reading files containing multiple Bitstreams, Sndstream uses multiple *bitstreamId* values; only one of which is the same as the *bitstreamId* used for file open and close operations. For further details, see "Multi-Layer/Bitstream Files and the Asynchronous File I/O Functions" in the "Working with Multi-Layer Streams" chapter of the *Sndstream Library Overview*.

A possible implementation of the asynchronous file I/O functions might create an array of structures for tracking asynchronous read requests. The *bufferId* value could then serve is an index into such an array.

See Also

SceScreamSndFileInterface, sceScreamSetDefaultFileInterface(), SceScreamSndStreamFileAsyncOpenFunction(), SceScreamSndStreamFileAsyncIsOpenCompleteFunction(), SceScreamSndStreamFileAsyncReadFunction(), SceScreamSndStreamFileAsyncCloseFunction()

SceScreamSndStreamFileAsyncCloseBitstreamFunction

Prototype for creating a custom function to reset Bitstream data following asynchronous reading.

Definition

```
typedef int32_t SceScreamSndStreamFileAsyncCloseBitstreamFunction(
    SceScreamSndStreamUserFileAsyncHandle asyncFileHandle,
    SceScreamSndStreamUserContext userContext,
    int32_t bitstreamId
);
```

Arguments

asyncFileHandle	(Input) Handle reference to the Stream file. Stored in an output variable specified by the `SceScreamSndStreamFileAsyncOpenFunction()` function's *pReturnedAsyncHandle* parameter.
userContext	(Input) The *userContext* member of the `SceScreamSndFileParams` data structure, user defined data passed to the `sceScreamStartStream()`, `sceScreamQueueToStream()`, or `sceScreamStartStreamFromTransition()` functions.
bitstreamId	(Input) Zero-based index of the target Bitstream. Range: 0 to (*handleCount* – 1), where *handleCount* is a value set when initializing Sndstream, as an argument to the `sceScreamInitStreaming()` function. See “Notes” below.

Return Values

If the close operation is successful, the custom function should return `SCE_SCREAM_SND_STREAM_FILE_ERROR_OK`. If an error occurred, the custom function should return `SCE_SCREAM_SND_STREAM_FILE_ERROR_CLOSE`.

Description

`SceScreamSndStreamFileAsyncCloseBitstreamFunction()` is a prototype for creating a custom function to reset Bitstream data following asynchronous reading. The custom function should operate synchronously, and should not wait for any pending reads to complete.

Notes

For efficiency purposes when opening or closing files containing multiple Bitstreams, Sndstream selects one *bitstreamId* to serve as a reference. Sndstream calls `SceScreamSndStreamFileAsyncCloseBitstreamFunction()`, however, in respect of all Bitstreams in a file, to reset the Bitstreams' data after reading. By the time Sndstream calls this function, any data read operations should have completed. This mechanism allows the application to verify the status of pending data read operations. For further details, see “Multi-Layer/Bitstream Files and the Asynchronous File I/O Functions” in the “Working with Multi-Layer Streams” chapter of the *Sndstream Library Overview* document.

See Also

SceScreamSndFileInterface, sceScreamSetDefaultFileInterface(),
SceScreamSndStreamFileAsyncOpenFunction(),
SceScreamSndStreamFileAsyncIsOpenCompleteFunction(),
SceScreamSndStreamFileAsyncReadFunction(),
SceScreamSndStreamFileAsyncIsReadCompleteFunction(),
SceScreamSndStreamFileAsyncCloseFunction(),
SceScreamSndStreamFileAsyncOpenBitstreamFunction()

SceScreamSndStreamFileAsyncCloseFunction

Prototype for creating a custom function to close a file that was opened for asynchronous reading.

Definition

```
typedef int32_t SceScreamSndStreamFileAsyncCloseFunction(
    SceScreamSndStreamUserFileAsyncHandle asyncFileHandle,
    SceScreamSndStreamUserContext userContext,
    int32_t bitstreamId
);
```

Arguments

asyncFileHandle	(Input) Handle reference to the Stream file. Stored in an output variable specified by the `SceScreamSndStreamFileAsyncOpenFunction()` function's *pReturnedAsyncHandle* parameter.
userContext	(Input) The *userContext* member of the `SceScreamSndFileParams` data structure, user defined data passed to the `sceScreamStartStream()`, `sceScreamQueueToStream()`, or `sceScreamStartStreamFromTransition()` functions.
bitstreamId	(Input) Zero-based index of the target Bitstream. Range: 0 to (*handleCount* – 1), where *handleCount* is a value set when initializing Sndstream, as an argument to the `sceScreamInitStreaming()` function. See “Notes” below.

Return Values

If the file close operation is successful, the custom function should return `SCE_SCREAM_SND_STREAM_FILE_ERROR_OK`. If an error occurred, the custom function should return `SCE_SCREAM_SND_STREAM_FILE_ERROR_CLOSE`.

Description

`SceScreamSndStreamFileAsyncCloseFunction()` is a prototype for creating a custom function to close a file that was opened for asynchronous reading. The custom function itself should operate synchronously, closing a file that was opened for asynchronous I/O. This function should not wait for any pending reads to complete.

Notes

For efficiency purposes when opening or closing files containing multiple Bitstreams, Sndstream selects one *bitstreamId* to serve as a reference. Sndstream uses the same *bitstreamId* to close a file as was to open it. For further details, see “Multi-Layer/Bitstream Files and the Asynchronous File I/O Functions” in the “Working with Multi-Layer Streams” chapter of the *Sndstream Library Overview* document.

See Also

`SceScreamSndFileInterface`, `sceScreamSetDefaultFileInterface()`, `SceScreamSndStreamFileAsyncOpenFunction()`, `SceScreamSndStreamFileAsyncIsOpenCompleteFunction()`, `SceScreamSndStreamFileAsyncReadFunction()`, `SceScreamSndStreamFileAsyncIsReadCompleteFunction()`, SceScreamSndStreamFileAsyncCloseBitstreamFunction(), SceScreamSndStreamFileAsyncOpenBitstreamFunction()

Type Definitions

Summary

The type definitions define data types for various Sndstream APIs.

Member	Description
SceScreamSndStreamFileToken	Data type for an opaque pointer to a pre-parsed Stream file.
SceScreamSndStreamFileTokenStorage	Data type for an opaque pointer to a storage for pre-parsed file tokens.
SceScreamSndStreamUserContext	Data type for an opaque pointer or integer used to reference a user context value.
SceScreamSndStreamUserFileAsyncHandle	Data type for an opaque pointer or integer used to reference an asynchronously opened Stream file.
SceScreamSndStreamUserFileHandle	Data type for an opaque pointer or integer used to reference a Stream file.

SceScreamSndStreamFileToken

Data type for an opaque pointer to a pre-parsed Stream file.

Definition

```
typedef void *SceScreamSndStreamFileToken;
```

Description

Use this type to reference a Stream file, pre-parsed using the `sceScreamParseStreamFile()` function. You can also obtain a token by calling `sceScreamGetFileTokenFromStorage()`. The constant `SCE_SCREAM_SND_STREAM_INVALID_FILE_TOKEN` designates an invalid token.

See Also

`sceScreamStartStreamByFileToken()`, `sceScreamQueueToStreamByFileToken()`, `sceScreamParseStreamFile()`, `sceScreamDeleteStreamFileToken()`, `sceScreamGetFileTokenFromStorage()`, `SCE_SCREAM_SND_STREAM_INVALID_FILE_TOKEN`

SceScreamSndStreamFileTokenStorage

Data type for an opaque pointer to a storage for pre-parsed file tokens.

Definition

```
typedef void SceScreamSndStreamFileTokenStorage;
```

Description

Use this type to reference a file token storage. You can create the storage at build time and then load it from a file to be used at run time. Create tokens by calling `sceScreamParseStreamFile()`.

See Also

`sceScreamCreateFileTokenStorage()`, `sceScreamByteReverseFileTokenStorage()`, `sceScreamDeleteFileTokenStorage()`, `sceScreamValidateFileTokenStorage()`, `sceScreamGetFileTokenFromStorage()`, `sceScreamParseStreamFile()`

SceScreamSndStreamUserContext

Data type for an opaque pointer or integer used to reference a user context value.

Definition

```
typedef void *SceScreamSndStreamUserContext;
```

Description

This type is used to reference a user context value. It is widely used to store custom user data. This data type is used in a wide variety of structures and functions. In some cases, the `SceScreamSndStreamUserContext` value is returned later in another function.

For example, the *`userContext`* member of the `SceScreamSndFileParams` structure takes a `SceScreamSndStreamUserContext` value. In turn, `SceScreamSndFileParams` is used as a parameter to the `sceScreamStartStream()` function to start a Stream. When the `sceScreamGetStreamFileLengthInSeconds()` function called on this Stream returns, its *`outContext`* parameter contains the `SceScreamSndStreamUserContext` value from the *`userContext`* member of the `SceScreamSndFileParams` structure passed to `sceScreamStartStream()`.

In other cases, the `SceScreamSndStreamUserContext` value is used in a structure and later passed to related functions. For example, the `SceScreamSndFileParams` structure also takes a `SceScreamSndFileInterface` structure that contains custom I/O function prototypes, such as `SceScreamSndStreamFileAsyncOpenFunction()`. `SceScreamSndStreamFileAsyncOpenFunction()` then takes this `SceScreamSndStreamUserContext` value in its *`userContext`* parameter.

See Also

`SceScreamSndFileParams`, `SceScreamSndStreamParseParams`, `sceScreamGetStreamFileLengthInSeconds()`, `sceScreamGetStreamFileLocationInSeconds()`, `sceScreamGetStreamFileLoopingCount()`, `sceScreamGetStreamFileSecondsRemaining()`

SceScreamSndStreamUserFileAsyncHandle

Data type for an opaque pointer or integer used to reference an asynchronously opened Stream file.

