Scream Library Reference

PlayStation®Vita Programmer Tool Runtime Library Release 3.50

Document serial number: 000004892117

Table of Contents

Introduction

Introduction

Scream is an interactive audio system that runs on the PlayStation®Vita and PlayStation®4 platforms. This manual, the *Scream Library Reference*, documents the Scream runtime library API used in conjunction with the NGS synthesizer, running on the PlayStation®Vita platform. It documents all constants, type definitions, enumerations, structures, and functions required for successful operation. The *Scream Library Overview* is a counterpart to this document, and guides programmers through the functionality of Scream runtime 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 functions.
- Check all return values for expected results.

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

Constants

Scream Initialization Flags

You apply Scream initialization flags to the SceScreamPlatformInitEx2 *initFlags* member when initializing Scream with the sceScreamStartSoundSystemEx2() function.

Define	Value	Description
SCE_SCREAM_SSS_FLAGS_SILENT	(1L << 1)	Prevents Scream from outputting text messages to the TTY. Optional Scream initialization flag.
SCE_SCREAM_SSS_FLAGS_RETURN_ON_BAD_PARAM	(1L << 2)	Causes all functions to return immediately if any parameter is invalid. Optional Scream initialization flag for function-level parameter validation. This flag is set by default. **Note:** Set this flag, or SCE_SCREAM_SSS_FLAGS_HALT_ON_BAD_PARAM, or neither, but not both. For further details, see the "Notes" section of SceScreamPlatformInitEx2.
SCE_SCREAM_SSS_FLAGS_HALT_ON_BAD_PARAM	(1L << 3)	Causes Scream to halt immediately if any parameter is invalid. Optional Scream initialization flag for function-level parameter validation. **Note:** Set this flag, or SCE_SCREAM_SSS_FLAGS_RETURN_ON_BAD_PARAM, or neither, but not both. For further details, see the "Notes" section of SceScreamPlatformInitEx2.
SCE_SCREAM_SSS_FLAGS_TICK_CALLBACK	(1L << 12)	Causes Scream to make an application callback on each tick using the SceScreamSndEventCallback() mechanism. See SceScreamSndEventCallback() and SceScreamPlatformInitEx2. For information on setting up this callback, see "Configuring Per-Tick Callbacks" in the "Working with System Globals" chapter of the *Scream Library Overview*. You can also define callbacks triggered by certain Sound events by setting the callback condition in the SceScreamSoundParams *flags* member.

Document serial number: 000004892117

NGS Initialization Flags

You apply NGS synth initialization flags to the `SceScreamSystemParams` *`initFlags`* member when initializing Scream.

Define	Value	Description
`SCE_SCREAM_SND_SYNTH_INIT_FLAG_VALIDATE_PARAMS`	`(1L << 0)`	Enables validation of DSP parameter arguments with respect to the NGS synthesizer. If out-of-range values are encountered, appropriate warnings are output to the TTY.
`SCE_SCREAM_SND_SYNTH_INIT_FLAG_THREAD_MODE_DUAL`	`(1L << 1)`	Causes the Scream tick to run in a separate thread from the synthesizer update/audio output thread. Both threads share the same priority, CPU affinity, and stack size settings.
`SCE_SCREAM_SND_SYNTH_INIT_FLAG_DISABLE_LEVELS`	`(1L << 2)`	Prevents Scream from calculating peak or RMS output levels arising from synthesizer output. Valid for NGS only.

Scream System Constants

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

Define	Value	Description
SCE_SCREAM_SND_MAX_STREAMING_FILE_DIRECTORIES	16	The maximum number of streaming file directory paths. See sceScreamGetStreamingFileDirectory(), sceScreamSetStreamingFileDirectory().
SCE_SCREAM_SND_MAX_GAIN	1.0f	The maximum gain level for Scream and Sndstream API functions and structures that set gain. See the Scream SceScreamGainComponents, SceScreamSoundParams, SceScreamSynthParams, sceScreamSetMasterVolume(), sceScreamAutoGain(), and the Sndstream SceScreamSndBitstreamParams and SceScreamSndTransitionParams, among others.
SCE_SCREAM_SND_MIN_GAIN	0.0f	The minimum gain level for Scream and Sndstream API functions and structures that set gain. See the Scream SceScreamGainComponents, SceScreamSoundParams, SceScreamSynthParams, sceScreamSetMasterVolume(), sceScreamAutoGain(), and the Sndstream SceScreamSndBitstreamParams and SceScreamSndTransitionParams, among others.
SCE_SCREAM_SND_MAX_GLOBAL_REGISTERS	64	The total number of global registers. See sceScreamGetSFXGlobalReg(), and sceScreamSetSFXGlobalReg().
SCE_SCREAM_SND_MAX_LOCAL_VARIABLES	16	The maximum number of local variables per Sound instance. See sceScreamSetLocalVariableByHash(), and sceScreamGetLocalVariableByHash().
SCE_SCREAM_SND_MAX_REGISTERS	8	The total number of Sound-specific registers. Sound-specific registers are set in Bank contents, and also through the SceScreamSoundParams *registers* member.
SCE_SCREAM_SND_MAX_PITCH_BEND_FACTOR	1.0f	The maximum pitchbend factor; upper limit for the SceScreamSoundParams *pitchBendFactor* member.
SCE_SCREAM_SND_MIN_PITCH_BEND_FACTOR	(-1.0f)	The minimum pitchbend factor; lower limit for the SceScreamSoundParams *pitchBendFactor* member.
SCE_SCREAM_SND_FINES_PER_OCTAVE	1536	Fines are 128th microtonal subdivisions of a semitone. Given 12 semitones per octave, there are 1536 fines per octave.

Define	Value	Description
SCE_SCREAM_SND_MAX_PITCH_TRANSPOSE	(SCE_SCREAM_SND_FINES_PER_OCTAVE * 5)	The maximum pitch transpose amount – up or down from original pitch – is 5 octaves, expressed in fines. See the SceScreamSoundParams *pitchTranspose* member, and sceScreamAutoPitchTranspose(). **Note:** Transposition amounts up to 5 octaves may not always be possible due to sampling rate constraints and/or limits of audibility.
SCE_SCREAM_SND_MAX_DUCKERS	32	The total number of volume duckers available for simultaneous activation. See sceScreamSetMasterVolumeDucker().
SCE_SCREAM_SND_MAX_LFOS_PER_INSTANCE	4	The maximum number of LFOs allowed per Sound instance. See the SceScreamLFOParameters structure's *whichLFO* member.
SCE_SCREAM_SND_MAX_VOLUME_GROUPS	32	The maximum number of system-wide Volume Groups. See sceScreamGetMasterVolume(), sceScreamSetMasterVolume(), sceScreamSetGroupVoiceRange().
SCE_SCREAM_SND_MAXLISTENERS	16	The maximum number of system-wide 3-D sound spatialization listeners. See sceScreamCreateListener().
SCE_SCREAM_SND_MAX_NAME_LENGTH	257	Maximum character length of a Sound name (256), plus one extra character for NULL-termination. See the sceScreamBankGetSoundNameByIndex() *soundName* parameter. **Note:** Use of many Sounds with long names may result in a small processor overhead.
SCE_SCREAM_SND_MAX_BANK_NAME_LENGTH	9	Maximum character length of a Bank (8), plus one extra character for NULL-termination. See the sceScreamFindLoadedBankNameByPointer() *outBankName* parameter.

NGS System Constants

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

Define	Value	Description
SCE_SCREAM_SND_MAX_REVERBS	3	The maximum number of reverb voices. See the SceScreamSystemParams structure's *numReverbs* member.
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.

TTY Output Flags

TTY output flags specify diagnostic output options. Apply them to the `sceScreamOutputHandlerInfoToTTY()` or `sceScreamOutputAllPlayingSoundInfoToTTY()` function's *flags* parameter.

Define	Value	Description
SCE_SCREAM_SND_OUTPUT_TTY_FLAGS	(1L << 0)	Include Sound instance flag information in TTY output.
SCE_SCREAM_SND_OUTPUT_TTY_TICK	(1L << 1)	Include the tick count time stamp when the Sound instance was started in TTY output.
SCE_SCREAM_SND_OUTPUT_TTY_GROUP	(1L << 2)	Include Sound instance Group index in TTY output.
SCE_SCREAM_SND_OUTPUT_TTY_GAIN	(1L << 3)	Include Sound instance gain factors in TTY output.
SCE_SCREAM_SND_OUTPUT_TTY_AZIMUTH	(1L << 4)	Include Sound instance pan azimuth information in TTY output.
SCE_SCREAM_SND_OUTPUT_TTY_FOCUS	(1L << 5)	Include Sound instance focus information in TTY output.
SCE_SCREAM_SND_OUTPUT_TTY_TRANSPOSE	(1L << 6)	Include Sound instance pitch transpose information in TTY output.
SCE_SCREAM_SND_OUTPUT_TTY_PITCHBEND	(1L << 7)	Include Sound instance pitchbend factors in TTY output.
SCE_SCREAM_SND_OUTPUT_TTY_REGISTERS	(1L << 8)	Include Sound instance registers information in TTY output.
SCE_SCREAM_SND_OUTPUT_TTY_VOICE_INFO	(1L << 9)	Include Sound instance voice information in TTY output.

Define	Value	Description
SCE_SCREAM_SND_OUTPUT_TTY_ALL	(SCE_SCREAM_SND_OUTPUT_TTY_FLAGS | SCE_SCREAM_SND_OUTPUT_TTY_TICK | SCE_SCREAM_SND_OUTPUT_TTY_GROUP | SCE_SCREAM_SND_OUTPUT_TTY_GAIN | SCE_SCREAM_SND_OUTPUT_TTY_AZIMUTH | SCE_SCREAM_SND_OUTPUT_TTY_FOCUS | SCE_SCREAM_SND_OUTPUT_TTY_TRANSPOSE | SCE_SCREAM_SND_OUTPUT_TTY_PITCHBEND | SCE_SCREAM_SND_OUTPUT_TTY_REGISTERS | SCE_SCREAM_SND_OUTPUT_TTY_VOICE_INFO)	Include all diagnostic data in TTY output.

Scream System Defaults

System defaults are values used by the sceScreamFillDefaultScreamPlatformInitArgsEx2() function when filling a SceScreamPlatformInitEx2 structure with default values.

Define	Value	Description
SCE_SCREAM_SND_DEFAULT_INIT_FLAGS	SCE_SCREAM_SSS_FLAGS_RETURN_ON_BAD_PARAM	Default initialization flags. See SceScreamPlatformInitEx2 *initFlags* member.
SCE_SCREAM_SND_DEFAULT_PLAYBACK_MODE	SCE_SCREAM_SPEAKER_MODE_STEREO	Default playback mode. See the SceScreamPlatformInitEx2 *playbackMode* member.
SCE_SCREAM_SND_DEFAULT_MAX_LFOS	10	Default maximum allowable number of system-wide LFOs. See the SceScreamPlatformInitEx2 *maxLFOs* member.
SCE_SCREAM_SND_DEFAULT_LFO_UPDATE	0.0084f	Default LFO update interval. Expressed in seconds (equivalent to 2 Scream ticks). See the SceScreamPlatformInitEx2 *lfoUpdateRate* member.
SCE_SCREAM_SND_DEFAULT_DUCKER_UPDATE	0.0167f	Default ducker update interval. Expressed in seconds (equivalent to 4 Scream ticks). See the SceScreamPlatformInitEx2 *duckerUpdateRate* member.
SCE_SCREAM_SND_DEFAULT_GROUP_MIXER_UPDATE	0.0167f	Default group mixer update interval. Expressed in seconds (equivalent to 4 Scream ticks). See the SceScreamPlatformInitEx2 *groupMixerUpdateRate* member.
SCE_SCREAM_SND_DEFAULT_CCSOUND_UPDATE	0.0084f	Default CCSound update interval. Expressed in seconds (equivalent to 2 Scream ticks). See the SceScreamPlatformInitEx2 *ccSoundUpdateRate* member.
SCE_SCREAM_SND_DEFAULT_MIN_RIPOFF_TIME	0.25f	Default minimum time a voice must be allowed to play before it can be stolen. Expressed in seconds. See the SceScreamPlatformInitEx2 *minRipoffTime* member.
SCE_SCREAM_SND_DEFAULT_MEM_ALLOC	NULL	Use default memory allocation function. **Note:** For platforms other than Windows, the application must provide memory allocation and free functions that conform to SceScreamExternSndMemAlloc() and SceScreamExternSndMemFree(). See the SceScreamPlatformInitEx2 *memAlloc* member.

Define	Value	Description
SCE_SCREAM_SND_DEFAULT_MEM_FREE	NULL	Use default memory free function. **Note:** For platforms other than Windows, the application must provide memory allocation and free functions that conform to SceScreamExternSndMemAlloc() and SceScreamExternSndMemFree(). See the SceScreamPlatformInitEx2 *memFree* member.
SCE_SCREAM_SND_DEFAULT_MAX_BANKS	1024	Default maximum allowable number of loaded Banks. See the SceScreamPlatformInitEx2 *maxBanks* member.
SCE_SCREAM_SND_DEFAULT_MAX_ SNAPSHOTS	32	Default maximum allowable number of simultaneously active group mixer snapshots. See the SceScreamPlatformInitEx2 *maxActiveSnapshots* member.
SCE_SCREAM_SND_DEFAULT_MAX_GLOBAL_ VARIABLES	64	Default maximum allowable number of global variables. See the SceScreamPlatformInitEx2 *maxGlobalVariables* member.
SCE_SCREAM_SND_DEFAULT_MAX_CCSOUNDS	32	Default maximum allowable number of simultaneously active CCSounds. See the SceScreamPlatformInitEx2 *maxCCSounds* member.
SCE_SCREAM_SND_DEFAULT_DOPPLER_ SLEW_RATE	240.0f	Default Doppler velocity slew rate. Constrains the rate of change for Doppler velocity calculations. Expressed in meters-per-second squared. See the SceScreamPlatformInitEx2 *dopplerSlewRate* member.
SCE_SCREAM_SND_DEFAULT_MAX_ POLYPHONY	64	Default maximum number of simultaneously playable voices, beyond which voice stealing is required. See the SceScreamPlatformInitEx2 *maxPolyphony* member.

NGS System Defaults

NGS System defaults are values used by the `sceScreamFillDefaultScreamPlatformInitArgsEx2()` function when filling a `SceScreamPlatformInitEx2` structure with default values.

Define	Value	Description
SCE_SCREAM_SND_DEFAULT_THREAD_PRIORITY	128	Default thread priority. See the Scream SceScreamSystemParams *tickThreadPriority* member, and the Sndstream SceScreamSndStream PlatformInit structure's *streaming_thread_priority* and *parsing_thread_priority* members.
SCE_SCREAM_SND_DEFAULT_THREAD_AFFINITY	-1	Default thread affinity CPU. See the SceScreamSystemParams *tickThreadAffinity* member.
SCE_SCREAM_SND_DEFAULT_THREAD_STACK_SIZE	(128* 1024)	Default thread stack size. See the SceScreamSystemParams *tickThreadStackSize* member.
SCE_SCREAM_SND_DEFAULT_NUM_VAG_MONO_VOICES	64	Default maximum number of mono VAG voices to allocate on the NGS synthesizer.
SCE_SCREAM_SND_DEFAULT_NUM_PCM_MONO_VOICES	16	Default maximum number of mono PCM voices to allocate on the NGS synthesizer.
SCE_SCREAM_SND_DEFAULT_NUM_AT9_MONO_VOICES	32	Default maximum number of mono ATRAC9™ voices to allocate on the NGS synthesizer.
SCE_SCREAM_SND_DEFAULT_NUM_VAG_STEREO_VOICES	8	Default maximum number of stereo VAG voices to allocate on the NGS synthesizer.
SCE_SCREAM_SND_DEFAULT_NUM_PCM_STEREO_VOICES	8	Default maximum number of stereo PCM voices to allocate on the NGS synthesizer.
SCE_SCREAM_SND_DEFAULT_NUM_AT9_STEREO_VOICES	8	Default maximum number of stereo ATRAC9™ voices to allocate on the NGS synthesizer.
SCE_SCREAM_SND_DEFAULT_NUM_REVERBS	1	Default number of reverb voices. See the SceScreamSystemParams *numReverbs* member.
SCE_SCREAM_SND_DEFAULT_NUM_PREMASTER_COMP_ SUBMIXES	0	Default number of pre-master compressor submix voices to create. See the SceScreamSystemParams *numPremasterCompSubmixes* member.
SCE_SCREAM_SND_DEFAULT_NUM_PREMASTER_SC_ COMP_SUBMIXES	0	Default number of pre-master side-chain compressor submix voices to create. See the SceScreamSystemParams *numPremasterScCompSubmixes* member.

Define	Value	Description
SCE_SCREAM_SND_DEFAULT_SYNTH_INIT_FLAGS	0	Default NGS synth initialization flags. See the SceScreamSystemParams *initFlags* member.

Volume Groups

Volume groups enable collective manipulation of constituent Sounds, including, but not limited to their volumes. Sounds are assigned to Groups in Scream Tool, and saved with Bank contents.

Define	Value	Description
SCE_SCREAM_GROUP_VOLUME_SFX	0	Sound effects Volume Group.
SCE_SCREAM_GROUP_VOLUME_MUSIC	1	Music Volume Group.
SCE_SCREAM_GROUP_VOLUME_DIALOG	2	Dialog Volume Group.
SCE_SCREAM_GROUP_VOLUME_USER_1	3	User 1 Volume Group.
SCE_SCREAM_GROUP_VOLUME_USER_2	4	User 2 Volume Group.
SCE_SCREAM_GROUP_VOLUME_USER_3	5	User 3 Volume Group.
SCE_SCREAM_GROUP_VOLUME_USER_4	6	User 4 Volume Group.
SCE_SCREAM_GROUP_VOLUME_USER_5	7	User 5 Volume Group.
SCE_SCREAM_GROUP_VOLUME_USER_6	8	User 6 Volume Group.
SCE_SCREAM_GROUP_VOLUME_USER_7	9	User 7 Volume Group.
SCE_SCREAM_GROUP_VOLUME_USER_8	10	User 8 Volume Group.
SCE_SCREAM_GROUP_VOLUME_USER_9	11	User 9 Volume Group.
SCE_SCREAM_GROUP_VOLUME_USER_10	12	User 10 Volume Group.
SCE_SCREAM_GROUP_VOLUME_USER_11	13	User 11 Volume Group.
SCE_SCREAM_GROUP_VOLUME_USER_12	14	User 12 Volume Group.
SCE_SCREAM_GROUP_VOLUME_USER_13	15	User 13 Volume Group.
SCE_SCREAM_GROUP_VOLUME_USER_14	16	User 14 Volume Group.
SCE_SCREAM_GROUP_VOLUME_USER_15	17	User 15 Volume Group.
SCE_SCREAM_GROUP_VOLUME_USER_16	18	User 16 Volume Group.
SCE_SCREAM_GROUP_VOLUME_USER_17	19	User 17 Volume Group.
SCE_SCREAM_GROUP_VOLUME_USER_18	20	User 18 Volume Group.
SCE_SCREAM_GROUP_VOLUME_USER_19	21	User 19 Volume Group.
SCE_SCREAM_GROUP_VOLUME_USER_20	22	User 20 Volume Group.
SCE_SCREAM_GROUP_VOLUME_USER_21	23	User 21 Volume Group.
SCE_SCREAM_GROUP_VOLUME_USER_22	24	User 22 Volume Group.
SCE_SCREAM_GROUP_VOLUME_USER_23	25	User 23 Volume Group.
SCE_SCREAM_GROUP_VOLUME_USER_24	26	User 24 Volume Group.
SCE_SCREAM_GROUP_VOLUME_USER_25	27	User 25 Volume Group.
SCE_SCREAM_GROUP_VOLUME_USER_26	28	User 26 Volume Group.
SCE_SCREAM_GROUP_VOLUME_USER_27	29	User 27 Volume Group.
SCE_SCREAM_GROUP_VOLUME_USER_28	30	User 28 Volume Group.
SCE_SCREAM_GROUP_VOLUME_EXTERNAL	31	External Volume Group.
SCE_SCREAM_GROUP_MASTER_VOLUME	32	Volume Groups master. Scales all other Volume Groups.

Group Flags

Group flags allow you to specify multiple groups as a single bit field value.

Define	Value	Description
SCE_SCREAM_GROUP_FLAG_SFX	(1L << SCE_SCREAM_GROUP_VOLUME_SFX)	Sound effects Group flag.
SCE_SCREAM_GROUP_FLAG_MUSIC	(1L << SCE_SCREAM_GROUP_VOLUME_MUSIC)	Music Group flag.
SCE_SCREAM_GROUP_FLAG_DIALOG	(1L << SCE_SCREAM_GROUP_VOLUME_DIALOG)	Dialog Group flag.
SCE_SCREAM_GROUP_FLAG_USER_1	(1L << SCE_SCREAM_GROUP_VOLUME_USER_1)	User 1 Group flag.
SCE_SCREAM_GROUP_FLAG_USER_2	(1L << SCE_SCREAM_GROUP_VOLUME_USER_2)	User 2 Group flag.
SCE_SCREAM_GROUP_FLAG_USER_3	(1L << SCE_SCREAM_GROUP_VOLUME_USER_3)	User 3 Group flag.
SCE_SCREAM_GROUP_FLAG_USER_4	(1L << SCE_SCREAM_GROUP_VOLUME_USER_4)	User 4 Group flag.
SCE_SCREAM_GROUP_FLAG_USER_5	(1L << SCE_SCREAM_GROUP_VOLUME_USER_5)	User 5 Group flag.
SCE_SCREAM_GROUP_FLAG_USER_6	(1L << SCE_SCREAM_GROUP_VOLUME_USER_6)	User 6 Group flag.
SCE_SCREAM_GROUP_FLAG_USER_7	(1L << SCE_SCREAM_GROUP_VOLUME_USER_7)	User 7 Group flag.
SCE_SCREAM_GROUP_FLAG_USER_8	(1L << SCE_SCREAM_GROUP_VOLUME_USER_8)	User 8 Group flag.
SCE_SCREAM_GROUP_FLAG_USER_9	(1L << SCE_SCREAM_GROUP_VOLUME_USER_9)	User 9 Group flag.
SCE_SCREAM_GROUP_FLAG_USER_10	(1L << SCE_SCREAM_GROUP_VOLUME_USER_10)	User 10 Group flag.
SCE_SCREAM_GROUP_FLAG_USER_11	(1L << SCE_SCREAM_GROUP_VOLUME_USER_11)	User 11 Group flag.
SCE_SCREAM_GROUP_FLAG_USER_12	(1L << SCE_SCREAM_GROUP_VOLUME_USER_12)	User 12 Group flag.
SCE_SCREAM_GROUP_FLAG_USER_13	(1L << SCE_SCREAM_GROUP_VOLUME_USER_13)	User 13 Group flag.
SCE_SCREAM_GROUP_FLAG_USER_14	(1L << SCE_SCREAM_GROUP_VOLUME_USER_14)	User 14 Group flag.
SCE_SCREAM_GROUP_FLAG_USER_15	(1L << SCE_SCREAM_GROUP_VOLUME_USER_15)	User 15 Group flag.
SCE_SCREAM_GROUP_FLAG_USER_16	(1L << SCE_SCREAM_GROUP_VOLUME_USER_16)	User 16 Group flag.
SCE_SCREAM_GROUP_FLAG_USER_17	(1L << SCE_SCREAM_GROUP_VOLUME_USER_17)	User 17 Group flag.
SCE_SCREAM_GROUP_FLAG_USER_18	(1L << SCE_SCREAM_GROUP_VOLUME_USER_18)	User 18 Group flag.
SCE_SCREAM_GROUP_FLAG_USER_19	(1L << SCE_SCREAM_GROUP_VOLUME_USER_19)	User 19 Group flag.
SCE_SCREAM_GROUP_FLAG_USER_20	(1L << SCE_SCREAM_GROUP_VOLUME_USER_20)	User 20 Group flag.
SCE_SCREAM_GROUP_FLAG_USER_21	(1L << SCE_SCREAM_GROUP_VOLUME_USER_21)	User 21 Group flag.
SCE_SCREAM_GROUP_FLAG_USER_22	(1L << SCE_SCREAM_GROUP_VOLUME_USER_22)	User 22 Group flag.
SCE_SCREAM_GROUP_FLAG_USER_23	(1L << SCE_SCREAM_GROUP_VOLUME_USER_23)	User 23 Group flag.
SCE_SCREAM_GROUP_FLAG_USER_24	(1L << SCE_SCREAM_GROUP_VOLUME_USER_24)	User 24 Group flag.

Define	Value	Description
SCE_SCREAM_GROUP_FLAG_USER_25	(1L << SCE_SCREAM_GROUP_VOLUME_USER_25)	User 25 Group flag.
SCE_SCREAM_GROUP_FLAG_USER_26	(1L << SCE_SCREAM_GROUP_VOLUME_USER_26)	User 26 Group flag.
SCE_SCREAM_GROUP_FLAG_USER_27	(1L << SCE_SCREAM_GROUP_VOLUME_USER_27)	User 27 Group flag.
SCE_SCREAM_GROUP_FLAG_USER_28	(1L << SCE_SCREAM_GROUP_VOLUME_USER_28)	User 28 Group flag.
SCE_SCREAM_GROUP_FLAG_EXTERNAL	(1L << SCE_SCREAM_GROUP_VOLUME_EXTERNAL)	External Group flag.
SCE_SCREAM_GROUP_FLAG_ALL	(0xFFFFFFFF)	Global Group flag. Includes all other Group flags.

Playback Modes

You use the playback mode constants when setting or retrieving the current playback mode with the `sceScreamSetPlaybackMode()` and `sceScreamGetPlaybackMode()` functions. It is initially set in the *playbackMode* member of the `SceScreamPlatformInitEx2` structure.

Define	Value	Description
SCE_SCREAM_SPEAKER_MODE_STEREO	(0)	Stereo output mode.
SCE_SCREAM_SPEAKER_MODE_DPL2	(2)	Dolby Pro Logic II surround sound output mode.
SCE_SCREAM_SPEAKER_MODE_BEST	(6)	Auto-configures optimal playback mode at initialization time. **Note:** This option is valid only at initialization time, and cannot be used as an argument to `sceScreamSetPlaybackMode()`. To verify the chosen playback mode, you can call `sceScreamGetPlaybackMode()` after initializing Scream.

Output Speaker Targets

The output speaker targets allow you to set the SceScreamSoundParams *azimuth* member to a specific speaker target, rather than a panning azimuth. Combine the SCE_SCREAM_SND_PAN_TO_SPEAKER flag with one of the specific speaker targets.

Define	Value	Description
SCE_SCREAM_SND_PAN_TO_SPEAKER	0x80000000	Indicates a specific speaker target, rather than a panning azimuth. Must be combined, using the bitwise OR operator, with any one of the output speaker targets below.
SCE_SCREAM_SND_PAN_FL	(SCE_SCREAM_SND_PAN_TO_SPEAKER | 1)	The Front Left speaker.
SCE_SCREAM_SND_PAN_FR	(SCE_SCREAM_SND_PAN_TO_SPEAKER | 2)	The Front Right speaker.

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 with. 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().

Define	Value	Description
SCE_SCREAM_SND_AUTO_USE_SEPARATE_FACTOR	(1L << 5)	Specifies that an automated parameter change uses an automation-specific parameter factor in the appropriate structure, rather than the default API parameter factor. See the Scream functions sceScreamAutoPan(), sceScreamAutoPitchTranspose(), sceScreamAutoPitchBend(), sceScreamAutoGain(), and the Sndstream function sceScreamAutoStreamLayerParams(). For more information, see the "Use Separate Factor" section in the "Working with Sounds" chapter of *Scream Library Overview*.

LFO Constants

LFO constants consist of LFO shapes, LFO target parameters, LFO setup flags, and LFO parameter masks. For details about using these constants, see “Initializing and Controlling Sound LFOs” in the “Working with Sounds” chapter of the *Scream Library Overview*.

Define	Value	Description
LFO Modulation Shapes		
SCE_SCREAM_SND_LFO_SHAPE_OFF	0	LFO shape is off. Apply to the SceScreamLFOParameters *shape* member.
SCE_SCREAM_SND_LFO_SHAPE_SINE	1	LFO shape is a sine wave. Apply to the SceScreamLFOParameters *shape* member.
SCE_SCREAM_SND_LFO_SHAPE_SQUARE	2	LFO shape is a square wave. Apply to the SceScreamLFOParameters *shape* member.
SCE_SCREAM_SND_LFO_SHAPE_TRIANGLE	3	LFO shape is a triangle wave. Apply to the SceScreamLFOParameters *shape* member.
SCE_SCREAM_SND_LFO_SHAPE_SAW	4	LFO shape is a sawtooth wave. Apply to the SceScreamLFOParameters *shape* member.
SCE_SCREAM_SND_LFO_SHAPE_RAND	5	LFO shape is a random wave. Apply to the SceScreamLFOParameters *shape* member.
LFO Target Parameters		
SCE_SCREAM_SND_LFO_TARGET_NONE	0	No LFO target. Apply to the SceScreamLFOParameters *targetParam* member.
SCE_SCREAM_SND_LFO_TARGET_VOL	1	LFO target is a Sound's volume parameter. Apply to the SceScreamLFOParameters *targetParam* member.
SCE_SCREAM_SND_LFO_TARGET_PAN	2	LFO target is a Sound's pan (azimuth) parameter. Apply to the SceScreamLFOParameters *targetParam* member.
SCE_SCREAM_SND_LFO_TARGET_PITCH	3	LFO target is a Sound's pitch parameter. Apply to the SceScreamLFOParameters *targetParam* member.
SCE_SCREAM_SND_LFO_TARGET_PITCHBEND	4	LFO target is a Sound's pitchbend parameter. Apply to the SceScreamLFOParameters *targetParam* member.
SCE_SCREAM_SND_LFO_TARGET_LFO_RATE	5	LFO target is the rate parameter on another LFO within the same Sound instance. Apply to the SceScreamLFOParameters *targetParam* member.
SCE_SCREAM_SND_LFO_TARGET_LFO_DEPTH	6	LFO target is the depth parameter on another LFO within the same Sound instance. Apply to the SceScreamLFOParameters *targetParam* member.