Definition

```
typedef void *SceScreamSndStreamUserFileAsyncHandle;
```

Description

This type is used to reference an asynchronously opened Stream file in the custom file I/O function prototypes.

See Also

SceScreamSndStreamUserFileHandle, SceScreamSndStreamFileAsyncOpenFunction(),
SceScreamSndStreamFileAsyncIsOpenCompleteFunction(),
SceScreamSndStreamFileAsyncReadFunction(),
SceScreamSndStreamFileAsyncIsReadCompleteFunction(),
SceScreamSndStreamFileAsyncCloseFunction(),
SceScreamSndStreamFileAsyncOpenBitstreamFunction(),
SceScreamSndStreamFileAsyncCloseBitstreamFunction()

SceScreamSndStreamUserFileHandle

Data type for an opaque pointer or integer used to reference a Stream file.

Definition

```
typedef void *SceScreamSndStreamUserFileHandle;
```

Description

This type is used to reference a Stream file in the custom file I/O function prototypes.

See Also

SceScreamSndStreamUserFileAsyncHandle, SceScreamSndStreamFileOpenFunction(), SceScreamSndStreamFileInfoCBFunction(), SceScreamSndStreamFileSeekFunction(), SceScreamSndStreamFileReadFunction(), SceScreamSndStreamFileCloseFunction()

System Functions

Summary

The system functions initialize and shutdown Sndstream.

Function	Description
sceScreamCloseStreaming	Shuts down Sndstream.
sceScreamFillDefaultPlatformInitArgs	Initializes a SceScreamSndStreamPlatformInit data structure for use in a call to sceScreamInitStreaming().
sceScreamInitStreaming	Initializes Sndstream for use by an application.

sceScreamCloseStreaming

Shuts down Sndstream.

Definition

```
int32_t sceScreamCloseStreaming(void);
```

Return Values

Returns 0 if Sndstream shutdown was successful. Returns `SCE_SCREAM_SND_STREAM_CLOSE_ERROR_NOT_INITED` if Sndstream was not initialized.

Description

Use this function to completely shut down Sndstream. This function stops all Streams, destroys all streaming threads, and releases all allocated memory.

Notes

This function uses `sceScreamStopAllStreams()` to stop all streaming threads, which blocks the calling thread until all Streams are stopped.

See Also

`sceScreamInitStreaming()`, `sceScreamStopAllStreams()`

sceScreamFillDefaultPlatformInitArgs

Initializes a `SceScreamSndStreamPlatformInit` data structure for use in a call to `sceScreamInitStreaming()`.

Definition

```
int32_t sceScreamFillDefaultPlatformInitArgs(
    SceScreamSndStreamPlatformInit *args
);
```

Arguments

args	(Input/Output) Pointer to a `SceScreamSndStreamPlatformInit` data structure to be initialized. See “Notes” below.

Return Values

Returns 0 if the `SceScreamSndStreamPlatformInit` data structure is successfully initialized. Returns `SCE_SCREAM_SND_STREAM_INIT_ERROR_INVALID_ARGS` if the *size* member of the specified `SceScreamSndStreamPlatformInit` structure is incorrect.

Description

Use this function to initialize a `SceScreamSndStreamPlatformInit` data structure with the following default values:

size	sizeof(`SceScreamSndStreamPlatformInit`)
thread_priority	`SCE_SCREAM_SND_DEFAULT_THREAD_PRIORITY` + 1
thread_affinity	-1
midiBufferCount	0
midiBufferSize	`SCE_SCREAM_SND_STREAM_DEFAULT_MIDI_BUFFER_SIZE`
flags	0
parsedFileCount	0
subBufferCount	4
parsingThreadStackSize	0
extraStreamsForStealing	0

Notes

You can override the default settings after initialization, as desired.

See Also

`SceScreamSndStreamPlatformInit`, `sceScreamInitStreaming()`

sceScreamInitStreaming

Initializes Sndstream for use by an application.

Definition

```
int32_t sceScreamInitStreaming(
    uint32_t handleCount,
    uint32_t bufferSize,
    const SceScreamSndStreamPlatformInit *args
);
```

Arguments

handleCount	(Input) Number of Stream handles to allocate. Set to the maximum number of simultaneously active Streams that occur in your game.
bufferSize	(Input) Size in bytes of one Stream buffer. Each Stream handle has a buffer of this size associated with it. See "Notes" below.
args	(Input) Pointer to an initialized `SceScreamSndStreamPlatformInit` structure. Call `sceScreamFillDefaultPlatformInitArgs()` to initialize this structure.

Return Values

Returns 0 if Sndstream was initialized successfully. Otherwise, returns one of the Initialization Errors; see `SCE_SCREAM_SND_STREAM_INIT_ERROR_INVALID_ARGS`, and so on.

Description

Prepares the library for streaming, instantiates threads, and allocates memory. This is the only Sndstream function that allocates memory.

For a full discussion of using this function, see the "Configuration, Initialization, and Shutdown" chapter of the *Sndstream Library Overview*.

Notes

The *handleCount* parameter must include any cross-fading Bitstreams in the overall count of simultaneously playing Bitstreams.

The specified *bufferSize* is subdivided into sub-buffers, the number of which is determined by the `SceScreamSndStreamPlatformInit` *subBufferCount* member. Sub-buffers are used to fill individual Stream data reads.

See Also

`SceScreamSndStreamPlatformInit`, `sceScreamFillDefaultPlatformInitArgs()`, `sceScreamCloseStreaming()`

Start and Stop Functions

Summary

These functions start and stop Streams.

Function	Description
sceScreamCueStreamToTime	Seeks to a time offset in the currently playing file and immediately continues playback.
sceScreamQueueToStream	Inserts a Stream file, referenced by path, into the queue of an existing Stream handle.
sceScreamQueueToStreamByFileToken	Inserts a Stream file, referenced by token, into the queue of an existing Stream handle.
sceScreamStartStream	Initializes a Stream and starts playback of a file referenced by path.
sceScreamStartStreamByFileToken	Initializes a Stream and starts playback of a file referenced by token.
sceScreamStartStreamFromTransition	Starts a new Stream as a synchronized transition from an existing Stream, or overlays a new Stream in synchronization with an existing Stream.
sceScreamStopAllStreams	Stops all currently playing Streams.
sceScreamWaitForStreamToBeDone	Waits for a Stream to complete playback.

sceScreamCueStreamToTime

Seeks to a time offset in the currently playing file and immediately continues playback.

Definition

```
int32_t sceScreamCueStreamToTime(
    uint32_t handle,
    float seconds
);
```

Arguments

handle	(Input) Handle of the Stream upon which to perform the cue operation. A value returned by the `sceScreamStartStream()`, `sceScreamQueueToStreamByFileToken()`, or `sceScreamStartStreamFromTransition()` functions.
seconds	(Input) Floating point value, in seconds, of an offset position from the beginning of the file to seek to and continue playback.

Return Values

Returns `SCE_SCREAM_SND_STREAM_ERROR_OK` if the operation was successful, otherwise returns `SCE_SCREAM_SND_STREAM_ERROR_OUT_OF_RANGE` if parameter value was out of range.

Description

This function causes the Stream to jump from the current playback position to a new playback position, specified by a time increment from the beginning of the file, and immediately continue playback.

Notes

Depending on the playback data type, the time position offset may be rounded to the nearest valid sample or compression frame starting boundary.

Audible artifacts can occur if there is a discontinuity between the current playback position and the new offset position.

See Also

`sceScreamGetStreamFileLengthInSeconds()`, `sceScreamGetStreamFileSecondsRemaining()`, `sceScreamGetStreamFileLocationInSeconds()`

sceScreamQueueToStream

Inserts a Stream file, referenced by path, into the queue of an existing Stream handle.

Definition

```
uint32_t sceScreamQueueToStream(
    uint32_t queueHandle,
    uint32_t queueIndex,
    const SceScreamSndFileParams *fileParams,
    const SceScreamSndStartParams *startParams
);
```

Arguments

queueHandle	(Input) Handle of an active Stream, into the queue of which to insert the Stream file. A value returned by the `sceScreamStartStream()`, `sceScreamStartStreamByFileToken()`, or `sceScreamStartStreamFromTransition()` functions.
queueIndex	(Input) Zero-based index indicating a position in the queue at which point to insert the Stream file. Range: `SCE_SCREAM_SND_QUEUE_INDEX_HEAD` to `SCE_SCREAM_SND_QUEUE_INDEX_TAIL`. Use `SCE_SCREAM_SND_QUEUE_INDEX_TAIL` to insert the file at the end of the queue.
fileParams	(Input) Pointer to a `SceScreamSndFileParams` structure initialized with appropriate Stream file parameter values.
startParams	(Input) Pointer to a `SceScreamSndStartParams` structure initialized with Stream parameter values to use in the event that the Stream to which you are queuing is already dead. Set to `NULL` if you do not want the new Stream to play if the parent Stream has expired. **Note:** if not `NULL`, and the parent Stream is still active, *startParams* settings are ignored.

Return Values

If successful, returns the supplied handle of the queue-inserted Stream. If not successful, returns 0.

Description

This function inserts a Stream file, referenced by path, into the queue of an active Stream handle. The *queueIndex* parameter allows you to specify an index point in the queue at which to position the new Stream file. You can obtain the number of files currently queued to an active Stream using the `sceScreamGetStreamQueueCount()` function. Specifying `SCE_SCREAM_SND_QUEUE_INDEX_HEAD` always inserts the file at the start of the queue. Specifying `SCE_SCREAM_SND_QUEUE_INDEX_TAIL` always inserts the file at the end of the queue, regardless of the current number of files in the queue.