Define	Value	Description
SCE_SCREAM_SND_LFO_TARGET_LOCAL_VAR	7	LFO target is a named local variable belonging to a Sound instance. Apply to the SceScreamLFOParameters *targetParam* member.
LFO Setup Flags		
SCE_SCREAM_SND_LFO_SETUP_FLAG_INVERT	(1L << 0)	Optional LFO behavior flag, specifies that the LFO shape is inverted. Apply to the SceScreamLFOParameters *setupFlags* member.
SCE_SCREAM_SND_LFO_SETUP_FLAG_RAND_START_OFFSET	(1L << 1)	Optional LFO behavior flag, specifies that the LFO shape has a random start offset. Apply to the SceScreamLFOParameters *setupFlags* member.
LFO Parameter Masks		
SCE_SCREAM_SND_LFO_MASK_TARGET_PARAM	(1L << 0)	The SceScreamLFOParameters *targetParam* member has been set. Apply one or more of the LFO member masks to the SceScreamLFOParameters *paramMask* member.
SCE_SCREAM_SND_LFO_MASK_TARGET_LFO	(1L << 1)	The SceScreamLFOParameters *targetLFO* member has been set. Apply one or more of the LFO member masks to the SceScreamLFOParameters *paramMask* member.
SCE_SCREAM_SND_LFO_MASK_SHAPE	(1L << 2)	The SceScreamLFOParameters *shape* member has been set. Apply one or more of the LFO member masks to the SceScreamLFOParameters *paramMask* member.
SCE_SCREAM_SND_LFO_MASK_RATE	(1L << 3)	The SceScreamLFOParameters *rate* member has been set. Apply one or more of the LFO member masks to the SceScreamLFOParameters *paramMask* member.
SCE_SCREAM_SND_LFO_MASK_DEPTH	(1L << 4)	The SceScreamLFOParameters *depth* member has been set. Apply one or more of the LFO member masks to the SceScreamLFOParameters *paramMask* member.
SCE_SCREAM_SND_LFO_MASK_START_OFFSET	(1L << 5)	The SceScreamLFOParameters *startOffset* member has been set. Apply one or more of the LFO member masks to the SceScreamLFOParameters *paramMask* member.
SCE_SCREAM_SND_LFO_MASK_DUTY_CYCLE	(1L << 6)	The SceScreamLFOParameters *dutyCycle* member has been set. Apply one or more of the LFO member masks to the SceScreamLFOParameters *paramMask* member.

Define	Value	Description
SCE_SCREAM_SND_LFO_MASK_LOCAL_VAR	(1L << 7)	The SceScreamLFOParameters *varNameHash*, *varRangeMax*, and *varRangeMin* members have been set. Apply one or more of the LFO member masks to the SceScreamLFOParameters *paramMask* member.

Script Speed Flags

Script speed flags provide optional behaviors for variable speed replays.

Define	Value	Description
SCE_SCREAM_SND_SCRIPTSPEED_AFFECT_PITCH	(1L << 0)	Flag option for sceScreamSetScriptSpeedFactor() and sceScreamSetGroupScriptSpeedFactor(). By default, variable speed replay affects time domain scripting properties only. Set this flag if you want variable speed replay to affect the pitch domain also.
SCE_SCREAM_SND_SCRIPTSPEED_AFFECT_ADSR	(1L << 1)	Flag option for sceScreamSetScriptSpeedFactor() and sceScreamSetGroupScriptSpeedFactor(). By default, variable speed replay does not affect ADSR envelope durations. Set this flag if you want variable speed replay to scale the durations of ADSR segments also.

NGS Synth Parameter Constants

These constants impose limits for various NGS synthesizer resources and parameters.

Define	Value	Description
SCE_SCREAM_BQ_MIN_GAIN	0.0f	The minimum value for peaking or shelving filters gain parameter. See the SceScreamSndIIRFilterParams *gain* member.
SCE_SCREAM_BQ_MAX_GAIN	16.0f	The maximum value for peaking or shelving filters gain parameter. See the SceScreamSndIIRFilterParams *gain* member.
SCE_SCREAM_BQ_MIN_RESONANCE	0.5f	The minimum value for filter resonance (or Q) parameter. See the SceScreamSndIIRFilterParams *resonance* member.
SCE_SCREAM_BQ_MAX_RESONANCE	10.0f	The maximum value for filter resonance (or Q) parameter. See the SceScreamSndIIRFilterParams *resonance* member.
SCE_SCREAM_NUM_AUX_SENDS	3	The maximum number of auxiliary sends per voice. See the SceScreamSynthParams *auxSendGain* and *auxSendDests* members.
SCE_SCREAM_NUM_AUX_BUSSES	3	The maximum number of system-wide auxiliary busses (that is, auxiliary send destinations). See the SceScreamSynthParams *auxSendDests* member.
SCE_SCREAM_SND_DIST_MIN_AB	0.0f	The minimum A or B coefficient value for the pre-send distortion effect. See the SceScreamSndDistortionParams *a* and *b* members.
SCE_SCREAM_SND_DIST_MAX_AB	10.0f	The maximum A or B coefficient value for the pre-send distortion effect. See the SceScreamSndDistortionParams *a* and *b* members.
SCE_SCREAM_SND_DIST_MIN_CLIP_GATE	0.0f	The minimum clip or gate values for the pre-send distortion effect. See the SceScreamSndDistortionParams *clip* and *gate* members.
SCE_SCREAM_SND_DIST_MAX_CLIP_GATE	1.0f	The maximum clip or gate value for the pre-send distortion effect. See the SceScreamSndDistortionParams *clip* and *gate* members.
SCE_SCREAM_SND_DIST_MIN_GAIN	0.0f	The minimum (linear) dry gain or wet gain values for the pre-send distortion effect. See the SceScreamSndDistortionParams *dryGain* and *wetGain* members.
SCE_SCREAM_SND_DIST_MAX_GAIN	4.0f	The maximum (linear) dry gain or wet gain values for the pre-send distortion effect. See the SceScreamSndDistortionParams *dryGain* and *wetGain* members.
SCE_SCREAM_SND_DISTORTION_ENABLED	(1L<<0)	A flag used to enable the pre-send distortion effect. See the SceScreamSndDistortionParams *flags* member.

NGS Synth Parameter Bit Masks

You apply NGS synthesizer parameter bit masks to the `SceScreamSynthParams` structure's *`mask`* member in order to specify which other members of the structure are being set.

Define	Value	Description
SCE_SCREAM_SND_MASK_PRESEND_FILTER_0	(1L<< 0)	The *preSendFilter0* member has been set.
SCE_SCREAM_SND_MASK_PRESEND_FILTER_1	(1L<< 1)	The *preSendFilter1* member has been set.
SCE_SCREAM_SND_MASK_PRESEND_FILTER_2	(1L<< 2)	The *preSendFilter2* member has been set.
SCE_SCREAM_SND_MASK_PRESEND_FILTER_3	(1L<< 3)	The *preSendFilter3* member has been set.
SCE_SCREAM_SND_MASK_PRESEND_DISTORTION	(1L<< 4)	The *preSendDistortion* member has been set.
SCE_SCREAM_SND_MASK_AUXSEND_GAIN_0	(1L<< 5)	The *auxSendGain[0]* member has been set.
SCE_SCREAM_SND_MASK_AUXSEND_GAIN_1	(1L<< 6)	The *auxSendGain[1]* member has been set.
SCE_SCREAM_SND_MASK_AUXSEND_GAIN_2	(1L<< 7)	The *auxSendGain[2]* member has been set.
SCE_SCREAM_SND_MASK_AUXSEND_GAINS	(SCE_SCREAM_SND_MASK_AUXSEND_GAIN_0| SCE_SCREAM_SND_MASK_AUXSEND_GAIN_1| SCE_SCREAM_SND_MASK_AUXSEND_GAIN_2)	All *auxSendGain* members have been set.
SCE_SCREAM_SND_MASK_AUXSEND_DEST_0	(1L<< 8)	The *auxSendDests[0]* member has been set.
SCE_SCREAM_SND_MASK_AUXSEND_DEST_1	(1L<< 9)	The *auxSendDests[1]* member has been set.
SCE_SCREAM_SND_MASK_AUXSEND_DEST_2	(1L<< 10)	The *auxSendDests[2]* member has been set.
SCE_SCREAM_SND_MASK_AUXSEND_DESTS	(SCE_SCREAM_SND_MASK_AUXSEND_DEST_0| SCE_SCREAM_SND_MASK_AUXSEND_DEST_1| SCE_SCREAM_SND_MASK_AUXSEND_DEST_2)	All *auxSendDests* members have been set.
SCE_SCREAM_SND_MASK_DIRECTSEND_GAIN	(1L<< 11)	The *directSendGain* member has been set.
SCE_SCREAM_SND_MASK_POSTSEND_FILTER	(1L<< 12)	The *postSendFilter* member has been set.
SCE_SCREAM_SND_MASK_RENDERING_FILTER	SCE_SCREAM_SND_MASK_POSTSEND_FILTER	The rendering filter has been set. **Note:** This is just an alias for the *postSendFilter* member being set.
SCE_SCREAM_SND_MASK_LFE_GAIN	(1L<< 13)	The *lfeGain* member has been set.

NGS Filter Mode Indices

Filter mode indices are values for the `SceScreamSndIIRFilterParams` *`type`* member.

Define	Value	Description
SCE_SCREAM_FLT_BQ_OFF	0	Specifies a filter bypass (transparent).
SCE_SCREAM_FLT_BQ_LPF	1	Specifies a low-pass filter.
SCE_SCREAM_FLT_BQ_HPF	2	Specifies a high-pass filter.
SCE_SCREAM_FLT_BQ_APF	3	Specifies an all-pass filter.
SCE_SCREAM_FLT_BQ_BPF	4	Specifies a band-pass filter.
SCE_SCREAM_FLT_BQ_NOTCH	5	Specifies a notch filter.
SCE_SCREAM_FLT_BQ_PEQ	6	Specifies a peaking-EQ filter.
SCE_SCREAM_FLT_BQ_LSH	7	Specifies a low-shelf filter.
SCE_SCREAM_FLT_BQ_HSH	8	Specifies a high-shelf filter.

I3DL2 Reverb Early Reflections Patterns

Early reflections pattern constants are values for the `SceScreamSndReverbProps` *`EarlyReflectionPattern`* member.

Define	Value	Description
SCE_SCREAM_SND_I3DL2_REVERB_ROOM1_LEFT	0	Room template 1, left channel.
SCE_SCREAM_SND_I3DL2_REVERB_ROOM1_RIGHT	1	Room template 1, right channel.
SCE_SCREAM_SND_I3DL2_REVERB_ROOM2_LEFT	2	Room template 2, left channel.
SCE_SCREAM_SND_I3DL2_REVERB_ROOM2_RIGHT	3	Room template 2, right channel.
SCE_SCREAM_SND_I3DL2_REVERB_ROOM3_LEFT	4	Room template 3, left channel.
SCE_SCREAM_SND_I3DL2_REVERB_ROOM3_RIGHT	5	Room template 3, right channel.

Sound Parameter Bit Masks

You apply sound parameter bit masks to the SceScreamSoundParams structure's *mask* member to specify which other members of the structure are being set.

Define	Value	Description
SCE_SCREAM_SND_MASK_PITCH_TRANSPOSE	(1L << 0)	The *pitchTranspose* member has been set.
SCE_SCREAM_SND_MASK_PITCH_BEND_FACTOR	(1L << 1)	The *pitchBendFactor* member has been set.
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.
SCE_SCREAM_SND_MASK_SYNTH_PARAMS	(1L << 5)	The *synthParams* member has been set.
SCE_SCREAM_SND_MASK_USERCTX	(1L << 6)	The *userCtx* member has been set.
SCE_SCREAM_SND_MASK_REGISTERS_0	(1L << 16)	The *registers*[0] member has been set.
SCE_SCREAM_SND_MASK_REGISTERS_1	(1L << 17)	The *registers*[1] member has been set.
SCE_SCREAM_SND_MASK_REGISTERS_2	(1L << 18)	The *registers*[2] member has been set.
SCE_SCREAM_SND_MASK_REGISTERS_3	(1L << 19)	The *registers*[3] member has been set.
SCE_SCREAM_SND_MASK_REGISTERS_4	(1L << 20)	The *registers*[4] member has been set.
SCE_SCREAM_SND_MASK_REGISTERS_5	(1L << 21)	The *registers*[5] member has been set.
SCE_SCREAM_SND_MASK_REGISTERS_6	(1L << 22)	The *registers*[6] member has been set.
SCE_SCREAM_SND_MASK_REGISTERS_7	(1L << 23)	The *registers*[7] member has been set.
SCE_SCREAM_SND_MASK_FLAGS	(1L << 24)	The *flags* member has been set.
SCE_SCREAM_SND_MASK_LOCAL_VARIABLES	(1L << 25)	The *localVariableData* member has been set.
SCE_SCREAM_SND_MASK_LISTENER	(1L << 26)	The *listenerHandle* member has been set.
SCE_SCREAM_SND_MASK_POSITION	(1L << 27)	The *position* member has been set.
SCE_SCREAM_SND_MASK_DOPPLER_FACTOR	(1L << 28)	The *dopplerFactor* member has been set.

Define	Value	Description
SCE_SCREAM_SND_MASK_REGISTERS	(SCE_SCREAM_SND_MASK_REGISTERS_0 | SCE_SCREAM_SND_MASK_REGISTERS_1 | SCE_SCREAM_SND_MASK_REGISTERS_2 | SCE_SCREAM_SND_MASK_REGISTERS_3 | SCE_SCREAM_SND_MASK_REGISTERS_4 | SCE_SCREAM_SND_MASK_REGISTERS_5 | SCE_SCREAM_SND_MASK_REGISTERS_6 | SCE_SCREAM_SND_MASK_REGISTERS_7)	All *registers* members are in use and have been set.

Sound Flags

Sound flags provide optional behaviors for Sounds.

Define	Value	Description
SCE_SCREAM_SND_STOP_BEHAVIOR_KEYOFF	(0L)	Specifies a graceful stop. Triggers any *On Stop Marker* Grain events and issues key-off messages to active voices with ADSR Release settings. Apply to the *behavior* parameter of the sceScreamStopSound(), sceScreamStopAllSoundsInBank(), and sceScreamStopAllSoundsInGroup() functions.
SCE_SCREAM_SND_STOP_BEHAVIOR_SILENCE	(1L)	Specifies an instantaneous stop. Does *not* trigger *On Stop Marker* Grain events or *Waveform* and *Stream* Grains with ADSR Release settings. Apply to the *behavior* parameter of the sceScreamStopSound(), sceScreamStopAllSoundsInBank(), and sceScreamStopAllSoundsInGroup() functions.
SCE_SCREAM_SND_FLAG_DO_FINISHED_CALLBACK	(1L << 0)	Sets up a callback that is made upon completion of a Sound. Apply to the SceScreamSoundParams *flags* member. For details, see "Configuring Sound Event Callbacks" in the "Working with Sounds" chapter of the *Scream Library Overview*.
SCE_SCREAM_SND_FLAG_START_STREAM_PAUSED	(1L << 1)	On initiation of playback, Scream finds the first *Stream* grain in a Sound's script and immediately starts the associated stream in a paused state. This mechanism helps prevent possible stream buffering delays. Apply to the SceScreamSoundParams *flags* member.
SCE_SCREAM_SND_FLAG_STREAM_GET_VOICE_LEVEL	(1L << 2)	Enables level detection for *Stream* Grains contained in a Sound's script. Apply to the SceScreamSoundParams *flags* member.
SCE_SCREAM_SND_FLAG_DIST_MODEL_NO_FILTER	(1L << 3)	Instructs Scream's distance modeling system not to apply air absorption filtering. Apply to the SceScreamSoundParams *flags* member.
SCE_SCREAM_SND_FLAG_DIST_MODEL_PRESEND_FILTER_3	(1L << 4)	Instructs Scream's distance modeling system to use the last of the 4 pre-send filters for air absorption. This avoids conflict over the SceScreamSynthParams.*postSendFilter* member (the default for air absorption) when a game is using it for obstruction/occlusion simulation. Apply to the SceScreamSoundParams *flags* member.

Define	Value	Description
SCE_SCREAM_SND_FLAG_DOPPLER_CAMERA_CUT	(1L << 5)	Indicates that a discontinuity in emitter location has occurred. This avoids the potential for a large instantaneous Doppler shift that might otherwise result. Apply to the SceScreamSoundParams *flags* member. You can also specify a camera cut when there has been a discontinuity in listener location. For further details, see the sceScreamSetListener() function's *cameraCut* parameter.

Polar-Pan Speaker Channel Indices

These constants identify channel data per target speaker. They are used when retrieving output levels with the `sceScreamGetMasterOutputLevel()` function.

Define	Value	Description
SCE_SCREAM_SND_POLPAN_INDEX_FL	1	Index of the Front-Left channel.
SCE_SCREAM_SND_POLPAN_INDEX_FR	6	Index of the Front-Right channel.

NGS Voice Data Type Constants

NGS voice data type constants are values you can use for the sceScreamGetAllocatedVoiceCountByType() function's *dataType* parameter.

Define	Value	Description
SCE_SCREAM_SND_VOICE_DATA_TYPE_VAG	(0)	Mono VAG-encoded (ADPCM) voice data type.
SCE_SCREAM_SND_VOICE_DATA_TYPE_PCMI16	(1)	Mono 16-bit fixed-point PCM voice data type.
SCE_SCREAM_SND_VOICE_DATA_TYPE_AT9	(2)	Mono ATRAC9™-encoded (AT9) voice data type.
SCE_SCREAM_SND_VOICE_DATA_TYPE_VAG_STEREO	(3)	Stereo (two-channel) VAG-encoded (ADPCM) voice data type.
SCE_SCREAM_SND_VOICE_DATA_TYPE_PCMI16_STEREO	(4)	Stereo (two-channel) 16-bit fixed-point PCM voice data type.
SCE_SCREAM_SND_VOICE_DATA_TYPE_AT9_STEREO	(5)	Stereo (two-channel) ATRAC9™-encoded (AT9) voice data type.
SCE_SCREAM_SND_VOICE_DATA_TYPE_COUNT	(6)	Total number of supported voice types.

NGS Buss Type Constants

NGS buss types are return values for the sceScreamGetBussPresetType() function.

Define	Value	Description
SCE_SCREAM_SND_BUSS_TYPE_UNKNOWN	(-1)	A specified name is invalid or the system was unable to determine the buss type.
SCE_SCREAM_SND_BUSS_TYPE_MASTER_BUSS	(0)	The master buss.
SCE_SCREAM_SND_BUSS_TYPE_PREMASTER_SUBMIX	(1)	A pre-master submix buss.
SCE_SCREAM_SND_BUSS_TYPE_REVERB	(2)	An auxiliary (or reverb) buss.
SCE_SCREAM_SND_BUSS_TYPE_COUNT	(3)	The number of buss types.

Output Level Constants

The output level constants define ranges of dB and linear output level values returned by the `sceScreamGetMasterOutputLevel()` function.

Define	Value	Description
SCE_SCREAM_SND_LEVEL_DB_MINIMUM	(-90.0f)	Minimum decibel value. Indicates silence.
SCE_SCREAM_SND_LEVEL_DB_NOMINAL	(0.0f)	Nominal decibel value. Indicates full-scale signal level. Values ≥ to this value indicate clipping.
SCE_SCREAM_SND_LEVEL_LINEAR_MINIMUM	(0.0f)	Minimum linear value. Indicates silence.
SCE_SCREAM_SND_LEVEL_LINEAR_NOMINAL	(1.0f)	Nominal linear value. Indicates full-scale signal level. Values ≥ to this value indicate clipping.

Memory Allocation Constants

Memory allocation constants are passed to the memory allocation prototype `SceScreamExternSndMemAlloc()` *use* parameter to indicate subsystem memory consumption. When Scream calls the application's `SceScreamExternSndMemAlloc()` function, it uses the appropriate constant in the *use* parameter, depending on how Scream will be using that memory.

Define	Value	Description
SCE_SCREAM_SND_MEM_USE_SYNTH	0	Indicates that requested memory is to be used for the underlying synthesizer (does not include reverb instances). The application receives this constant in the *use* parameter when calling `sceScreamStartSoundSystemEx2()` to start the sound system.
SCE_SCREAM_SND_MEM_USE_BANK	1	Indicates that requested memory is to be used for Banks. The application receives this constant in the *use* parameter when calling `sceScreamBankLoadEx()` to load a Bank.
SCE_SCREAM_SND_MEM_USE_REVERB	2	Indicates that requested memory is to be used for reverb instance processing. The application receives this constant in the *use* parameter when allocating a reverb instance.
SCE_SCREAM_SND_MEM_USE_PRESETS	3	Indicates that requested memory is to be used for storing effects presets.
SCE_SCREAM_SND_MEM_USE_DECODE	4	Indicates that requested memory is to be used for storing temporary decoded data.
SCE_SCREAM_SND_MEM_USE_CCS	5	Indicates that requested memory is to be used for managing the Continuous Controller Sound system.
SCE_SCREAM_SND_MEM_USE_GLOBAL_VARS	6	Indicates that requested memory is to be used for managing the global variable system. See the `SceScreamPlatformInitEx2` *maxGlobalVariables* member.

Callback Constants

Callback constants are used to communicate state information back to the host application.

Define	Value	Description
SCE_SCREAM_SND_EVENT_CB_REASON_FINISHED	0	An event callback is issued because the specified Sound instance has completely finished and is about to be deactivated and a Sound event callback was set up. See the SceScreamSndEventCallback() type definition's *reason* parameter. For information on setting up this callback, see "Configuring Sound Event Callbacks" in the "Working with Sounds" chapter of the *Scream Library Overview*.
SCE_SCREAM_SND_EVENT_CB_REASON_DESTROYED	1	An event callback is issued because the specified effect instance has been completely destroyed, and it is now safe to recover the associated memory. See the SceScreamSndEventCallback() type definition's *reason* parameter.
SCE_SCREAM_SND_EVENT_CB_REASON_SCREAM_TICK	2	An event callback is issued because Scream was initialized with SCE_SCREAM_SSS_FLAGS_TICK_CALLBACK, requesting a system-wide per-tick application callback. See the SceScreamSndEventCallback() type definition's *reason* parameter. For information on setting up this callback, see "Configuring Per-Tick Callbacks" in the "Working with System Globals" chapter of the *Scream Library Overview*.
SCE_SCREAM_SND_EVENT_CB_REASON_POST_SYNTH_INIT	3	This event callback provides an opportunity for applications to make direct synthesizer calls: after Scream has initialized the underlying synth — but before synth processing begins. It is issued from the application's main thread during the sceScreamStartSoundSystemEx2() function call. See the SceScreamSndEventCallback() type definition's *reason* parameter.
SCE_SCREAM_SND_EVENT_CB_REASON_NO_RENDER_SUBMIX	4	An event callback is issued because Scream was unable to allocate the requested rendering submix voice for the specified Sound. See the SceScreamSndEventCallback() type definition's *reason* parameter.

Sound Output Destinations

You use Sound output destinations when specifying a value for the *outputDest* parameter in such functions as sceScreamSetGroupVoiceOutputDest(), sceScreamPlaySoundByIndexEx(), and sceScreamPlaySoundByNameEx().

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.
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.

Enumerations

SceScreamI3DL2StockPresets

The I3DL2 reverb stock presets provide parameter settings for a variety of commonly used reverb effects – indoor and outdoor. Apply the macros as optional values for `sceScreamReverbSetStockPreset()` *`presetIndex`* parameter.

Enumeration Values

Macro	Value	Description
SCE_SCREAM_SND_I3DL2_ENVIRONMENT_PRESET_DEFAULT	0	The default preset. **Note:** this preset sets the I3DL2 Room parameter to -10,000 mB. Reverb output level is therefore inaudible.
SCE_SCREAM_SND_I3DL2_ENVIRONMENT_PRESET_GENERIC	1	A generic preset.
SCE_SCREAM_SND_I3DL2_ENVIRONMENT_PRESET_PADDEDCELL	2	Padded cell.
SCE_SCREAM_SND_I3DL2_ENVIRONMENT_PRESET_ROOM	3	Basic room.
SCE_SCREAM_SND_I3DL2_ENVIRONMENT_PRESET_BATHROOM	4	Bathroom.
SCE_SCREAM_SND_I3DL2_ENVIRONMENT_PRESET_LIVINGROOM	5	Living room.
SCE_SCREAM_SND_I3DL2_ENVIRONMENT_PRESET_STONEROOM	6	Stone room.
SCE_SCREAM_SND_I3DL2_ENVIRONMENT_PRESET_AUDITORIUM	7	Auditorium.
SCE_SCREAM_SND_I3DL2_ENVIRONMENT_PRESET_CONCERTHALL	8	Concert hall.
SCE_SCREAM_SND_I3DL2_ENVIRONMENT_PRESET_CAVE	9	Cave.
SCE_SCREAM_SND_I3DL2_ENVIRONMENT_PRESET_ARENA	10	Sports arena.
SCE_SCREAM_SND_I3DL2_ENVIRONMENT_PRESET_HANGAR	11	Aircraft hangar.
SCE_SCREAM_SND_I3DL2_ENVIRONMENT_PRESET_CARPETEDHALLWAY	12	Carpeted hallway.
SCE_SCREAM_SND_I3DL2_ENVIRONMENT_PRESET_HALLWAY	13	Hallway.
SCE_SCREAM_SND_I3DL2_ENVIRONMENT_PRESET_STONECORRIDOR	14	Stone corridor.
SCE_SCREAM_SND_I3DL2_ENVIRONMENT_PRESET_ALLEY	15	Alley.
SCE_SCREAM_SND_I3DL2_ENVIRONMENT_PRESET_FOREST	16	Forest.
SCE_SCREAM_SND_I3DL2_ENVIRONMENT_PRESET_CITY	17	City.
SCE_SCREAM_SND_I3DL2_ENVIRONMENT_PRESET_MOUNTAINS	18	Mountains.
SCE_SCREAM_SND_I3DL2_ENVIRONMENT_PRESET_QUARRY	19	Quarry.
SCE_SCREAM_SND_I3DL2_ENVIRONMENT_PRESET_PLAIN	20	Plain.
SCE_SCREAM_SND_I3DL2_ENVIRONMENT_PRESET_PARKINGLOT	21	Parking lot.
SCE_SCREAM_SND_I3DL2_ENVIRONMENT_PRESET_SEWERPIPE	22	Sewer pipe.
SCE_SCREAM_SND_I3DL2_ENVIRONMENT_PRESET_UNDERWATER	23	Underwater.
SCE_SCREAM_SND_I3DL2_ENVIRONMENT_PRESET_SMALLROOM	24	Small room.
SCE_SCREAM_SND_I3DL2_ENVIRONMENT_PRESET_MEDIUMROOM	25	Medium room.
SCE_SCREAM_SND_I3DL2_ENVIRONMENT_PRESET_LARGEROOM	26	Large room.
SCE_SCREAM_SND_I3DL2_ENVIRONMENT_PRESET_MEDIUMHALL	27	Medium hall.
SCE_SCREAM_SND_I3DL2_ENVIRONMENT_PRESET_LARGEHALL	28	Large hall.
SCE_SCREAM_SND_I3DL2_ENVIRONMENT_PRESET_PLATE	29	Plate.
SCE_SCREAM_SND_I3DL2_ENVIRONMENT_PRESET_NB	30	(Number of reverb presets in stock library).

Scream Data Structures

Summary

Scream data structures store data applicable to APIs that are common to all supported platforms and synthesizers.

Item	Description
SceScreamDesignerParams	Stores the basic designer parameters associated with a Sound.
SceScreamDuckerDef	Stores volume ducker parameter values.
SceScreamGainComponents	Stores the constituent gain components of a Sound instance.
SceScreamLFOParameters	Stores LFO parameter values.
SceScreamPanAzimuthComponents	Stores the constituent panning azimuth components of a Sound instance.
SceScreamPitchBendFactorComponents	Stores the constituent pitchbend factor components of a Sound instance.
SceScreamPitchTransposeComponents	Stores the constituent pitch transpose components of a Sound instance.
SceScreamPlatformInitEx2	Stores Scream platform initialization values.
SceScreamSFXBlock2	Stores data for loaded Sound Banks.
SceScreamSnd3DComponents	Stores dynamic Sound instance 3D attenuation components.
SceScreamSnd3DGrainData	Stores asset Grain 3D parameter data.
SceScreamSnd3DVector	Stores coordinates used for 3D sound spatialization.
SceScreamSndLocalVarData	Stores local variable descriptor data.
SceScreamSoundParams	Stores Sound-specific parameter values.