Notes

While this function accepts specification of a Stream file by path (expressed in the *fileParams* `SceScreamSndFileParams` structure's *file* member), it automatically pre-parses the file's header to produce a file token, by which the file is actually referenced internally. As an alternative, you can explicitly create the file token (at build or load time) using the `sceScreamParseStreamFile()` function, and then use the `sceScreamQueueToStreamByFileToken()` function to insert a Stream file, referenced by token, into the queue of an existing Stream handle.

See Also

`sceScreamGetStreamQueueCount()`, `sceScreamStartStream()`,
`sceScreamQueueToStreamByFileToken()`

sceScreamQueueToStreamByFileToken

Inserts a Stream file, referenced by token, into the queue of an existing Stream handle.

Definition

```
uint32_t sceScreamQueueToStreamByFileToken(
    uint32_t queueHandle,
    uint32_t queueIndex,
    SceScreamSndStreamFileToken fileToken,
    const SceScreamSndStreamQueueParams *queueParams,
    const SceScreamSndStartParams *startParams
);
```

Arguments

queueHandle	(Input) Handle of an active Stream, into the queue of which to insert the Stream file. A value returned by the `sceScreamStartStreamByFileToken()`, `sceScreamStartStream()`, or `sceScreamStartStreamFromTransition()` functions.
queueIndex	(Input) Zero-based index indicating a position in the queue at which point to insert the Stream file. Range: `SCE_SCREAM_SND_QUEUE_INDEX_HEAD` to `SCE_SCREAM_SND_QUEUE_INDEX_TAIL`. Use `SCE_SCREAM_SND_QUEUE_INDEX_TAIL` to insert the file at the end of the queue.
fileToken	(Input) A token that represents a Stream file, containing information from the file's header. You create a Stream file token using the `sceScreamParseStreamFile()` function.
queueParams	(Input) Pointer to an appropriately initialized `SceScreamSndStreamQueueParams` structure, specifying playback information for the inserted stream.
startParams	(Input) Pointer to a `SceScreamSndStartParams` structure initialized with Stream parameter values to use in the event that the Stream to which you are queuing is already dead. Set to `NULL` if you do not want the new Stream to play if the parent Stream has expired. **Note:** if not `NULL`, and the parent Stream is still active, *startParams* settings are ignored.

Return Values

If successful, returns the supplied handle of the queue-inserted Stream. If not successful, returns 0.

Description

This function inserts a specified Stream file, referenced by token, into the queue of an active Stream handle. The *queueIndex* parameter allows you to specify an index point in the queue at which to position the new Stream file. You can obtain the number of files currently queued to an active Stream using the `sceScreamGetStreamQueueCount()` function. Specifying `SCE_SCREAM_SND_QUEUE_INDEX_HEAD` always inserts the file at the start of the queue. Specifying `SCE_SCREAM_SND_QUEUE_INDEX_TAIL` always inserts the file at the end of the queue, regardless of the current number of files in the queue.

See Also

SceScreamSndStreamQueueParams, SceScreamSndStartParams, sceScreamStartStreamByFileToken(), sceScreamParseStreamFile(), sceScreamGetStreamQueueCount(), sceScreamQueueToStream()

sceScreamStartStream

Initializes a Stream and starts playback of a file referenced by path.

Definition

```
uint32_t sceScreamStartStream(
    const SceScreamSndFileParams *fileParams,
    const SceScreamSndStartParams *startParams,
    int32_t outputDest = SCE_SCREAM_SND_OUTPUT_DEST_MASTER
);
```

Arguments

fileParams	(Input) Pointer to a SceScreamSndFileParams structure, initialized with appropriate Stream file parameter values.
startParams	(Input) Pointer to a SceScreamSndStartParams structure, initialized with parameter values for the new Stream.
outputDest	(Input) Index of an output destination. Defaults to SCE_SCREAM_SND_OUTPUT_DEST_MASTER for master output. To inherit an output destination from the Group to which the Sound is assigned, use SCE_SCREAM_SND_OUTPUT_DEST_BY_GROUP. To specify an allocated pre-master submix buss, use the number of the desired submix, indexing from zero, and within the range: SCE_SCREAM_SND_OUTPUT_DEST_PREMASTER_0 to (SCE_SCREAM_SND_MAX_PREMASTER_SUBMIXES – 1). See "Notes" below.

Return Values

If successful, returns the handle of the initialized Stream. If not successful, returns 0.

Description

This function initializes a Stream and starts playback of a file referenced by path.

The returned handle is used as input to numerous Sndstream functions that manipulate or retrieve information about a Stream. The handle can also be used as input to Scream Library functions, enabling functionality from the Scream Library API to be applied.

Notes

While this function accepts specification of a Stream file by path (expressed in the *fileParams* SceScreamSndFileParams structure's *file* member), it automatically pre-parses the file's header to produce a file token, by which the file is actually referenced internally. As an alternative, you can explicitly create the file token (at build or load time) using the `sceScreamParseStreamFile()` function, and then use the `sceScreamStartStreamByFileToken()` function to initialize a Stream and start playback of a file referenced by token.

Pre-master submix busses must be allocated at initialization time using the *numPremasterCompSubmixes* and *numPremasterScCompSubmixes* members of the Scream `SceScreamSystemParams` structure. Make sure that you do not set a pre-master submix output destination that has not been allocated.

See Also

SceScreamSndFileParams, SceScreamSndStartParams, `sceScreamQueueToStream()`, `sceScreamStartStreamFromTransition()`, `sceScreamStartStreamByFileToken()`

sceScreamStartStreamByFileToken

Initializes a Stream and starts playback of a file referenced by token.

Definition

```
uint32_t sceScreamStartStreamByFileToken(
    SceScreamSndStreamFileToken fileToken,
    const SceScreamSndStreamQueueParams *queueParams,
    const SceScreamSndStartParams *startParams,
    int32_t outputDest = SCE_SCREAM_SND_OUTPUT_DEST_MASTER
);
```

Arguments

`fileToken`	(Input) A token that represents a Stream file, containing information from the file's header. You create a Stream file token using the `sceScreamParseStreamFile()` function.
`queueParams`	(Input) Pointer to an appropriately initialized `SceScreamSndStreamQueueParams` structure, specifying playback information.
`startParams`	(Input) Pointer to a `SceScreamSndStartParams` structure, initialized with parameter values for the new Stream.
`outputDest`	(Input) Index of an output destination. Defaults to `SCE_SCREAM_SND_OUTPUT_DEST_MASTER` for master output. To inherit an output destination from the Group to which the Sound is assigned, use `SCE_SCREAM_SND_OUTPUT_DEST_BY_GROUP`. To specify an allocated pre-master submix buss, use the number of the desired submix, indexing from zero, and within the range: `SCE_SCREAM_SND_OUTPUT_DEST_PREMASTER_0` to (`SCE_SCREAM_SND_MAX_PREMASTER_SUBMIXES` – 1). See "Notes" below.

Return Values

If successful, returns the handle of the initialized Stream. If not successful, returns 0.

Description

This function initializes a Stream and starts playback of a file referenced by a token.

The returned handle is used as input to numerous Sndstream functions that manipulate or retrieve information about a Stream. The handle can also be used as input to Scream Library functions, enabling functionality from the Scream Library API to be applied.

Notes

Pre-master submix busses must be allocated at initialization time using the *`numPremasterCompSubmixes`* and *`numPremasterScCompSubmixes`* members of the Scream `SceScreamSystemParams` structure. Make sure that you do not set a pre-master submix output destination that has not been allocated.

See Also

`SceScreamSndStreamQueueParams`, `SceScreamSndStartParams`, `SceScreamSndStreamFileToken`, `sceScreamQueueToStreamByFileToken()`, `sceScreamParseStreamFile()`, `sceScreamStartStream()`, `sceScreamStartStreamFromTransition()`

sceScreamStartStreamFromTransition

Starts a new Stream as a synchronized transition from an existing Stream, or overlays a new Stream in synchronization with an existing Stream.

Definition

```
uint32_t sceScreamStartStreamFromTransition(
    uint32_t transitionHandle,
    const SceScreamSndSyncParams *syncParams,
    const SceScreamSndTransitionParams *transitionParams,
    const SceScreamSndFileParams *fileParams,
    const SceScreamSndStartParams *startParams
);
```

Arguments

transitionHandle	(Input) Handle of the existing Stream to transition from or to play along with. See "Notes" below.
syncParams	(Input) Pointer to a `SceScreamSndSyncParams` structure, initialized with appropriate synchronization properties for the new Stream.
transitionParams	(Input) Pointer to a `SceScreamSndTransitionParams` structure, initialized with appropriate transition properties for the new Stream.
fileParams	(Input) Pointer to a `SceScreamSndFileParams` structure, initialized with appropriate Stream file parameter values for the new Stream.
startParams	(Input) Pointer to a `SceScreamSndStartParams` structure, initialized with parameter values for the new Stream. See "Notes" below.

Return Values

If successful, returns the handle of the new Stream. If not successful, returns 0.

Description

This function either starts a new Stream in a synchronized transition from an existing Stream, or overlays a new Stream in synchronization with the sync clock Stream.

The returned handle is used as input to numerous Sndstream functions that manipulate or retrieve information about a Stream. You can also use the handle as input to Scream Library functions, enabling functionality from the Scream Library API to be applied.

Notes

Only one Stream transition on a given Stream handle can be pending at a time. That is, if you call this function while a previous call with the same `transitionHandle` is still pending (has not yet reached a transition point), the previous call is flushed in favor of the more recent call, and so on.

If transitioning from a Stream that is set as the current sync clock Stream, note that, at the transition point, the current sync clock Stream therefore terminates. You have the following options:

- set the transitioned Stream to become the sync clock Stream following the transition point
- continue without a sync clock Stream
- start another new Stream and set it as the sync clock Stream

To set the transitioned Stream to become the sync clock Stream, include `SCE_SCREAM_SND_SS_START_SYNC_CLOCK` in the *flags* member of the `SceScreamSndStartParams` structure pointed to in the *startParams* parameter.

To continue without a sync clock Stream, use `SCE_SCREAM_SND_SYNC_FLAG_START_IF_NO_CLOCK` in the *`syncFlags`* member of the `SceScreamSndSyncParams` structure pointed to in the *`syncParams`* parameter.

To start another new Stream and set it as the sync clock Stream, start a new Stream by calling `sceScreamStartStream()`. In this call, use `SCE_SCREAM_SND_SS_START_SYNC_CLOCK` in the *`flags`* member of the `SceScreamSndStartParams` structure pointed to by the *`startParams`* parameter.

When setting up game-interactive Stream transitions, it is sometimes programmatically possible that a Stream may ultimately transition to the file it is currently playing. If a Stream attempts to transition back to the beginning of the file it is currently playing, the transition is automatically canceled, allowing the Stream to simply continue playing its current file. This prevents unwanted discontinuities in musical intensity.

For more details, see “Starting a Stream as a Transition from or Overlay with an Existing Stream” in the “Synchronizing Stream Transitions, Overlays, and Scream Sounds” chapter in the *Sndstream Library Overview*.

See Also

`SceScreamSndFileParams`, `SceScreamSndStartParams`, `sceScreamQueueToStream()`, `sceScreamStartStream()`, `sceScreamStartStreamByFileToken()`, `SceScreamSndSyncParams`, `SceScreamSndTransitionParams`

sceScreamStopAllStreams

Stops all currently playing Streams.

Definition

```
int32_t sceScreamStopAllStreams(void);
```

Arguments

None

Return Values

Returns SCE_SCREAM_SND_STREAM_ERROR_OK.

Description

Use this function to stop all currently playing Streams. The sceScreamCloseStreaming() function calls this function.

Notes

This function blocks the calling thread until all Streams are stopped.

To stop a single Stream, call the Scream Library function sceScreamStopSound().

See Also

sceScreamWaitForStreamToBeDone(), sceScreamCloseStreaming()

sceScreamWaitForStreamToBeDone

Waits for a Stream to complete playback.

Definition

```
int32_t sceScreamWaitForStreamToBeDone(
    uint32_t handle
);
```

Arguments

handle	(Input) Handle of the Stream for which to wait for completion. A value returned by the `sceScreamStartStream()` or `sceScreamStartStreamFromTransition()` functions.

Return Values

Returns `SCE_SCREAM_SND_STREAM_ERROR_OK`.

Description

This function blocks the calling thread until the specified Stream has completed all playback. Since there may be a slight latency involved in closing a Stream, this function enables coordination of subsequent actions immediately, but not before, the completion of a specified Stream. If you call the Scream Library function `sceScreamStopSound(`*handle*`)` and then call this function with the same handle, your code blocks until the Stream has actually finished stopping.

Notes

To simply check whether a Stream is still playing, use the Scream Library function `sceScreamSoundIsStillPlaying()`, with the Stream handle as its only parameter.

This function does not issue any kind of stop command on the Stream. To stop a single Stream, call the Scream Library function `sceScreamStopSound()`.

See Also

`sceScreamStopAllStreams()`, `sceScreamCloseStreaming()`

Status Functions

Summary

The status functions retrieve information about Streams.

Function	Description
sceScreamGetStreamFileLengthInSeconds	Retrieves the total duration, in seconds, of the currently playing file on a Stream.
sceScreamGetStreamFileLocationInSeconds	Retrieves the current playback position, in seconds, of the currently playing file on a Stream.
sceScreamGetStreamFileLoopingCount	Retrieves the number of loops initially assigned and the number of completed loops of the currently playing file on a Stream.
sceScreamGetStreamFileSecondsRemaining	Retrieves the remaining duration, in seconds, of the currently playing file on a Stream.
sceScreamGetStreamInfo	Retrieves buffered status, Bitstream count, channel count, and sample rate information from a Stream handle.
sceScreamGetStreamLevel	Retrieves the current voice level of a Stream.
sceScreamGetStreamQueueCount	Retrieves the number of files queued on a Stream.
sceScreamSetStreamFileLoopingCount	Sets the remaining number of loops for the currently playing file on a Stream.

sceScreamGetStreamFileLengthInSeconds

Retrieves the total duration, in seconds, of the currently playing file on a Stream.

Definition

```
uint32_t sceScreamGetStreamFileLengthInSeconds(
    uint32_t handle,
    float *outSeconds,
    SceScreamSndStreamUserContext *outContext
);
```

Arguments

handle	(Input) Handle of the Stream to query. A value returned by the `sceScreamStartStream()`, `sceScreamStartStreamByFileToken()`, or `sceScreamStartStreamFromTransition()` functions.
outSeconds	(Output) Pointer to a `float` variable in which to receive the total duration, in seconds, of the currently playing file on the Stream.
outContext	(Output) Pointer to a `uint32_t` variable in which to receive the user context value assigned to the file by the application in the *userContext* member of the `SceScreamSndFileParams` structure. Can be `NULL`.

Return Values

Returns the specified Stream handle if the time information was successfully retrieved. Otherwise, returns 0.

Description

This function retrieves the duration of the currently playing file on a Stream. It stores a floating-point value, representing the total duration (in seconds) of the file, in a user-supplied variable.

Notes

The duration is obtained in respect of one iteration of the file. Any loop count setting is not considered.

See Also

`sceScreamCueStreamToTime()`, `sceScreamGetStreamFileSecondsRemaining()`, `sceScreamGetStreamFileLocationInSeconds()`

sceScreamGetStreamFileLocationInSeconds

Retrieves the current playback position, in seconds, of the currently playing file on a Stream.

Definition

```
uint32_t sceScreamGetStreamFileLocationInSeconds(
    uint32_t handle,
    float *outLocation,
    SceScreamSndStreamUserContext *outContext
);
```

Arguments

handle	(Input) Handle of the Stream to query. A value returned by the `sceScreamStartStream()`, `sceScreamStartStreamByFileToken()`, or `sceScreamStartStreamFromTransition()` functions.
outLocation	(Output) Pointer to a `float` variable in which to receive the current playback position from the beginning of the file, in seconds, of the currently playing file.
outContext	(Output) Pointer to a `uint32_t` variable in which to receive the user context value assigned to the file by the application in the *`userContext`* member of the `SceScreamSndFileParams` structure. Can be `NULL`.

Return Values

Returns the specified Stream handle if the time information was successfully retrieved. Otherwise, returns 0.

Description

This function retrieves the current playback position of the currently playing file on a Stream. It stores a floating-point value, representing the current playback position (in seconds) from the beginning of the file, in a user-supplied variable. The time is calculated from the current read position as reported by the playback voice being used.

Notes

Any loop count setting is not considered.

See Also

`sceScreamCueStreamToTime()`, `sceScreamGetStreamFileLengthInSeconds()`, `sceScreamGetStreamFileSecondsRemaining()`

sceScreamGetStreamFileLoopingCount

Retrieves the number of loops initially assigned and the number of completed loops of the currently playing file on a Stream.

Definition

```
uint32_t sceScreamGetStreamFileLoopingCount(
    uint32_t handle,
    int32_t *outSetting,
    int32_t *outCount,
    SceScreamSndStreamUserContext *outContext
);
```

Arguments

handle	(Input) Handle of the Stream that the loop count should be retrieved from. A value returned by the `sceScreamStartStream()`, `sceScreamStartStreamByFileToken()`, or `sceScreamStartStreamFromTransition()` functions.
outSetting	(Output) Pointer to an `int32_t` variable in which to receive the loop setting currently assigned to the file by the application in the *loopCount* member of the `SceScreamSndFileParams` structure. Return values can be interpreted as follows: –**2**: The file is set to loop until a new file is queued on the same handle; see `SCE_SCREAM_SND_SS_LOOP_TILL_QUEUED`. –**1**: The file is set to loop indefinitely; see `SCE_SCREAM_SND_SS_LOOP_INFINITE`. ≥**0**: The file is set to loop a finite number of times. 0 means it was set to play once without looping; 1 means to play twice; 2 means to play 3 times, and so on.
outCount	(Output) Pointer to an `int32_t` variable in which to receive the completed loop count of the playing file. This value starts at zero and is incremented each time the synthesizer plays the end of the file and starts over.
outContext	(Output) Pointer to a `uint32_t` variable in which to receive the user context value assigned to the file by the application in the *userContext* member of the `SceScreamSndFileParams` structure. Can be `NULL`.

Return Values

Returns the specified Stream handle if the looping information was successfully retrieved. Otherwise, returns 0.

Description

This function retrieves the number of completed loops played on the current file on a specified Stream handle. The *outSetting* parameter receives the initial loop setting assigned to the file by the application in the *loopCount* member of the `SceScreamSndFileParams` structure. The *outCount* parameter receives the number of completed loops that have already played. If the value received by *outSetting* is greater than zero, you can determine the remaining number of complete loops still to play by subtracting *outCount* from *outSetting*.

The initial loop count is set by the `SceScreamSndFileParams` *loopCount* member.

See Also

sceScreamSetStreamFileLoopingCount(), sceScreamGetStreamQueueCount(), SceScreamSndFileParams

sceScreamGetStreamFileSecondsRemaining

Retrieves the remaining duration, in seconds, of the currently playing file on a Stream.

Definition

```
uint32_t sceScreamGetStreamFileSecondsRemaining(
    uint32_t handle,
    float *outSeconds,
    SceScreamSndStreamUserContext *outContext
);
```

Arguments

handle	(Input) Handle of the Stream to query. A value returned by the `sceScreamStartStream()`, `sceScreamStartStreamByFileToken()`, or `sceScreamStartStreamFromTransition()` functions.
outSeconds	(Output) Pointer to a `float` variable in which to receive the remaining duration, in seconds, of the currently playing file on the Stream.
outContext	(Output) Pointer to a `uint32_t` variable in which to receive the user context value assigned to the file by the application in the *`userContext`* member of the `SceScreamSndFileParams` structure. Can be `NULL`.