SceScreamDesignerParams

Stores the basic designer parameters associated with a Sound.

Definition

```
struct SceScreamDesignerParams {
    int8_t vol;
    int8_t volGroup;
    int16_t pan;
    int8_t instanceLimit;
    bool muteAtOuterRadius;
    float probabilityOfPlay;
    float dopplerFactor;
};
```

Members

vol	Original volume of the Sound as defined in the Sound's data when loaded from a Bank. Specified in linear units, ranging from 0 to 127.
volGroup	Volume Group to which the Sound is assigned as defined in the Sound's data when loaded from a Bank. One of the Volume Groups constants.
pan	Original panning azimuth of the Sound as defined in the Sound's data when loaded from a Bank. Range: 0 to 359 degrees.
instanceLimit	Instance limit count of the Sound as defined in the Sound's data when loaded from a Bank, or -1 if no instance limiting is applied.
muteAtOuterRadius	For 3D Sounds, indicates whether distance model attenuation reaches silence as distance input reaches a Sound's Outer Radius setting.
probabilityOfPlay	The probability that a Sound actually plays upon attempting playback (using the `sceScreamPlaySoundByIndexEx()` or `sceScreamPlaySoundByNameEx()` functions). Default: 1.0 (100%; designers set this value as a percentage). Note that with a probability value of 0.0, a Sound would never actually play!
dopplerFactor	Doppler Factor. A value of 1.0 indicates a natural Doppler effect. Greater-than 1.0 indicates an exaggerated Doppler effect, and less-than 1.0 indicates a diminished effect. 0.0 indicates no Doppler effect. Defaults to 1.0.

Description

This structure stores the basic designer parameters associated with a Sound, resulting from a call to `sceScreamGetSoundIndexDesignerParams()`, `sceScreamGetSoundNameDesignerParams()`, or `sceScreamGetSoundInstanceDesignerParams()`.

See Also

`sceScreamGetSoundIndexDesignerParams()`, `sceScreamGetSoundNameDesignerParams()`, `sceScreamGetSoundInstanceDesignerParams()`

SceScreamDuckerDef

Stores volume ducker parameter values.

Definition

```
struct SceScreamDuckerDef {
    int32_t source_group;
    uint32_t target_groups;
    float full_duck_amt;
    float attack_time;
    float release_time;
};
```

Members

source_group	The Volume Group that controls the volume ducker.
target_groups	The Volume Group(s) to duck when *source_group* is playing sound. One or more of the Group Flags. Use the bitwise OR operator for multiple selections.
full_duck_amt	The volume level of the *target_groups* when fully ducked. Range: 0.0 to 1.0, where 0.0 is equal to a 100% volume duck; 0.5 is equal to a 50% volume duck; 1.0 is equal to a 0% volume duck, and so on.
attack_time	The time taken for the *target_groups* volume to change from non-ducked to ducked level. Expressed in seconds. Range: 0.0 to 10.0.
release_time	The time taken for the *target_groups* volume to return from ducked to non-ducked level. Expressed in seconds. Range: 0.0 to 10.0.

Description

This structure stores parameter values used for initializing a volume ducker. Scream provides for up to SCE_SCREAM_SND_MAX_DUCKERS simultaneous volume duckers.

You reference the pointer returned on instantiating a SceScreamDuckerDef structure as an argument to the sceScreamSetMasterVolumeDucker() function.

Volume ducking is the technique of reducing the volume of certain Sounds in order to highlight other Sounds. For example, in a sports game, the volume level of crowd Sounds might be reduced during an announcement or commentary.

See Also

sceScreamSetMasterVolumeDucker()

SceScreamGainComponents

Stores the constituent gain components of a Sound instance.

Definition

```
struct SceScreamGainComponents {
    float masterVolume;
    float snapshotsMaster;
    float sfxGain;
    float apiGain;
    float lfoGain;
    float autoGain;
    float groupGain;
    float duckerGain;
    float directSendGain;
};
```

Members

masterVolume	Master volume setting, Scream's global volume (as set by sceScreamSetMasterVolume(), with SCE_SCREAM_GROUP_MASTER_VOLUME as the *which* parameter). Range: SCE_SCREAM_SND_MIN_GAIN to SCE_SCREAM_SND_MAX_GAIN.
snapshotsMaster	Group Mixer Master level based on currently active mix snapshots. Range: SCE_SCREAM_SND_MIN_GAIN to SCE_SCREAM_SND_MAX_GAIN.
sfxGain	Original gain level set on the Sound instance by the content (that is, as defined in the Sound's data when loaded from a Bank). Range: SCE_SCREAM_SND_MIN_GAIN to SCE_SCREAM_SND_MAX_GAIN.
apiGain	Gain level set on the Sound instance from an API call, such as sceScreamAutoGain(). Range: SCE_SCREAM_SND_MIN_GAIN to SCE_SCREAM_SND_MAX_GAIN.
lfoGain	Gain factor created by an active LFO that is targeting the Sound (SceScreamLFOParameters *targetParam* parameter set to SCE_SCREAM_SND_LFO_TARGET_VOL when calling sceScreamSetSoundInstanceLFO()). Range: SCE_SCREAM_SND_MIN_GAIN to SCE_SCREAM_SND_MAX_GAIN. If no active LFO, value is set to 1.0f.
autoGain	Gain factor of an automated gain change applied to the Sound. See "Notes" below. Range: SCE_SCREAM_SND_MIN_GAIN to SCE_SCREAM_SND_MAX_GAIN.
groupGain	Gain level of the Sound's Volume Group set by sceScreamSetMasterVolume(). Range: SCE_SCREAM_SND_MIN_GAIN to SCE_SCREAM_SND_MAX_GAIN.
duckerGain	Gain factor created by an active ducker activated by sceScreamSetMasterVolumeDucker(). Range: SCE_SCREAM_SND_MIN_GAIN to SCE_SCREAM_SND_MAX_GAIN. If no active ducker, value is set at 1.0f.
directSendGain	Level of the Sound's direct send gain (as set in the SceScreamSynthParams *directSendGain* member). Range: SCE_SCREAM_SND_MIN_GAIN to SCE_SCREAM_SND_MAX_GAIN.

Description

This structure stores the constituent gain components of a Sound instance and is filled by a call to sceScreamGetSoundGainComponents().

Notes

You can determine a Sound's aggregate gain level by multiplying together the values of its constituent gain components — with the exception of the *directSendGain* member, which relates to the synthesizer's auxiliary buss mechanism.

By default, the automated gain change function sceScreamAutoGain() targets the *apiGain* component. However, if you set the SCE_SCREAM_SND_AUTO_USE_SEPARATE_FACTOR behavior flag when calling sceScreamAutoGain(), the function instead targets the *autoGain* component.

See Also

sceScreamGetSoundGainComponents(), sceScreamAutoGain()

Document serial number: 000004892117

SceScreamLFOParameters

Stores LFO parameter values.

Definition

```
struct SceScreamLFOParameters {
    uint8_t whichLFO;
    uint8_t targetParam;
    uint8_t targetLFO;
    uint8_t setupFlags;
    uint8_t shape;
    int8_t depth;
    uint8_t dutyCycle;
    uint8_t startOffset;
    float rate;
    uint32_t varNameHash;
    float varRangeMax;
    float varRangeMin;
    uint32_t paramMask;
};
```

Members

whichLFO	One-based index of a Sound instance LFO to set. Range: 1 to SCE_SCREAM_SND_MAX_LFOS_PER_INSTANCE.
targetParam	Target parameter to be modulated by the LFO. One of the LFO Target Constants.
targetLFO	Where the LFO is modulating a parameter on another LFO (that is, *targetParam* is set to SCE_SCREAM_SND_LFO_TARGET_LFO_RATE or SCE_SCREAM_SND_LFO_TARGET_LFO_DEPTH), this member specifies the one-based index of the target LFO. Range: 1 to SCE_SCREAM_SND_MAX_LFOS_PER_INSTANCE. See "Notes" below.
setupFlags	Optional LFO shape behaviors. One or more of the LFO Setup Flags. Use the bitwise OR operator for multiple selections.
shape	Shape of the LFO waveform. One of the LFO Shape Constants.
depth	Modulation depth of the LFO. Expressed as a percentage of the target parameter's range. Range: 0 to 100.
dutyCycle	Duty cycle of a square-shaped LFO waveform. Applies only if *shape* is SCE_SCREAM_SND_LFO_SHAPE_SQUARE. Range: 1 to 99. Default: 50. For further details about duty cycle, see "Initializing and Controlling Sound LFOs" in the "Working with Sounds" chapter of the *Scream Library Overview*.
startOffset	A start offset into the LFO waveform. Expressed as a percentage of one cycle of the LFO *shape*. Range: 0 to 100.
rate	Rate (or frequency) of the LFO. Expressed in Hertz (cycles per second). Range: 0.0 to 50.0.
varNameHash	32-bit hash of a named local variable to set. Applicable if *targetParam* is SCE_SCREAM_SND_LFO_TARGET_LOCAL_VAR only. See the sceScreamGetHashFromName() function.
varRangeMax	Maximum (upper) extent of local variable range when *depth* is 100. Applicable if *targetParam* is SCE_SCREAM_SND_LFO_TARGET_LOCAL_VAR only.
varRangeMin	Minimum (lower) extent of local variable range when *depth* is 100. Applicable if *targetParam* is SCE_SCREAM_SND_LFO_TARGET_LOCAL_VAR only.
paramMask	Specifies which members of this structure have valid settings. One or more of the LFO Parameter Mask Constants. Use the bitwise OR operator for multiple selections.

Description

This structure stores parameter values used for setting an LFO on a Sound instance. Scream provides for up to SCE_SCREAM_SND_MAX_LFOS_PER_INSTANCE simultaneous LFOs per Sound instance. To initialize or modify an LFO, you use the sceScreamSetSoundInstanceLFO() function, pointing to this structure in its *params* parameter.

Notes

When modifying the parameters of a running LFO (whether set up from Bank contents or from the Scream API), only changes to rate and depth can be performed seamlessly. Changes to all other parameters (for example, *shape*, *targetParam*, *startOffset*, and so on) cause the LFO to restart from the beginning of its shape.

When using an LFO to modulate the *rate* or *depth* parameter of another LFO within the same Sound instance, you must ensure that the LFO you specify in the *targetLFO* member is not the same LFO that you are setting in the *whichLFO* member.

For further details about setting these parameters, see "Initializing and Controlling Sound LFOs" in the "Working with Sounds" chapter of the *Scream Library Overview*.

See Also

sceScreamSetSoundInstanceLFO()

SceScreamPanAzimuthComponents

Stores the constituent panning azimuth components of a Sound instance.

Definition

```
struct SceScreamPanAzimuthComponents {
    uint32_t sfxPanAzimuth;
    uint32_t apiPanAzimuth;
    uint32_t lfoPanAzimuth;
    uint32_t autoPanAzimuth;
};
```

Members

sfxPanAzimuth	Original panning azimuth set on the Sound instance by the content (that is, as defined in the Sound's data when loaded from a Bank). Range: 0 to 359.
apiPanAzimuth	Panning azimuth set on the Sound instance from an API call, such as sceScreamAutoPan(). Range: 0 to 359.
lfoPanAzimuth	Panning azimuth produced by an active LFO that is targeting the Sound (SceScreamLFOParameters *targetParam* parameter set to SCE_SCREAM_SND_LFO_TARGET_PAN when calling sceScreamSetSoundInstanceLFO()). Range: 0 to 359. If no active LFO, value is set at 0.
autoPanAzimuth	Panning azimuth offset of an automated panning azimuth change applied to the Sound. See "Notes" below. Range: 0 to 359.

Description

This structure stores the constituent panning azimuth components of a Sound instance and is filled by a call to sceScreamGetSoundPanAzimuthComponents().

Notes

You can determine a Sound's aggregate panning azimuth by adding together the values of its constituent azimuth components modulo 360.

By default, the automated panning azimuth change function sceScreamAutoPan() targets the *apiPanAzimuth* component. However, if you set the SCE_SCREAM_SND_AUTO_USE_SEPARATE_FACTOR behavior flag when calling sceScreamAutoPan(), the function instead targets the *autoPanAzimuth* component.

See Also

sceScreamGetSoundPanAzimuthComponents(), sceScreamAutoPan()

SceScreamPitchBendFactorComponents

Stores the constituent pitchbend factor components of a Sound instance.

Definition

```
struct SceScreamPitchBendFactorComponents {
    float sfxPitchBendFactor;
    float apiPitchBendFactor;
    float lfoPitchBendFactor;
    float autoPitchBendFactor;
};
```

Members

sfxPitchBendFactor	Original pitchbend factor amount set on the Sound instance by the content (that is, as defined in the Sound's data when loaded from a Bank). Range: SCE_SCREAM_SND_MIN_PITCH_BEND_FACTOR to SCE_SCREAM_SND_MAX_PITCH_BEND_FACTOR.
apiPitchBendFactor	Pitchbend factor amount set on the Sound instance from an API call, such as sceScreamAutoPitchBend(). Range: SCE_SCREAM_SND_MIN_PITCH_BEND_FACTOR to SCE_SCREAM_SND_MAX_PITCH_BEND_FACTOR.
lfoPitchBendFactor	Pitchbend factor amount created by an active LFO (SceScreamLFOParameters *targetParam* parameter set to SCE_SCREAM_SND_LFO_TARGET_PITCHBEND when calling sceScreamSetSoundInstanceLFO()). Range: SCE_SCREAM_SND_MIN_PITCH_BEND_FACTOR to SCE_SCREAM_SND_MAX_PITCH_BEND_FACTOR. If no active LFO, value is set at 0.
autoPitchBendFactor	Pitchbend factor of an automated pitchbend change applied to the Sound. See "Notes" below. Range: SCE_SCREAM_SND_MIN_PITCH_BEND_FACTOR to SCE_SCREAM_SND_MAX_PITCH_BEND_FACTOR.

Description

This structure stores the constituent pitchbend factor components of a Sound instance and is filled by a call to sceScreamGetSoundPitchBendFactorComponents().

Notes

You can determine a Sound's aggregate pitchbend factor by multiplying together the values of its constituent pitchbend components and clamping between SCE_SCREAM_SND_MIN_PITCH_BEND_FACTOR and SCE_SCREAM_SND_MAX_PITCH_BEND_FACTOR.