Return Values

Returns the specified Stream handle if the time information was successfully retrieved. Otherwise, returns 0.

Description

This function retrieves the remaining duration of the currently playing file on a Stream. It stores a floating-point value, representing the duration (in seconds) from the current playback position to the end of the file, in a user-supplied variable. The time is calculated from the current read position as reported by the playback voice being used.

Notes

Any loop count setting is not considered.

See Also

`sceScreamCueStreamToTime()`, `sceScreamGetStreamFileLengthInSeconds()`, `sceScreamGetStreamFileLocationInSeconds()`

sceScreamGetStreamInfo

Retrieves buffered status, Bitstream count, channel count, and sample rate information from a Stream handle.

Definition

```
uint32_t sceScreamGetStreamInfo(
    uint32_t handle,
    uint32_t *outBufferedStatus,
    uint32_t *outBitstreamCount,
    uint32_t *outChannelCount,
    float *outSampleRate
);
```

Arguments

handle	(Input) Handle of the Stream to query. A value returned by the `sceScreamStartStream()`, `sceScreamStartStreamByFileToken()`, or `sceScreamStartStreamFromTransition()` functions.
outBufferedStatus	(Output) Pointer to a `uint32_t` variable in which to receive the buffered status. If the Stream is buffered, a non-zero value is returned.
outBitstreamCount	(Output) Pointer to a `uint32_t` variable in which to receive the number of Bitstreams in the Stream. If the Stream is not buffered, a zero value is returned. Can be `NULL`. **Note:** Multiple Bitstreams are not supported in this release. Bitstream count is therefore 1.
outChannelCount	(Output) Pointer to a `uint32_t` variable in which to receive the number of channels (per Bitstream) in the Stream. If the Stream is not buffered, a zero value is returned. Can be `NULL`.
outSampleRate	(Output) Pointer to a `float` variable in which to receive the sample rate of the Stream. If the Stream is not buffered, a zero value is returned. Can be `NULL`.

Return Values

Returns the specified Stream handle if the handle is still valid. Otherwise, returns 0.

Description

This function queries a Stream, then stores buffered status, Bitstream count, channel count, and sample rate information in user-supplied variables. The function checks the Stream handle to see if it is still valid, and if so, the Stream's buffered status is queried. If the Stream is buffered, the Stream file's Bitstream count, channel count, and sample rate are also queried. If the Stream is not buffered, zero is stored in the variables referenced in the *`outBufferedStatus`*, *`outBitstreamCount`*, *`outChannelCount`*, and *`outSampleRate`* parameters.

See Also

`sceScreamStartStream()`, `sceScreamGetStreamQueueCount()`

sceScreamGetStreamLevel

Retrieves the current voice level of a Stream.

Definition

```
float sceScreamGetStreamLevel(
    uint32_t handle,
    bool rms,
    bool linear
);
```

Arguments

handle	(Input) Handle of the Stream to query. A value returned by the `sceScreamStartStream()`, `sceScreamStartStreamByFileToken()`, or `sceScreamStartStreamFromTransition()` functions.
rms	(Input) Set to `TRUE` if you want the result as an averaged RMS level. Otherwise, the result is an instantaneous peak level.
linear	(Input) Set to `TRUE` if you want the result on a linear scale. Otherwise, the result is expressed in decibels (dB).

Return Values

If successful, returns a Stream's current voice level in the requested format. Otherwise, returns 0.

Description

This function retrieves the current voice level of a Stream. The returned voice level is either a RMS or instantaneous peak level, and expressed either on a linear or decibel (dB) scale.

Notes

To retrieve voice level from a Stream, it must have been initialized with the `SCE_SCREAM_SND_SS_START_GET_VOICE_LEVEL` option included in the Stream's `SceScreamSndStartParams` *`flags`* member.

See Also

`sceScreamGetStreamInfo()`, `SceScreamSndStartParams`, `SCE_SCREAM_SND_SS_START_GET_VOICE_LEVEL`

sceScreamGetStreamQueueCount

Retrieves the number of files queued on a Stream.

Definition

```
uint32_t sceScreamGetStreamQueueCount(
    uint32_t handle,
    uint32_t *outQueueCount
);
```

Arguments

handle	(Input) Handle of the Stream to query. A value returned by the `sceScreamStartStream()`, `sceScreamStartStreamByFileToken()`, or `sceScreamStartStreamFromTransition()` functions.
outQueueCount	(Output) Pointer to a `uint32_t` variable in which to receive count of items queued on the Stream handle. A value of zero can indicate either that the handle has no queued items but may still be playing an active Stream, or is not active at all.

Return Values

Returns the specified Stream handle if the queue count was successfully retrieved. Otherwise, returns 0.

Description

This function retrieves the count of queued files on the specified Stream handle that have not yet played, not including the actively playing Stream, even if it is paused. A maximum of `SCE_SCREAM_SND_FILE_QUEUE_MAX` files can be queued on a Stream handle.

See Also

`sceScreamGetStreamInfo()`, `sceScreamQueueToStream()`, `sceScreamQueueToStreamByFileToken()`

sceScreamSetStreamFileLoopingCount

Sets the remaining number of loops for the currently playing file on a Stream.

Definition

```
uint32_t sceScreamSetStreamFileLoopingCount(
    uint32_t handle,
    int32_t loopCount
);
```

Arguments

handle	(Input) Handle of the Stream to which the loop count should be applied. A value returned by the `sceScreamStartStream()`, `sceScreamStartStreamByFileToken()`, or `sceScreamStartStreamFromTransition()` functions.
loopCount	(Input) New loop count value to apply to the Stream. See "Description" below for possible settings.

Return Values

Returns the specified Stream handle if the Stream is still active. Otherwise, returns 0.

Description

This function sets or updates the remaining number of complete loops for the currently playing file on a specified Stream handle. A Stream can have multiple queued files, however, the new loop count is applied only to the currently playing file.

Setting a value of zero stops playback of the file at the end of its current cycle, then immediately begins playback of the next queued file on the same handle, if any. A value of 1 specifies playback of one more complete loop after the current cycle. A value of 2 specifies playback of two more complete loops after the current cycle, and so on. Setting `SCE_SCREAM_SND_SS_LOOP_INFINITE` specifies indefinite looping. Setting `SCE_SCREAM_SND_SS_LOOP_TILL_QUEUED` specifies looping until a new file is queued on the same handle; at which time playback continues to the end of the current cycle, and then begins playback of the new file.

The initial loop count is set by the `SceScreamSndFileParams` *loopCount* member.

Notes

Use the `sceScreamGetStreamFileLoopingCount()` function to retrieve the number of completed loops and to determine the remaining number of remaining loops. Use the `sceScreamGetStreamQueueCount()` function to get the number of files currently queued on the Stream.

See Also

`sceScreamGetStreamFileLoopingCount()`, `sceScreamGetStreamQueueCount()`, `SceScreamSndFileParams`

File System Functions

Summary

Sets a custom file i/o interface.

Function	Description
sceScreamSetDefaultFileInterface	Overrides the default file I/O functions with custom file I/O functions.

sceScreamSetDefaultFileInterface

Overrides the default file I/O functions with custom file I/O functions.

Definition

```
int32_t sceScreamSetDefaultFileInterface(
    SceScreamSndFileInterface *fileInterface
);
```

Arguments

fileInterface	(Input) Pointer to a SceScreamSndFileInterface structure initialized with the addresses of the custom file I/O functions.

Return Values

Returns SCE_SCREAM_SND_STREAM_ERROR_OK if the operation was successful, otherwise returns SCE_SCREAM_SND_STREAM_ERROR_OUT_OF_RANGE if *fileInterface* is NULL.

Description

This function allows the default file I/O functions to be overridden with custom file I/O functions. To override the defaults and use custom file I/O functions:

(1) Ensure that your custom file functions match the file I/O type definitions. See SceScreamSndStreamFileOpenFunction, and so on.

(2) Store their addresses as the corresponding members of the SceScreamSndFileInterface data structure.

(3) After initializing Sndstream and before calling any other Sndstream functions, call this function with the initialized SceScreamSndFileInterface data structure as its argument.

For further information, see “Using Custom File I/O Functions” in “Working with System Globals” chapter of the *Sndstream Library Overview*.

See Also

SceScreamSndFileInterface, SceScreamSndStreamFileOpenFunction(), SceScreamSndStreamFileInfoCBFunction(), SceScreamSndStreamFileSeekFunction(), SceScreamSndStreamFileReadFunction(), SceScreamSndStreamFileCloseFunction()

File Token Functions

Summary

These functions manipulate Stream file tokens.

Function	Description
`sceScreamByteReverseFileTokenStorage`	Reverses the endianness of values contained in a file token storage.
`sceScreamCreateFileTokenStorage`	Creates a Stream file token storage.
`sceScreamDeleteFileTokenStorage`	Deletes a file token storage.
`sceScreamDeleteStreamFileToken`	Marks a file token as unused.
`sceScreamGetFileTokenFromStorage`	Retrieves a file token from a specified storage loaded from a file.
`sceScreamParseStreamFile`	Parses an audio file and stores header information in a file token.
`sceScreamValidateFileTokenStorage`	Validates the correctness of a file token storage.

sceScreamByteReverseFileTokenStorage

Reverses the endianness of values contained in a file token storage.

Definition

```
int32_t sceScreamByteReverseFileTokenStorage(
    SceScreamSndStreamFileTokenStorage *storage
);
```

Arguments

storage	(Input) SceScreamSndStreamFileTokenStorage pointer to a storage to be endianness-reversed.

Return Values

If successful, returns zero. Otherwise returns a negative error code.