By default, the automated pitchbend change function sceScreamAutoPitchBend() targets the *apiPitchBendFactor* component. However, if you set the SCE_SCREAM_SND_AUTO_USE_SEPARATE_FACTOR behavior flag when calling sceScreamAutoPitchBend(), the function instead targets the *autoPitchBendFactor* component.

See Also

sceScreamGetSoundPitchBendFactorComponents(), sceScreamAutoPitchBend()

SceScreamPitchTransposeComponents

Stores the constituent pitch transpose components of a Sound instance.

Definition

```
struct SceScreamPitchTransposeComponents {
    int32_t sfxPitchTranspose;
    int32_t apiPitchTranspose;
    int32_t lfoPitchTranspose;
    int32_t autoPitchTranspose;
};
```

Members

sfxPitchTranspose	Original pitch transposition amount set on the Sound instance by the content (that is, as defined in the Sound's data when loaded from a Bank). Range: -SCE_SCREAM_SND_MAX_PITCH_TRANSPOSE to +SCE_SCREAM_SND_MAX_PITCH_TRANSPOSE.
apiPitchTranspose	Pitch transposition amount set on the Sound instance from an API call, such as sceScreamAutoPitchTranspose(). Range: -SCE_SCREAM_SND_MAX_PITCH_TRANSPOSE to +SCE_SCREAM_SND_MAX_PITCH_TRANSPOSE.
lfoPitchTranspose	Pitch transposition amount created by an active LFO (SceScreamLFOParameters *targetParam* parameter set to SCE_SCREAM_SND_LFO_TARGET_PITCH when calling sceScreamSetSoundInstanceLFO()). Range: -SCE_SCREAM_SND_MAX_PITCH_TRANSPOSE to +SCE_SCREAM_SND_MAX_PITCH_TRANSPOSE. If no active LFO, value is set at 0.
autoPitchTranspose	Pitch transposition offset of an automated pitch transpose change applied to the Sound. See "Notes" below. Range: -SCE_SCREAM_SND_MAX_PITCH_TRANSPOSE to +SCE_SCREAM_SND_MAX_PITCH_TRANSPOSE.

Description

This structure stores the constituent pitch transpose components of a Sound instance and is filled by a call to sceScreamGetSoundPitchTransposeComponents().

Notes

You can determine a Sound's aggregate pitch transposition by adding together the values of its constituent pitch components and clamping to SCE_SCREAM_SND_MAX_PITCH_TRANSPOSE.

By default, the automated pitch transpose change function sceScreamAutoPitchTranspose() targets the *apiPitchTranspose* component. However, if you set the SCE_SCREAM_SND_AUTO_USE_SEPARATE_FACTOR behavior flag when calling sceScreamAutoPitchTranspose(), the function instead targets the *autoPitchTranspose* component.

See Also

sceScreamGetSoundPitchTransposeComponents(), sceScreamAutoPitchTranspose()

SceScreamPlatformInitEx2

Stores Scream platform initialization values.

Definition

```
struct SceScreamPlatformInitEx2 {
    uint32_t size;
    uint32_t initFlags;
    uint32_t playbackMode;
    uint32_t maxLFOs;
    float lfoUpdateRate;
    float duckerUpdateRate;
    float groupMixerUpdateRate;
    float ccSoundUpdateRate;
    float minRipoffTime;
    SceScreamExternSndMemAlloc memAlloc;
    SceScreamExternSndMemFree memFree;
    SceScreamSndEventCallback eventCallback;
    uint32_t maxBanks;
    uint32_t maxPolyphony;
    uint32_t maxActiveSnapshots;
    uint32_t maxGlobalVariables;
    uint32_t maxCCSounds;
    float dopplerSlewRate;
    void *pGroupMixerFile;
    uint32_t groupMixerFileSize;
    void *pBussConfigFile;
    uint32_t bussConfigFileSize;
    void *pDistanceModelFile;
    SceScreamSystemParams synthParams;
};
```

Members

size — The size of this data structure; use `sizeof(SceScreamPlatformInitEx2)` to determine.

initFlags — Any combination of the Initialization Flags. Defaults to `SCE_SCREAM_SND_DEFAULT_INIT_FLAGS`. See "Notes" below.

playbackMode — Initial Playback Mode. Defaults to `SCE_SCREAM_SND_DEFAULT_PLAYBACK_MODE`. See "Notes" below.

maxLFOs — Maximum allowable number of system-wide LFOs. Minimum: 0. Defaults to `SCE_SCREAM_SND_DEFAULT_MAX_LFOS`.

lfoUpdateRate — Interleave LFO update interval. Expressed in fractions of a second. Range: 0.0 to 1.0. Set to 0.0 for no LFO processing. Defaults to `SCE_SCREAM_SND_DEFAULT_LFO_UPDATE`.

duckerUpdateRate — Volume ducker update interval. Expressed in fractions of a second. Range: 0.0 to 1.0. Set to 0.0 for no ducker processing. Defaults to `SCE_SCREAM_SND_DEFAULT_DUCKER_UPDATE`.

groupMixerUpdateRate — Group mixer update interval. Expressed in fractions of a second. Range: 0.0 to 1.0. Set to 0.0 for no group mixer processing. Defaults to `SCE_SCREAM_SND_DEFAULT_GROUP_MIXER_UPDATE`.

ccSoundUpdateRate — Continuous Controller Sound (CCSound) update interval. Expressed in fractions of a second. Range: 0.0 to 1.0. Set to 0.0 for no CCSound processing. Defaults to `SCE_SCREAM_SND_DEFAULT_CCSOUND_UPDATE`.

minRipoffTime	Minimum time that a voice must be allowed to play before it can be stolen. Expressed in seconds. Recommended range: 0.25 to 5.0. Setting to 0.0 specifies that voices are immediately available for stealing. Defaults to SCE_SCREAM_SND_DEFAULT_MIN_RIPOFF_TIME.
memAlloc	Address of an application-defined memory allocation function. See SceScreamExternSndMemAlloc(). Defaults to SCE_SCREAM_SND_DEFAULT_MEM_ALLOC. **Note:** For platforms other than Windows, the application must provide a memory allocation function. On Windows, you can set *memAlloc* to NULL to use an internal memory allocation function.
memFree	Address of an application-defined memory free function. See SceScreamExternSndMemFree(). Defaults to SCE_SCREAM_SND_DEFAULT_MEM_FREE. **Note:** For platforms other than Windows, the application must provide a memory free function. On Windows, you can set *memFree* to NULL to use an internal memory free function.
eventCallback	Address of a single application-defined event callback function. See SceScreamSndEventCallback(). Defaults to NULL. For information on using this callback, see "Configuring Sound Event Callbacks" in the "Working with Sounds" chapter, and "Configuring Per-Tick Callbacks" in the "Working with System Globals" chapter of the *Scream Library Overview*.
maxBanks	Maximum allowable number of loaded banks. Minimum: 1. Defaults to SCE_SCREAM_SND_DEFAULT_MAX_BANKS. See "Notes" below.
maxPolyphony	Maximum allowable number of simultaneous voices (of all types). Range: 1 to 512. Defaults to SCE_SCREAM_SND_DEFAULT_MAX_POLYPHONY. See "Notes" below.
maxActiveSnapshots	Maximum allowable number of simultaneously active group mixer snapshots. Defaults to SCE_SCREAM_SND_DEFAULT_MAX_SNAPSHOTS. You can effectively disable the group mixer system by setting this member to zero.
maxGlobalVariables	Maximum allowable number of global variables. Defaults to SCE_SCREAM_SND_DEFAULT_MAX_GLOBAL_VARIABLES.
maxCCSounds	Maximum allowable number of simultaneously active CCSounds. Defaults to SCE_SCREAM_SND_DEFAULT_MAX_CCSOUNDS.
dopplerSlewRate	The slew rate for Doppler velocity calculations, specifying the maximum allowable velocity change per second. Expressed in meters-per-second squared. Defaults to SCE_SCREAM_SND_DEFAULT_DOPPLER_SLEW_RATE.
pGroupMixerFile	Optional memory pointer to a pre-loaded group mixer file (GMX). Defaults to NULL. See sceScreamActivateMixSnapshot() and other group mix functions.
groupMixerFileSize	Values for this member are ignored. It is no longer necessary to specify the size of a group mixer file.
pBussConfigFile	Optional memory pointer to a pre-loaded buss configuration file (BUS); also known as an effect presets file. Defaults to NULL. See sceScreamApplyBussPreset().
bussConfigFileSize	Values for this member are ignored. It is no longer necessary to specify the size of a buss configuration file.
pDistanceModelFile	Optional memory pointer to a pre-loaded distance model file (DML). Defaults to NULL.
synthParams	An embedded SceScreamSystemParams structure, containing parameters required for configuring the underlying synthesizer.

Description

This structure stores initialization values for the Scream platform. It is used in a call to `sceScreamFillDefaultScreamPlatformInitArgsEx2()`, which initializes the structure with default values for each member, as specified above. `sceScreamFillDefaultScreamPlatformInitArgsEx2()` also initializes the embedded `SceScreamSystemParams` structure with default values. The `sceScreamStartSoundSystemEx2()` function uses this structure to initialize Scream.

Notes

The *`initFlags`* member provides two – mutually exclusive – options for function-level parameter validation: `SCE_SCREAM_SSS_FLAGS_RETURN_ON_BAD_PARAM` (default), and `SCE_SCREAM_SSS_FLAGS_HALT_ON_BAD_PARAM`. Users can also set *`initFlags`* to include *neither* of these options, which allows execution to proceed even if one or more parameter values are bad. Note however, that this may produce unpredictable behaviors or cause Scream to crash.

When initializing Scream, you can set *`playbackMode`* to `SCE_SCREAM_SPEAKER_MODE_BEST`, which auto-configures Scream to use the optimal playback mode on the host system. Playback mode can also be changed at run time with a call to `sceScreamSetPlaybackMode()`; call `sceScreamGetPlaybackMode()` to determine which mode was selected.

The *`lfoUpdateRate`* and *`duckerUpdateRate`* members are expressed as floating-point values in fractions of a second. In practice, however, specified values are rounded to the nearest Scream tick (equivalent to one 240th of a second).

There is no upper limit to the value specified for the *`maxBanks`* member. The memory cost on the PlayStation®Vita/NGS platform is 4 bytes per Bank.

The approximate memory overhead for a single voice of polyphony is 120 bytes. So, for example, with default maximum polyphony, the dynamic memory requirement when calling `sceScreamStartSoundSystemEx2()` is around 7.5 kB.

See Also

`sceScreamFillDefaultScreamPlatformInitArgsEx2()`, `SceScreamSystemParams`, `sceScreamStartSoundSystemEx2()`, `SceScreamExternSndMemAlloc()`, `SceScreamExternSndMemFree()`, `SceScreamSndEventCallback()`

SceScreamSFXBlock2

Stores data for loaded Sound Banks.

Definition

```
struct SceScreamSFXBlock2 {
    uint32_t reserved[16];
};
```

Members

reserved For internal use only.

Description

This structure stores data for loaded Sound Banks.

Notes

Pointers to SceScreamSFXBlock2 structures are often used as handles for Scream banks. De-referencing these pointers, and inspecting or modifying the members of this structure is not recommended.

SceScreamSnd3DComponents

Stores dynamic Sound instance 3D attenuation components.

Definition

```
struct SceScreamSnd3DComponents {
    float dryGainFactor;
    float wetGainFactor;
    float lfeGainFactor;
    float lpfCutoff;
    float spreadFactor;
};
```

Members

dryGainFactor	Dry (or direct) signal gain factor applied to a Grain. Range: SCE_SCREAM_SND_MIN_GAIN to SCE_SCREAM_SND_MAX_GAIN.
wetGainFactor	Wet (or auxiliary) signal gain factor applied to a Grain. Range: SCE_SCREAM_SND_MIN_GAIN to SCE_SCREAM_SND_MAX_GAIN.
lfeGainFactor	LFE gain factor applied to a Grain. Range: SCE_SCREAM_SND_MIN_GAIN to SCE_SCREAM_SND_MAX_GAIN.
lpfCutoff	Air absorption (or direct path) low-pass filter cut-off frequency applied to a Grain. Range: 0.0 to 23,999.0 Hz.
spreadFactor	Spread factor applied to a Grain. Range: 0.0 to 100.0. This factor maps to smart pan for split-channel assets, and to focus for multi-channel assets.

Description

This structure stores dynamic 3D attenuation component values, resulting from an active distance model on a Sound instance. You can retrieve 3D component values from a Sound instance by calling the sceScreamSoundInstanceGet3DComponents() function, which stores its results in an array of SceScreamSnd3DComponents structures, one structure per Grain queried.

See Also

sceScreamSoundInstanceGet3DComponents(), sceScreamSoundIndexGet3DDesignerParams(), sceScreamSoundNameGet3DDesignerParams(), sceScreamSoundInstanceGet3DDesignerParams(), SceScreamSnd3DGrainData

SceScreamSnd3DGrainData

Stores asset Grain 3D parameter data.

Definition

```
struct SceScreamSnd3DGrainData {
    float innerRadius;
    float outerRadius;
    float dryRolloffFactor;
    float wetRolloffFactor;
    float lfeRolloffFactor;
    float airAbsorptionFactor;
};
```

Members

innerRadius	Inner radius for the assigned distance model. Expressed in meters. See "Notes" below.
outerRadius	Outer radius for the assigned distance model. Expressed in meters. See "Notes" below.
dryRolloffFactor	Rolloff factor for the assigned distance model's dry level curve.
wetRolloffFactor	Rolloff factor for the assigned distance model's wet level curve.
lfeRolloffFactor	Rolloff factor for the assigned distance model's LFE level curve.
airAbsorptionFactor	Rolloff factor for the assigned distance model's air absorption (low-pass filter) curve.

Description

This structure stores 3D parameter data belonging to asset Grains (*Waveform* and *Stream* Grains). You can query Sounds for 3D parameter data using the sceScreamSoundIndexGet3DDesignerParams(), sceScreamSoundNameGet3DDesignerParams(), and sceScreamSoundInstanceGet3DDesignerParams() functions. These functions store retrieved 3D parameter data in array of SceScreamSnd3DGrainData structures.

Notes

If distance in your game is expressed in units other than meters, you must convert values stored in the *innerRadius* and *outerRadius* members back to your game's distance units. See the sceScreamGetWorldUnitsPerMeter() function.

See Also

sceScreamSoundIndexGet3DDesignerParams(), sceScreamSoundNameGet3DDesignerParams(), sceScreamSoundInstanceGet3DDesignerParams(), sceScreamSetWorldUnitsPerMeter(), sceScreamGetWorldUnitsPerMeter()

SceScreamSnd3DVector

Stores coordinates used for 3D sound spatialization.

Definition

```
struct SceScreamSnd3DVector {
    float fX;
    float fY;
    float fZ;
};
```

Members

fX	X coordinate. Specifies a position on the left↔right axis. With respect to a listener's view of the screen: negative values are left of center; positive values are right of center.
fY	Y coordinate. Specifies a position on the down↔up axis. With respect to a listener's view of the screen: negative values are below center; positive values are above center.
fZ	Z coordinate. Specifies a position on the backward↔forward axis. With respect to a listener's view of the screen: negative values are out of the screen; positive values are into the screen.

Description

This structure stores coordinates used for 3D sound spatialization. Use this structure to specify the location of a listener, the front- and top-orientation vectors relative to the listener, and the location of sound-emitting objects.

See Also

sceScreamCreateListener(), sceScreamDeleteListener(), sceScreamGetListener(), sceScreamSetListener(), sceScreamCalcSoundAngles()

SceScreamSndLocalVarData

Stores local variable descriptor data.

Definition

```
struct SceScreamSndLocalVarData {
    uint32_t numDescs;
    struct
    {
        uint32_t namehash;
        float val;
    } desc[SCE_SCREAM_SND_MAX_LOCAL_VARIABLES];
};
```

Members

numDescs	The number of valid descriptors in this structure. Range: 1 to SCE_SCREAM_SND_MAX_LOCAL_VARIABLES.
namehash	32-bit hash of a named local variable to set. See the sceScreamGetHashFromName() function.
val	A floating-point value to set the local variable to.
desc	Local variable descriptor.

Description

This structure stores descriptor data for local variables, and is used to set the value of one or more local variables belonging to a Sound instance. The structure is embedded as the *localVariableData* member of the SceScreamSoundParams structure.

Notes

Local variables initialize with a default value of 0.0. If you wish to initialize a Sound's local variables to values other than 0.0, you can use this structure as the SceScreamSoundParams's *localVariableData* parameter in a call to sceScreamPlaySoundByIndexEx() or sceScreamPlaySoundByNameEx().

See Also

sceScreamPlaySoundByNameEx(), sceScreamPlaySoundByIndexEx(), sceScreamSetSoundParamsEx(), sceScreamSetLocalVariableByHash(), sceScreamGetLocalVariableByHash(), sceScreamGetHashFromName(), SceScreamSoundParams, sceScreamSetSoundParamsEx(), sceScreamGetSoundParamsEx()

SceScreamSoundParams

Stores Sound-specific parameter values.

Definition

```
struct SceScreamSoundParams {
    uint32_t size;
    uint32_t mask;
    uint32_t flags;
    void *userCtx;
    float gain;
    uint32_t azimuth;
    uint32_t focus;
    int32_t pitchTranspose;
    float pitchBendFactor;
    int8_t registers[SCE_SCREAM_SND_MAX_REGISTERS];
    SceScreamSndLocalVarData *localVariableData;
    uint32_t listenerHandle;
    SceScreamSnd3DVector position;
    float dopplerFactor;
    SceScreamSynthParams synthParams;
};
```

Members

size — The size of this data structure; use `sizeof(SceScreamSoundParams)` to determine.

mask — A mask indicating which of the following members have active settings. One or more of the Sound Parameter Bit Masks. Use the bitwise `OR` operator for multiple selections.

flags — Zero or more of the following behavior options:
`SCE_SCREAM_SND_FLAG_DO_FINISHED_CALLBACK`,
`SCE_SCREAM_SND_FLAG_START_STREAM_PAUSED`,
`SCE_SCREAM_SND_FLAG_STREAM_GET_VOICE_LEVEL`,
`SCE_SCREAM_SND_FLAG_DIST_MODEL_NO_FILTER`,
`SCE_SCREAM_SND_FLAG_DIST_MODEL_PRESEND_FILTER_3`,
`SCE_SCREAM_SND_FLAG_DOPPLER_CAMERA_CUT`.

userCtx — A user-defined context value that will be passed back with any event callbacks made on this Sound instance. Can be `NULL`. See `SceScreamSndEventCallback()` and `SceScreamPlatformInitEx2`.

gain — Overall gain setting. Scales the levels of a Sound's `SceScreamSynthParams` *auxSendGain* and *directSendGain* members. Range: `SCE_SCREAM_SND_MIN_GAIN` to `SCE_SCREAM_SND_MAX_GAIN`.

azimuth — Panning azimuth of Sound, expressed in degrees clockwise relative to the listener. Range: 0 to 359. A value of 0 specifies that Sound is straight ahead; 90 specifies directly to the right; 180 specifies behind, and so on. Alternatively, you can specify a single speaker target using one of the Output Speaker Targets. See "Notes" below.

focus — Width of panning focus of Sound, expressed in degrees. Range: 0 to 360. A value of 0 specifies a point source; 360 specifies Sound coming from all directions. **Note:** Setting *focus* to 360 makes *azimuth* irrelevant for mono assets or multi-channel assets if Mix Mode is set to `Discrete` in Bank contents. Also, if *azimuth* is set to one or more specific Output Speaker Targets, *focus* is ignored. See "Notes" below.

pitchTranspose	Transposition amount in fines above/below original pitch. Range: -SCE_SCREAM_SND_MAX_PITCH_TRANSPOSE to +SCE_SCREAM_SND_MAX_PITCH_TRANSPOSE. Zero specifies original pitch.
pitchBendFactor	Pitchbend amount. Range: SCE_SCREAM_SND_MIN_PITCH_BEND_FACTOR to SCE_SCREAM_SND_MAX_PITCH_BEND_FACTOR. For details on setting pitchbend, see the "Notes" section of the sceScreamAutoPitchBend() function.
registers	Sound-specific register settings. An array of length SCE_SCREAM_SND_MAX_REGISTERS, with values in the range -128 to 127.
localVariableData	A pointer to a SceScreamSndLocalVarData structure containing values for one or more local variables of a Sound instance.
listenerHandle	Handle of a listener to which this Sound instance is relative in 3D space. See sceScreamCreateListener(). See "Notes" below.
position	A SceScreamSnd3DVector structure containing a Sound's world-space position. See "Notes" below.
dopplerFactor	A scalar for the Doppler effect that is automatically generated by 3D *position* changes relative to a specified *listenerHandle*. Scales the Doppler Factor property set on a Sound in Bank contents. Set to 0.0 for no Doppler. Set to 1.0 for a natural Doppler effect or to leave the content setting unchanged. Set to greater than 1.0 for an exaggerated effect.
synthParams	A SceScreamSynthParams data structure, holds Sound-specific synthesizer parameter settings.

Description

This structure stores Sound-specific parameter values. You use this structure when setting or retrieving Sound parameters with the sceScreamSetSoundParamsEx() and sceScreamGetSoundParamsEx() functions. You also use it when you play a sound with sceScreamPlaySoundByIndexEx() or sceScreamPlaySoundByNameEx().

Notes

Setting *focus* to 360 makes *azimuth* irrelevant for mono assets or multi-channel assets in which the Mix Mode property is set to Discrete in Bank contents. For multi-channel assets in which the Mix Mode property is set to Diffuse, *azimuth* does have an effect with respect to the image of the multi-channel asset. For example, setting *azimuth* to 180 degrees and *focus* to 360 degrees inverts the multi-channel image.

Setting *azimuth* to a speaker target overrides and nullifies any Bank contents pan settings. An *azimuth* setting expressed in degrees will be added (modulo 360) to any Bank contents pan settings.

pitchBendFactor settings have no effect on a Sound unless one or more of its component *Waveform* or *Stream* Grains has a corresponding upper/lower pitchbend range setting.

Sound distance calculations are based on SceScreamSnd3DVector coordinates set in the *position* member, relative to those set on a listener specified in the *listenerHandle* member.

See Also

sceScreamSetSoundParamsEx(), sceScreamGetSoundParamsEx(), sceScreamPlaySoundByIndexEx(), sceScreamPlaySoundByNameEx(), Sound Parameter Bit Masks, sceScreamCreateListener()

NGS Data Structures

Summary

NGS data structures store NGS-specific API data.

Item	Description
SceScreamSndDistortionParams	Stores parameter values for the pre-send distortion effect.
SceScreamSndIIRFilterParams	Stores parameter values for the pre-send and post-send filters.
SceScreamSndPremasterSubmixProps	Stores pre-master submix property values.
SceScreamSndReverbProps	Stores I3DL2 reverb parameter values.
SceScreamSynthParams	Stores synthesizer parameter values.
SceScreamSystemParams	Stores rendering synthesizer parameters values for the NGS synthesizer. This is embedded in the `SceScreamPlatformInitEx2` structure used when calling `sceScreamStartSoundSystemEx2()` to initialize Scream.

SceScreamSndDistortionParams

Stores parameter values for the pre-send distortion effect.

Definition

```
struct SceScreamSndDistortionParams {
    float a;
    float b;
    float clip;
    float gate;
    float wetGain;
    float dryGain;
    uint32_t flags;
};
```

Members

a	Distortion coefficient A. Range: SCE_SCREAM_SND_DIST_MIN_AB (0) to SCE_SCREAM_SND_DIST_MAX_AB (10.0).
b	Distortion coefficient B. Range: SCE_SCREAM_SND_DIST_MIN_AB (0) to SCE_SCREAM_SND_DIST_MAX_AB (10.0).
clip	Limiter on the audio output of the polynomial stage. Range: SCE_SCREAM_SND_DIST_MIN_CLIP_GATE (0) to SCE_SCREAM_SND_DIST_MAX_CLIP_GATE (1.0).
gate	Noise gate on the audio output. Range: SCE_SCREAM_SND_DIST_MIN_CLIP_GATE (0) to SCE_SCREAM_SND_DIST_MAX_CLIP_GATE (1.0).
wetGain	Wet (distorted signal) gain factor. Range: SCE_SCREAM_SND_DIST_MIN_GAIN (0) to SCE_SCREAM_SND_DIST_MAX_GAIN (4.0), where 1.0 is unity gain.
dryGain	Dry (original signal) gain factor. Range: SCE_SCREAM_SND_DIST_MIN_GAIN (0) to SCE_SCREAM_SND_DIST_MAX_GAIN (4.0), where 1.0 is unity gain.
flags	Must be set to SCE_SCREAM_SND_DISTORTION_ENABLED to enable the pre-send distortion effect.

Description

This structure stores parameter values for the pre-send distortion effect. You use this structure when setting or retrieving pre-send distortion effect values with the sceScreamSetSoundParamsEx() and sceScreamGetSoundParamsEx() functions. SceScreamSndDistortionParams is embedded in the SceScreamSynthParams structure, which in turn is embedded in the SceScreamSoundParams structure, which is referenced (in the *params* parameter) when calling the sceScreamSetSoundParamsEx() and sceScreamGetSoundParamsEx() functions.

For more information on distortion, see "Configuring the Distortion Effect" in the "Working with Pre- and Post-Send Filters and Effects" chapter of the *Scream Library Overview*.

See Also

sceScreamSetSoundParamsEx(), sceScreamGetSoundParamsEx(), SceScreamSynthParams, SceScreamSoundParams

SceScreamSndIIRFilterParams

Stores parameter values for the pre-send and post-send filters.

Definition

```
struct SceScreamSndIIRFilterParams {
    int32_t type;
    float gain;
    float cutoff;
    float resonance;
};
```

Members

type	Filter type. One of the Filter Modes Indices.
gain	Filter gain. Range (where applicable): SCE_SCREAM_BQ_MIN_GAIN to SCE_SCREAM_BQ_MAX_GAIN. For further information, see the "Working with Pre- and Post-Send Filters and Effects" chapter of the *Scream Library Overview*.
cutoff	Filter cut-off frequency. Range 10.0 to 23999.0 (that is, less than the half sampling rate).
resonance	Filter resonance or Q. Range: SCE_SCREAM_BQ_MIN_RESONANCE to SCE_SCREAM_BQ_MAX_RESONANCE.

Description

This structure stores parameter values for the pre-send and post-send biquad filters. You use this structure when setting or retrieving pre- and post-send filter values with the sceScreamSetSoundParamsEx() and sceScreamGetSoundParamsEx() functions. Several SceScreamSndIIRFilterParams structures are embedded in the SceScreamSynthParams structure, which in turn is embedded in the SceScreamSoundParams structure, which is referenced (in the *params* parameter) when calling the sceScreamSetSoundParamsEx() and sceScreamGetSoundParamsEx() functions.

Biquad filters are infinite impulse response (IIR) filters with a roll-off of 12dB per octave.

Note that the specific meaning of the parameters varies with the different filter types. And in the case of the SCE_SCREAM_FLT_BQ_LPF, SCE_SCREAM_FLT_BQ_HPF, SCE_SCREAM_FLT_BQ_APF, SCE_SCREAM_FLT_BQ_BPF, and SCE_SCREAM_FLT_BQ_NOTCH filters, the *gain* parameter is not applicable.

For important information on using the pre-send and post-send filters, see "Available Filter Types and Their Parameters" in the "Working with Pre- and Post-Send Filters and Effects" chapter of the *Scream Library Overview*.

Run the Scream example program, Runtime Filter, to hear the effect of different parameter values for the various filter types. For further information on Scream example programs, see the introductory chapter of the *Scream Library Overview*.

See Also

sceScreamSetSoundParamsEx(), sceScreamGetSoundParamsEx(), SceScreamSynthParams, SceScreamSoundParams

SceScreamSndPremasterSubmixProps

Stores pre-master submix property values.

Definition

```
struct SceScreamSndPremasterSubmixProps {
    BOOL compEffectOn;
    BOOL compLinkedChannels;
    BOOL compPeakMode;
    float compThresholdDB;
    float compRatio;
    float compAttackTimeMS;
    float compReleaseTimeMS;
    float compMakeupGainDB;
    float compSoftKneeDB;
};
```

Members

compEffectOn	Compressor enable. A Boolean value that specifies whether the compressor is on or off. `TRUE` for on; `FALSE` for off.
compLinkedChannels	Compressor channel linking. A Boolean value that specifies whether to link the input channels to retain the original panning image. `TRUE` for channel linking; `FALSE` for channel independence. See "Notes" below.
compPeakMode	Compressor peak mode. A Boolean value that specifies whether or not to use peak mode. `TRUE` for peak mode; `FALSE` for RMS mode. See "Notes" below.
compThresholdDB	Compressor operation threshold. Expressed in dB. Range: -100.0 to 0.0.
compRatio	Compression ratio. For input signal compression, use values greater than 1.0; for expansion, use values less than 1.0. For example, for a compression ratio of 2:1, use 2.0. Must be greater than 0.0 and less than or equal to 10.0.
compAttackTimeMS	Compressor attack time. Expressed in milliseconds. Range: 0 to 10.0. Typically within the range of around 0 to 5 milliseconds.
compReleaseTimeMS	Compressor release time. Expressed in milliseconds. Range: 0 to 2000.0. Typically within the range of around 500 to 1000 milliseconds.
compMakeupGainDB	Compressor output make-up gain. Expressed in dB. Range: -100.0 to 100.0.
compSoftKneeDB	Compressor soft knee. Defines the limits of an amplitude range, centered around the *compThresholdDB* level, over which the compression response curve operates. Range: -100.0 to 0.0 dB, where 0 dB produces no softening of the compression response curve (that is, a 'hard knee').