Description

The function reverses the endianness of values contained in a file token storage. You use this function if you are creating a storage on a computer with a different endianness from that of the target platform. For example, if you build on Windows and run on the PlayStation®3 platform, you would need to reverse the endianness. This can be done once on the build machine, avoiding use of runtime cycles on the target platform.

Create a SndStreamFileTokenStorage instance by calling `sceScreamCreateFileTokenStorage()`.

See Also

`sceScreamCreateFileTokenStorage()`, `sceScreamDeleteFileTokenStorage()`, `sceScreamValidateFileTokenStorage()`, `sceScreamGetFileTokenFromStorage()`

sceScreamCreateFileTokenStorage

Creates a Stream file token storage.

Definition

```
SceScreamSndStreamFileTokenStorage *sceScreamCreateFileTokenStorage(
    SceScreamSndStreamFileToken *tokens,
    uint32_t numTokens,
    uint32_t *sizePtr
);
```

Arguments

tokens	(Input) Pointer to an array of file tokens. Obtain a file token with the `sceScreamParseStreamFile()` function.
numTokens	(Input) Number of file tokens in the array specified in *tokens*.
sizePtr	(Output) Pointer to a variable in which to receive the size of the resulting structure. This is needed if you want to save the storage to disk.

Return Values

If successful, returns a pointer to a file token storage. Otherwise returns `NULL`.

Description

This function creates a Stream file token storage that can contain one or more file tokens. You can create a storage at build time, and save it in a file. At run time, you can then use the file token storage to reduce processing load.

For further information about working with file tokens, see the "Working with File Tokens" chapter of the *Sndstream Library Overview* or the Pre-parse sample program.

See Also

`sceScreamValidateFileTokenStorage()`, `sceScreamByteReverseFileTokenStorage()`, `sceScreamDeleteFileTokenStorage()`, `sceScreamGetFileTokenFromStorage()`, `sceScreamParseStreamFile()`

sceScreamDeleteFileTokenStorage

Deletes a file token storage.

Definition

```
int32_t sceScreamDeleteFileTokenStorage(
    SceScreamSndStreamFileTokenStorage *storage
);
```

Arguments

storage	(Input) SceScreamSndStreamFileTokenStorage pointer to a storage to delete.

Return Values

If successful, returns SCE_SCREAM_SND_STREAM_ERROR_OK. Otherwise returns a negative error code.

Description

This function deletes a file token storage. After calling this function, the specified storage is invalidated, so that any tokens contained within it are unusable.

Use this function only for storage areas created using the sceScreamCreateFileTokenStorage() function.

Notes

Stream files that have already been queued or started playback are unaffected by deletion of an associated file token storage. This is because the contents of file tokens are copied into the corresponding queue.

See Also

sceScreamCreateFileTokenStorage(), sceScreamValidateFileTokenStorage(), sceScreamGetFileTokenFromStorage()

sceScreamDeleteStreamFileToken

Marks a file token as unused.

Definition

```
int32_t sceScreamDeleteStreamFileToken(
    SceScreamSndStreamFileToken fileToken
);
```

Arguments

fileToken	(Input) The file token to delete, as returned by the `sceScreamParseStreamFile()` or `sceScreamGetFileTokenFromStorage()` functions.

Return Values

Returns `SCE_SCREAM_SND_STREAM_ERROR_OK` if the operation was successful, otherwise returns `SCE_SCREAM_SND_STREAM_ERROR_NOT_FOUND` if the file token was not found.

Description

This function deletes (marks as unused) a file token. You can delete a file token any time after the corresponding Stream file is queued. This is because the pre-parsed header information that is contained in the token and needed during playback is copied into the queue.

See Also

`sceScreamParseStreamFile()`, `sceScreamGetFileTokenFromStorage()`, `sceScreamStartStreamByFileToken()`, `sceScreamQueueToStreamByFileToken()`

sceScreamGetFileTokenFromStorage

Retrieves a file token from a specified storage loaded from a file.

Definition

```
SceScreamSndStreamFileToken sceScreamGetFileTokenFromStorage(
    SceScreamSndStreamFileTokenStorage *storage,
    uint32_t tokenIndex,
    const SceScreamSndStreamParseParams *parseParams
);
```

Arguments

storage	(Input) Pointer to a file token storage. Returned by the `sceScreamCreateFileTokenStorage()` function when the storage was created.
tokenIndex	(Input) Zero-based index of the token to retrieve.
parseParams	(Input) Pointer to a `SceScreamSndStreamParseParams` structure, initialized when the file token being retrieved was created with the `sceScreamParseStreamFile()` function. The `SceScreamSndStreamParseParams` structure contains a path pointer that is used to reopen the Stream file in asynchronous mode when streaming.

Return Values

If successful, returns a valid `SceScreamSndStreamFileToken`. Otherwise returns `SCE_SCREAM_SND_STREAM_INVALID_FILE_TOKEN`.

Description

This function retrieves a file token from a specified file token storage loaded from a file. The returned token can be used as input to the `sceScreamStartStreamByFileToken()` and `sceScreamQueueToStreamByFileToken()` functions, which respectively, start or queue Stream files by reference to a file token.

Note that a file token is an opaque pointer into an associated file token storage. So if you delete a token storage with `sceScreamDeleteFileTokenStorage()`, the tokens it contains become invalid.

See Also

`SceScreamSndStreamParseParams`, `sceScreamCreateFileTokenStorage()`, `sceScreamDeleteFileTokenStorage()`, `sceScreamValidateFileTokenStorage()`, `sceScreamStartStreamByFileToken()`, `sceScreamQueueToStreamByFileToken()`, `sceScreamParseStreamFile()`

sceScreamParseStreamFile

Parses an audio file and stores header information in a file token.

Definition

```
SceScreamSndStreamFileToken sceScreamParseStreamFile(
    const SceScreamSndStreamParseParams *parseParams,
    int32_t *errorCodePtr
);
```

Arguments

parseParams	(Input) Pointer to a `SceScreamSndStreamParseParams` structure, appropriately initialized for the associated Stream file.
errorCodePtr	(Output) Pointer to a variable in which to receive an error code (see `General Errors`), or zero if no error occurs.

Return Values

If successful, returns a valid `SceScreamSndStreamFileToken`. Otherwise returns `SCE_SCREAM_SND_STREAM_INVALID_FILE_TOKEN`.

Description

This function parses an audio file (intended for streaming), gathering information from the file's header and storing it in a file token. The returned token can be used as input to the `sceScreamStartStreamByFileToken()` and `sceScreamQueueToStreamByFileToken()` functions, which respectively, start or queue Stream files by reference to a file token. Tokens can be stored for future use in a `SceScreamSndStreamFileTokenStorage` with the `sceScreamCreateFileTokenStorage()` function and then retrieved by `sceScreamGetFileTokenFromStorage()`.

For further information about working with file tokens, see the "Working with File Tokens" chapter of the *Sndstream Library Overview* or the Pre-parse sample program.

See Also

`sceScreamStartStreamByFileToken()`, `sceScreamQueueToStreamByFileToken()`, `SceScreamSndStreamParseParams`, `sceScreamDeleteStreamFileToken()`, `sceScreamCreateFileTokenStorage()`, `sceScreamGetFileTokenFromStorage()`

sceScreamValidateFileTokenStorage

Validates the correctness of a file token storage.

Definition

```
int32_t sceScreamValidateFileTokenStorage(
    SceScreamSndStreamFileTokenStorage *storage,
    uint32_t storageSize
);
```

Arguments

storage	(Input) Pointer to a file token storage. Returned by the `sceScreamCreateFileTokenStorage()` function when the storage was created.
storageSize	(Input) The size, in bytes, that the storage occupies. Saved by the `sceScreamCreateFileTokenStorage()` function into a variable specified by its *sizePtr* parameter. **Note:** To bypass file token storage size checking, you can set this parameter to zero.

Return Values

If successful, returns zero. Otherwise returns a negative error code.

Description

This function verifies the correctness of a file token storage. You call this function before retrieving a file token from a storage with the `sceScreamGetFileTokenFromStorage()` function. The function verifies the appropriate endianness, version, and integrity of the storage. The function may also make changes in the storage, so the storage should reside in read-write memory.

See Also

`sceScreamCreateFileTokenStorage()`, `sceScreamDeleteFileTokenStorage()`, `sceScreamByteReverseFileTokenStorage()`, `sceScreamGetFileTokenFromStorage()`

Synchronized Play Functions

Summary

Synchronized play functions play Scream Sounds in synchronization with Streams.

Function	Description
sceScreamGetCurrentSyncClockStreamHandle	Retrieves the handle of the current sync clock Stream.
sceScreamPlaySoundSyncedByIndexEx	Plays a Scream Sound in synchronization with the sync clock, by reference to its index.
sceScreamPlaySoundSyncedByNameEx	Plays a Scream Sound in synchronization with the sync clock, by reference to its name.

sceScreamGetCurrentSyncClockStreamHandle

Retrieves the handle of the current sync clock Stream.

Definition

```
uint32_t sceScreamGetCurrentSyncClockStreamHandle(void);
```

Return Values

Returns the sync clock Stream handle, if there is one. Otherwise, returns 0.

Description

This function retrieves the handle of the Stream currently providing the master sync clock. The `sceScreamStartStreamFromTransition()` function sets the current sync clock Stream.

See Also

`sceScreamPlaySoundSyncedByIndexEx()`, `sceScreamPlaySoundSyncedByNameEx()`, `sceScreamStartStreamFromTransition()`

sceScreamPlaySoundSyncedByIndexEx

Plays a Scream Sound in synchronization with the sync clock, by reference to its index.

Definition

```
uint32_t sceScreamPlaySoundSyncedByIndexEx(
    const SceScreamSFXBlock2 *bank,
    int16_t index,
    const SceScreamSoundParams *params,
    const SceScreamSndSyncParams *syncParams
);
```

Arguments

bank	(Input) SceScreamSFXBlock2 pointer to the bank that contains the Sound to play. A `SceScreamSFXBlock2` handle is returned by the Scream Library functions `sceScreamBankLoadEx()`, `sceScreamBankLoadFromMemEx()`, or `sceScreamFindLoadedBankByName()`.
index	(Input) Zero-based index of the requested Sound within its bank.
params	(Input) Pointer to an initialized Scream Library `SceScreamSoundParams` structure containing Sound parameter settings.
syncParams	(Input) Pointer to an initialized `SceScreamSndSyncParams` structure containing synchronization properties.

Return Values

Returns the handle of the requested Sound. Returns 0 if the specified *index* is out of range or if `SceScreamSoundParams.size` specified in the *params* member is invalid.

Description

This function plays a Scream Sound in synchronization with the sync clock, and by reference to its index within a Sound Bank.

Notes

To play a Scream Sound by reference to its index within a Sound Bank — without synchronization — use the Scream Library function `sceScreamPlaySoundByIndexEx()`.

See Also

`sceScreamGetCurrentSyncClockStreamHandle()`,
`sceScreamPlaySoundSyncedByNameEx()`

sceScreamPlaySoundSyncedByNameEx

Plays a Scream Sound in synchronization with the sync clock, by reference to its name.

Definition

```
uint32_t sceScreamPlaySoundSyncedByNameEx(
    const SceScreamSFXBlock2 *bank,
    const char *name,
    const SceScreamSoundParams *params,
    const SceScreamSndSyncParams *syncParams
);
```

Arguments

bank	(Input) SceScreamSFXBlock2 pointer to the bank that contains the Sound to play. A `SceScreamSFXBlock2` handle is returned by the Scream Library functions `sceScreamBankLoadEx()`, `sceScreamBankLoadFromMemEx()`, or `sceScreamFindLoadedBankByName()`. Can also be `NULL`. See "Notes" below.
name	(Input) Name of the requested Sound within its bank.
params	(Input) Pointer to an initialized Scream Library `SceScreamSoundParams` data structure containing Sound parameter settings.
syncParams	(Input) Pointer to an initialized `SceScreamSndSyncParams` structure containing synchronization properties.

Return Values

Returns the handle of the requested Sound. Returns 0 if the specified *name* or `SceScreamSoundParams.size` specified in the *params* member is invalid.

Description

This function plays a Scream Sound in synchronization with the sync clock, and by reference to its name within a Sound Bank.

Notes

If *bank* is `NULL`, this function searches through all registered Sound Banks and plays the first Sound it finds having the specified *name*.

To play a Scream Sound by reference to its name within a Sound Bank — without synchronization — use the Scream Library function `sceScreamPlaySoundByNameEx()`.

See Also

`sceScreamGetCurrentSyncClockStreamHandle()`, `sceScreamPlaySoundSyncedByIndexEx()`

Layer Functions

Summary

These functions retrieve information, and manipulate Layers in a multi-Layer Stream file.

Function	Description
sceScreamAutoStreamLayerParams	Automates Layer gain and azimuth changes over a specified time.
sceScreamGetStreamLayerCountByFileToken	Retrieves the number of Layers associated with a Stream by file token reference.
sceScreamGetStreamLayerCountByHandle	Retrieves the number of Layers associated with a Stream handle.
sceScreamGetStreamLayerHandle	Retrieves the handle of an individual Stream Layer.
sceScreamGetStreamLayerParams	Retrieves Layer gain, azimuth, and focus parameters values.
sceScreamSetStreamLayerParams	Sets Layer gain, azimuth, and focus parameters.

sceScreamAutoStreamLayerParams

Automates Layer gain and azimuth changes over a specified time.

Definition

```
uint32_t sceScreamAutoStreamLayerParams(
    uint32_t handle,
    const SceScreamSndBitstreamParams *params,
    uint32_t layerCount,
    float32_t timeToTarget[SCE_SCREAM_SND_STREAM_MAX_BITSTREAMS],
    uint32_t behaviorFlags[SCE_SCREAM_SND_STREAM_MAX_BITSTREAMS]
);
```

Arguments

handle	(Input) Handle of a Stream containing Layers upon which to automate parameter changes. A value returned by the `sceScreamStartStream()`, `sceScreamStartStreamByFileToken()`, or `sceScreamStartStreamFromTransition()` functions.
params	(Input) Pointer to a `SceScreamSndBitstreamParams` structure, specifying target parameter values for each Layer.
layerCount	(Input) The number of Layers upon which to automate parameters. Specifies the length of the arrays used in the `SceScreamSndBitstreamParams` structure pointed to in *params*.
timeToTarget	(Input) Time taken to reach the target parameter values. Expressed in seconds, as an array of time values, one for each Layer.
behaviorFlags	(Input) Zero or more of the `Automated Parameter Change Flags`, expressed in an array, one value per Layer. Use the `OR` operator for multiple selections.

Return Values

Returns the number of Layers upon which parameters were successfully automated. Otherwise, returns 0 if the specified *handle* is invalid.

Description

This function automates smooth gain and azimuth changes on Layers over time periods specified in the *timeToTarget* parameter. Changes are made to the specified number of Layers associated with the Stream handle, starting at the first Layer.

Notes

The *timeToTarget* and *behaviorFlags* parameters apply to gain and azimuth changes only. That is, they do not affect focus settings.

Azimuth values must be expressed in degrees — not as specific speaker target constants.

See Also

`SceScreamSndBitstreamParams`, `sceScreamGetStreamLayerParams()`, `sceScreamSetStreamLayerParams()`

sceScreamGetStreamLayerCountByFileToken

Retrieves the number of Layers associated with a Stream by file token reference.

Definition

```
int32_t sceScreamGetStreamLayerCountByFileToken(
    SceScreamSndStreamFileToken fileToken
);
```

Arguments

fileToken	(Input) The file token to query, as returned by the `sceScreamParseStreamFile()` or `sceScreamGetFileTokenFromStorage()` functions.

Return Values

If successful, returns the number of Layers. Otherwise, returns `SCE_SCREAM_SND_STREAM_INVALID_FILE_TOKEN`.

Description

This function retrieves the number of Layers associated with a Stream, referenced by its file token.

Notes

The maximum number of Layers permissible on a Stream is defined by the system constant `SCE_SCREAM_SND_STREAM_MAX_BITSTREAMS`.

See Also

`SceScreamSndStreamFileToken`, `sceScreamParseStreamFile()`, `sceScreamGetFileTokenFromStorage()`, `sceScreamGetStreamLayerCountByHandle()`, `sceScreamGetStreamLayerHandle()`

sceScreamGetStreamLayerCountByHandle

Retrieves the number of Layers associated with a Stream handle.

Definition

```
int32_t sceScreamGetStreamLayerCountByHandle(
    uint32_t handle
);
```

Arguments

handle	(Input) Handle of a Stream to query. A value returned by the `sceScreamStartStream()`, `sceScreamQueueToStreamByFileToken()`, or `sceScreamStartStreamFromTransition()` functions.

Return Values

If successful, returns the number of Layers. Otherwise, returns zero.

Description

This function retrieves the number of Layers associated with a specified Stream handle.

Notes

For standard Streams that do not contain multiple Layers, the returned count is 1.

The maximum number of Layers permissible on a Stream is defined by the system constant `SCE_SCREAM_SND_STREAM_MAX_BITSTREAMS`.

See Also

`sceScreamGetStreamLayerCountByFileToken()`, `sceScreamGetStreamLayerHandle()`

sceScreamGetStreamLayerHandle

Retrieves the handle of an individual Stream Layer.

Definition

```
uint32_t sceScreamGetStreamLayerHandle(
    uint32_t handle,
    uint32_t layerIndex
);
```

Arguments

handle	(Input) Handle of a Stream from which to retrieve a Layer handle. A value returned by the `sceScreamStartStream()`, `sceScreamStartStreamByFileToken()`, or `sceScreamStartStreamFromTransition()` functions.
layerIndex	(Input) Zero-based index of a Layer for which to obtain a handle. Range: Zero to the number of Layers in the Stream minus one. You can obtain the number of Layers using the `sceScreamGetStreamLayerCountByHandle()` or `sceScreamGetStreamLayerCountByFileToken()` functions.

Return Values

If successful, returns the specified Layer handle. Otherwise, returns 0.

Description

This function retrieves the handle of an individual Layer contained within a multi-Layer Stream.

You can reference Stream Layer handles when using the Scream functions `sceScreamSetSoundParamsEx()`, `sceScreamAutoGain()`, and `sceScreamAutoPan()`. This allows you to set parameters or to automate gain and pan changes on individual Layers. For further details, see the “Working with Multi-Layer Streams” chapter of the *Sndstream Library Overview*.

Notes

For standard Streams that contain one Layer only, the returned Layer handle is the same as the specified Stream *handle*.

Layer handles can only be used with the following Scream functions: `sceScreamSetSoundParamsEx()`, `sceScreamAutoGain()`, and `sceScreamAutoPan()`.

Do not use Layer handles as arguments to the Scream `sceScreamStopSound()` function to stop a Layer. You cannot stop individual Layers. You must stop Layers collectively by stopping the containing Stream.

See Also

`sceScreamGetStreamLayerCountByHandle()`, `sceScreamGetStreamLayerCountByFileToken()`

sceScreamGetStreamLayerParams

Retrieves Layer gain, azimuth, and focus parameters values.

Definition

```
uint32_t sceScreamGetStreamLayerParams(
    uint32_t handle,
    SceScreamSndBitstreamParams *params,
    uint32_t *layerCountPtr
);
```

Arguments

handle	(Input) Handle of a Stream from which to retrieve Layer parameter values. A value returned by the `sceScreamStartStream()`, `sceScreamStartStreamByFileToken()`, or `sceScreamStartStreamFromTransition()` functions.
params	(Output) Pointer to a `SceScreamSndBitstreamParams` structure into which to store the retrieved parameter values.
layerCountPtr	(Output) Pointer to a `uint32_t` variable in which to store a count of the number of Layers from which parameters were retrieved. Set to `NULL` if count information is not required.

Return Values

Returns the specified Stream *`handle`* if its Layer parameters were successfully retrieved. Otherwise, returns 0 if the specified *`handle`* is invalid.

Description

This function retrieves gain, azimuth, and focus parameter values for all Layers associated with a Stream handle. The function stores retrieved parameter values in a `SceScreamSndBitstreamParams` structure.

See Also

`SceScreamSndBitstreamParams`, `sceScreamSetStreamLayerParams()`

sceScreamSetStreamLayerParams

Sets Layer gain, azimuth, and focus parameters.

Definition

```
uint32_t sceScreamSetStreamLayerParams(
    uint32_t handle,
    const SceScreamSndBitstreamParams *params,
    uint32_t layerCount
);
```

Arguments

handle	(Input) Handle of a Stream containing Layers upon which to set parameters. A value returned by the `sceScreamStartStream()`, `sceScreamStartStreamByFileToken()`, or `sceScreamStartStreamFromTransition()` functions.
params	(Input) Pointer to a `SceScreamSndBitstreamParams` structure, specifying parameter values for each Layer.
layerCount	(Input) The number of Layers upon which to set parameters. Specifies the length of the arrays used in the `SceScreamSndBitstreamParams` structure pointed to in *params*. You can obtain the number of Layers using the `sceScreamGetStreamLayerCountByHandle()` or `sceScreamGetStreamLayerCountByFileToken()` functions.

Return Values

Returns a count of the number of Layers upon which parameters were successfully set. Otherwise, returns 0 if the specified *handle* or *params* structure is invalid.

Description

This function sets Layer gain, azimuth, and focus parameter values, individually, for a specified number of Layers associated with a Stream handle, starting at the first Layer.

See Also

`SceScreamSndBitstreamParams`, `sceScreamGetStreamLayerParams()`, `sceScreamAutoStreamLayerParams()`

Error Codes

Error Code Macros

Macros used to create Scream error codes.

Define	Value	Description
`SCE_ERROR_ERROR_FLAG`	`0x80000000`	SDK base error code identifier.
`SCE_ERROR_MAKE_ERROR`	`(SCE_ERROR_ERROR_FLAG | ((_fac)<<16) | (_sts))`	Macro to create an error code.
`SCE_ERROR_FACILITY_SCREAM`	`0x100`	Scream facility code identifier.
`SCE_SCREAM_MAKE_ERROR`	`SCE_ERROR_MAKE_ERROR(SCE_ERROR_FACILITY_SCREAM,(_rc))`	Helper macro to create Scream-specific error code values. The `SCE_SCREAM_MAKE_ERROR` macro bit-combines `SCE_ERROR_ERROR_FLAG` and a shifted `SCE_ERROR_FACILITY_SCREAM` with a Scream-specific error value. For example, `SCE_SCREAM_MAKE_ERROR(0x101)` evaluates to `0x81000101`.

Initialization and Shutdown Errors

These error codes may be returned when initializing or shutting down the Sndstream library.

Define	Value	Description
SCE_SCREAM_SND_STREAM_INIT_ERROR_UNKNOWN	SCE_SCREAM_MAKE_ERROR(0x200)	Unknown system error occurred while attempting to initialize Sndstream. Returned by sceScreamInitStreaming().
SCE_SCREAM_SND_STREAM_INIT_ERROR_MEMORY	SCE_SCREAM_MAKE_ERROR(0x201)	sceScreamInitStreaming() was unable to allocate memory.
SCE_SCREAM_SND_STREAM_INIT_ERROR_THREADS	SCE_SCREAM_MAKE_ERROR(0x202)	sceScreamInitStreaming() was unable to create a thread.
SCE_SCREAM_SND_STREAM_INIT_ERROR_ALREADY_INITED	SCE_SCREAM_MAKE_ERROR(0x203)	Sndstream has already been initialized. Returned by sceScreamInitStreaming().
SCE_SCREAM_SND_STREAM_INIT_ERROR_INVALID_ARGS	SCE_SCREAM_MAKE_ERROR(0x204)	Some specified parameter(s) used with sceScreamInitStreaming() or sceScreamFillDefaultPlatformInitArgs() are out of range.
SCE_SCREAM_SND_STREAM_CLOSE_ERROR_NOT_INITED	SCE_SCREAM_MAKE_ERROR(0x400)	Returned by sceScreamCloseStreaming() to indicate that Sndstream has not been initialized.

File I/O Errors

These error codes may be returned by various custom file I/O prototypes.

Define	Value	Description
SCE_SCREAM_SND_STREAM_FILE_ERROR_PASS	SCE_SCREAM_MAKE_ERROR(0x600)	The file I/O system passed on (omitted) a read request. Returned by SceScreamSndStreamFileReadFunction() or SceScreamSndStreamFileAsyncReadFunction().
SCE_SCREAM_SND_STREAM_FILE_ERROR_UNKNOWN	SCE_SCREAM_MAKE_ERROR(0x601)	An error of unknown origin occurred. Returned by any of the custom file I/O prototypes.
SCE_SCREAM_SND_STREAM_FILE_ERROR_OPEN	SCE_SCREAM_MAKE_ERROR(0x602)	A file open error occurred. Returned by SceScreamSndStreamFileOpenFunction(), SceScreamSndStreamFileAsyncOpenFunction(), SceScreamSndStreamFileAsyncIsOpenCompleteFunction(), or SceScreamSndStreamFileAsyncOpenBitstreamFunction().
SCE_SCREAM_SND_STREAM_FILE_ERROR_SEEK	SCE_SCREAM_MAKE_ERROR(0x603)	A file seek error occurred. Returned by SceScreamSndStreamFileSeekFunction().
SCE_SCREAM_SND_STREAM_FILE_ERROR_READ	SCE_SCREAM_MAKE_ERROR(0x604)	A file read error occurred. Returned by SceScreamSndStreamFileReadFunction(), SceScreamSndStreamFileAsyncReadFunction(), or SceScreamSndStreamFileAsyncIsReadCompleteFunction().
SCE_SCREAM_SND_STREAM_FILE_ERROR_CLOSE	SCE_SCREAM_MAKE_ERROR(0x605)	A file close error occurred. Returned by SceScreamSndStreamFileCloseFunction(), SceScreamSndStreamFileAsyncCloseBitstreamFunction(), or SceScreamSndStreamFileAsyncCloseFunction().
SCE_SCREAM_SND_STREAM_FILE_ERROR_OK	0	No error occurred. Returned by many of the custom file I/O prototypes.

General Errors

These error codes may be returned by any Sndstream function.

Define	Value	Description
SCE_SCREAM_SND_STREAM_ERROR_UNKNOWN	SCE_SCREAM_MAKE_ERROR(0x301)	An unknown error occurred.
SCE_SCREAM_SND_STREAM_ERROR_NOT_INITED	SCE_SCREAM_MAKE_ERROR(0x302)	Sndstream has not been initialized.
SCE_SCREAM_SND_STREAM_ERROR_NOT_FOUND	SCE_SCREAM_MAKE_ERROR(0x303)	A specified file was not found.
SCE_SCREAM_SND_STREAM_ERROR_CORRUPT	SCE_SCREAM_MAKE_ERROR(0x304)	A specified file was found but contained invalid data.
SCE_SCREAM_SND_STREAM_ERROR_UNSUPPORTED	SCE_SCREAM_MAKE_ERROR(0x305)	A specified file type is not supported.
SCE_SCREAM_SND_STREAM_ERROR_UNREADABLE	SCE_SCREAM_MAKE_ERROR(0x306)	Sndstream failed to seek in or read from file.
SCE_SCREAM_SND_STREAM_ERROR_INTERNAL	SCE_SCREAM_MAKE_ERROR(0x307)	An unexpected internal error.
SCE_SCREAM_SND_STREAM_ERROR_UNRECOGNIZED	SCE_SCREAM_MAKE_ERROR(0x308)	The specified reference is not recognized.
SCE_SCREAM_SND_STREAM_ERROR_OUT_OF_RANGE	SCE_SCREAM_MAKE_ERROR(0x309)	A parameter index is out of range.
SCE_SCREAM_SND_STREAM_ERROR_NULL_POINTER	SCE_SCREAM_MAKE_ERROR(0x30A)	A specified address was `NULL`.
SCE_SCREAM_SND_STREAM_ERROR_VERSION_MISMATCH	SCE_SCREAM_MAKE_ERROR(0x30B)	The version of a specified data structure is not supported.
SCE_SCREAM_SND_STREAM_ERROR_ENDIANNESS	SCE_SCREAM_MAKE_ERROR(0x30C)	Data with incorrect endianness was used.
SCE_SCREAM_SND_STREAM_ERROR_VOICE_UNAVAILABLE	SCE_SCREAM_MAKE_ERROR(0x30D)	A voice could not be allocated for this data.
SCE_SCREAM_SND_STREAM_ERROR_ALLOCATION	SCE_SCREAM_MAKE_ERROR(0x30E)	A resource could not be allocated.
SCE_SCREAM_SND_STREAM_ERROR_NOT_ACTIVE	SCE_SCREAM_MAKE_ERROR(0x30F)	The handle or other object is no longer active.
SCE_SCREAM_SND_STREAM_ERROR_NOT_BIG_ENOUGH	SCE_SCREAM_MAKE_ERROR(0x310)	A buffer or other resource is not big enough.
SCE_SCREAM_SND_STREAM_ERROR_OK	(0)	No error occurred.