Description

This structure stores pre-master submix property values. You use this structure when setting all pre-master submix buss effect properties directly using the `sceScreamPremasterSubmixSetAllProperties()` function.

You can also set pre-master submix property values with the `sceScreamSynthPremasterSubmixSetupCompressor()` function. Its parameters are nearly identical to the members in `SceScreamSndPremasterSubmixProps`.

Notes

The compressor property values apply to pre-master submix effects with both standard and side-chain compressors.

Channel linking preserves the panning image, but is a more intrusive compression mode because each channel is compressed equally based on the *compPeakMode* setting.

In peak mode, the compressor responds to the instantaneous level of the input signal. Peak mode can produce more quick-reacting and obvious results. In RMS mode, the compressor responds to an averaged level of the input signal. RMS mode can produce more relaxed and subtle results.

See Also

sceScreamPremasterSubmixSetAllProperties(),
sceScreamSynthPremasterSubmixSetupCompressor()

SceScreamSndReverbProps

Stores I3DL2 reverb parameter values.

Definition

```
struct SceScreamSndReverbProps {
    float Room;
    float Room_HF;
    float Decay_time;
    float Decay_HF_ratio;
    float Reflections;
    float Reflections_delay;
    float Reverb;
    float Reverb_delay;
    float Diffusion;
    float Density;
    float HF_reference;
    int32_t EarlyReflectionPattern[2];
    float EarlyReflectionScalar;
    float LF_reference;
    float Room_LF;
    float DryMB;
};
```

Members

Room	Output level of wet (reverb-treated) signal, in mB (milliBells; 100 mB = 1 dB). Range: -10000.0 to 0.0 mB.
Room_HF	Attenuation of high-frequencies, in mB, with respect to the *HF_reference* value. Range -10000.0 to 0.0 mB, where 0.0 mB produces no coloration.
Decay_time	Late reverberation (diffuse tail) decay time. Larger, more reflective environments have longer decay times. Smaller, less reflective environments have shorter decay times. Range: 0.1 to 20 seconds.
Decay_HF_ratio	Ratio of late reverberation high-frequency decay to low-frequency decay with respect to the *HF_reference* value. Range: 0.1 to 2.0.
Reflections	Early reflections level, in mB. Use this parameter in conjunction with the *Reverb* value to set the balance between early reflections and late reverberation. Range: -10000 to +1000 mB.
Reflections_delay	Pre-delay time before the onset of early reflections. Simulates room size. Larger spaces have a longer pre-delay before the onset of early reflections. Range: 0.0 to 0.3 seconds.
Reverb	Late reverberation level, in mB. Use this parameter in conjunction with the *Reflections* value to set the balance between early reflections and late reverberation. Range: -10000 to +2000 mB.
Reverb_delay	Delay time before the onset of late reverberation. Simulates room size; larger spaces have a longer delay before the onset of late reverberation. Range: 0.0 to 0.1 seconds.
Diffusion	Echo density of late reverberation. Simulates the reflectivity of the environment surfaces. Range: 0 to 100%, where 0% means minimal reflection.
Density	Modal density of late reverberation. Range: 0 to 100%, where 0% means minimal late reverberation.
HF_reference	Reference high frequency. Used in conjunction with *Room_HF* and *Decay_HF_ratio*. Range: 20 to 20000 Hertz.

EarlyReflectionPattern	(Optional) One of the Early Reflections Pattern Constants per channel. `EarlyReflectionPattern[0]` is for the left and `EarlyReflectionPattern[1]` is for the right channel.
EarlyReflectionScalar	Scales the early reflections spread. Higher values produce more spread-out early reflections (as found in larger spaces); lower values produce more tightly-packed early reflections (as found in smaller spaces). Range: 0 to 100%, where 0% specifies single reflection, and 100% specifies widely spread reflections.
LF_reference	Reference low frequency. Used in conjunction with *Room_LF*. Range: 20 to 20000 Hertz.
Room_LF	Attenuation of low-frequencies with respect to the *LF_reference* value. Range: -10000 to 0 mB.
DryMB	Output level of dry (untreated) signal, in mB. Range: -10000 to 0 mB.

Description

This structure stores parameter values used in the I3DL2 reverb. You can set I3DL2 reverb parameters directly by storing desired values in this structure, and calling the `sceScreamReverbSetAllProperties()` function.

You can also set I3DL2 reverb parameters by setting a stock preset. To set a stock preset, use the `sceScreamReverbSetStockPreset()` function, and select from the list of `SceScreamI3DL2StockPresets`.

See Also

`sceScreamReverbSetAllProperties()`, `SceScreamI3DL2StockPresets`, `sceScreamReverbSetStockPreset()`

SceScreamSynthParams

Stores synthesizer parameter values.

Definition

```
struct SceScreamSynthParams {
    uint32_t mask;
    SceScreamSndIIRFilterParams preSendFilter0;
    SceScreamSndIIRFilterParams preSendFilter1;
    SceScreamSndIIRFilterParams preSendFilter2;
    SceScreamSndIIRFilterParams preSendFilter3;
    SceScreamSndDistortionParams preSendDistortion;
    float auxSendGain[SCE_SCREAM_NUM_AUX_SENDS];
    uint32_t auxSendDests[SCE_SCREAM_NUM_AUX_SENDS];
    float directSendGain;
    SceScreamSndIIRFilterParams postSendFilter;
    float lfeGain;
};
```

Members

mask	A bit mask value indicating which of the following members are active. One or more of the Synthesizer Parameter Bit Masks. Use the bitwise OR operator for multiple selections.
preSendFilter0	A SceScreamSndIIRFilterParams structure, initialized with parameter values for pre-send filter 0.
preSendFilter1	A SceScreamSndIIRFilterParams structure, initialized with parameter values for pre-send filter 1.
preSendFilter2	A SceScreamSndIIRFilterParams structure, initialized with parameter values for pre-send filter 2.
preSendFilter3	A SceScreamSndIIRFilterParams structure, initialized with parameter values for pre-send filter 3.
preSendDistortion	A SceScreamSndDistortionParams structure, initialized with parameter values for the pre-send distortion effect.
auxSendGain	An array of auxiliary send gain values, of length SCE_SCREAM_NUM_AUX_SENDS. These values scale the level of signal that is branched into a Sound's respective auxiliary send busses. Range: SCE_SCREAM_SND_MIN_GAIN to SCE_SCREAM_SND_MAX_GAIN.
auxSendDests	An array of auxiliary send destinations, of length SCE_SCREAM_NUM_AUX_SENDS. Range: 0 to (SCE_SCREAM_NUM_AUX_BUSSES – 1). For details, see "Configuring Auxiliary Sends" in the "Working with the I3DL2 Reverb" chapter of the *Scream Library Overview*.
directSendGain	Direct send gain. This value scales the level of a Sound's direct path signal as it passes through the pre- and post-send filters. Range: SCE_SCREAM_SND_MIN_GAIN to SCE_SCREAM_SND_MAX_GAIN.
postSendFilter	A SceScreamSndIIRFilterParams structure, initialized with parameter values for the post-send filter.
lfeGain	LFE (low frequency effect) gain. Range: SCE_SCREAM_SND_MIN_GAIN to SCE_SCREAM_SND_MAX_GAIN.

Description

This structure stores synthesizer parameter values. You use it when setting or retrieving synthesizer parameter values through the sceScreamSetSoundParamsEx() and

sceScreamGetSoundParamsEx() functions. These functions take a SceScreamSoundParams parameter, which contains a SceScreamSynthParams parameter.

The *preSendFilter0*, *preSendFilter1*, *preSendFilter2*, *preSendFilter3*, and *postSendFilter* members point to SceScreamSndIIRFilterParams structures, initialized with parameter values for the respective pre-send and post-send filters. The *preSendDistortion* points to a SceScreamSndDistortionParams structure, initialized with parameter values for the pre-send distortion effect.

On the NGS synthesizer, there are SCE_SCREAM_NUM_AUX_SENDS (3) auxiliary sends with fixed routing to SCE_SCREAM_NUM_AUX_BUSSES (3) corresponding auxiliary busses. The *auxSendGain* member is an array of gain values for each auxiliary send. Because auxiliary sends have fixed destinations, there is no need to set auxiliary send destinations. On the NGS synth, settings to the *auxSendDests* member are ignored.

Notes

Setting any of the pre-send filters (filters 0 to 3, and the distortion effect) from the Scream API overrides any settings on a Sound that may have been made in Bank contents. It is generally good practice to coordinate filter settings with your audio design team.

See Also

SceScreamSndIIRFilterParams, SceScreamSndDistortionParams, SceScreamSoundParams, sceScreamSetSoundParamsEx(), sceScreamGetSoundParamsEx()

SceScreamSystemParams

Stores rendering synthesizer parameters values for the NGS synthesizer. This is embedded in the SceScreamPlatformInitEx2 structure used when calling sceScreamStartSoundSystemEx2() to initialize Scream.

Definition

```
struct SceScreamSystemParams {
    int32_t tickThreadPriority;
    int32_t tickThreadAffinity;
    uint32_t tickThreadStackSize;
    uint32_t voiceTypeCount[SCE_SCREAM_SND_VOICE_DATA_TYPE_COUNT];
    uint32_t numReverbs;
    uint32_t numPremasterCompSubmixes;
    uint32_t numPremasterScCompSubmixes;
    uint32_t numExternalRacks;
    uint32_t numExternalVoices;
    uint32_t initFlags;
    uint32_t reserved0;
};
```

Members

tickThreadPriority	Priority value to use when creating the synthesizer thread that ticks the Scream runtime. Defaults to SCE_SCREAM_SND_DEFAULT_THREAD_PRIORITY.
tickThreadAffinity	CPU affinity with which to create the synthesizer thread that ticks the Scream runtime. Range: -1 to 2. A value of -1 indicates all cores. Defaults to SCE_SCREAM_SND_DEFAULT_THREAD_AFFINITY.
tickThreadStackSize	Size of the stack used for audio output and the Scream tick thread. Defaults to SCE_SCREAM_SND_DEFAULT_THREAD_STACK_SIZE.
voiceTypeCount	A count of the maximum number of voices of each respective type to allocate on the synthesizer. Expressed as an array, of length SCE_SCREAM_SND_VOICE_DATA_TYPE_COUNT, indexed in the order of the Voice Data Type Constants. Range for each voice type: 0 to SceScreamPlatformInitEx2.*maxPolyphony*. Defaults to [SCE_SCREAM_SND_DEFAULT_NUM_VAG_MONO_VOICES, SCE_SCREAM_SND_DEFAULT_NUM_PCM_MONO_VOICES, SCE_SCREAM_SND_DEFAULT_NUM_AT9_MONO_VOICES, SCE_SCREAM_SND_DEFAULT_NUM_VAG_STEREO_VOICES, SCE_SCREAM_SND_DEFAULT_NUM_PCM_STEREO_VOICES, SCE_SCREAM_SND_DEFAULT_NUM_AT9_STEREO_VOICES]. See "Notes" below.
numReverbs	The number of I3DL2 reverb voices to instantiate. Range: 0 to SCE_SCREAM_SND_MAX_REVERBS. Defaults to SCE_SCREAM_SND_DEFAULT_NUM_REVERBS.
numPremasterCompSubmixes	The number of pre-master compressor submix voices to instantiate. Range: 0 to SCE_SCREAM_SND_MAX_PREMASTER_SUBMIXES. Defaults to SCE_SCREAM_SND_DEFAULT_NUM_PREMASTER_COMP_SUBMIXES.
numPremasterScCompSubmixes	The number of pre-master side-chain compressor submix voices to instantiate. Range: 0 to (SCE_SCREAM_SND_MAX_PREMASTER_SUBMIXES minus *numPremasterCompSubmixes*). Defaults to SCE_SCREAM_SND_DEFAULT_NUM_PREMASTER_SC_COMP_SUBMIXES.

numExternalRacks	The number of NGS racks to create. Note that NGS racks must be managed outside of Scream. Defaults to zero.
numExternalVoices	The number of voices across all external racks to create. Note that NGS racks must be managed outside of Scream. Defaults to zero.
initFlags	Any combination of the NGS Initialization Flags. Defaults to SCE_SCREAM_SND_DEFAULT_SYNTH_INIT_FLAGS.
reserved0	Reserved for internal use. Set to 0.

Description

This structure stores parameter values for the NGS rendering synthesizer running on PlayStation®Vita. It is embedded in the SceScreamPlatformInitEx2 structure in the *synthParams* member. And, along with SceScreamPlatformInitEx2, it is initialized with default values with a call to sceScreamFillDefaultScreamPlatformInitArgsEx2().

Notes

The *voiceTypeCount* member takes an array of values specifying a maximum number of voices to allocate for each respective voice type. In the event that an application requests more voices at one time than are available on the synthesizer, voice management is controlled by prioritization. It is permissible for the combined total of *voiceTypeCount* array values to be greater than the number of voices that Scream can simultaneously address, as returned by sceScreamGetMaxPolyphony(). For further details, see "Allocating Synthesizer Voice Types" in the "Configuration, Initialization, and Shutdown" chapter of the *Scream Library Overview*.

See Also

SceScreamPlatformInitEx2, sceScreamFillDefaultScreamPlatformInitArgsEx2(), sceScreamGetMaxPolyphony()

Scream Type Definitions

Summary

Scream type definitions define data types for APIs that are common to all supported platforms and synthesizers.

Member	Description
SceScreamExternSndMemAlloc	Prototype for a Scream memory allocation callback function.
SceScreamExternSndMemFree	Prototype for a Scream memory release callback function.
SceScreamSndDebugHandler	Prototype for a Scream debug text output function.
SceScreamSndEventCallback	Prototype for the Scream event callback function.

SceScreamExternSndMemAlloc

Prototype for a Scream memory allocation callback function.

Definition

```
typedef void * (*SceScreamExternSndMemAlloc)(
    int32_t bytes,
    int32_t use
);
```

Arguments

bytes	(Input) The number of bytes required.
use	(Input) The type of memory required. One of the Memory Allocation Constants.

Return Values

The return value is a `void` pointer to the allocated memory.

Description

Scream invokes the `SceScreamExternSndMemAlloc()` callback function prototype whenever memory allocation is required. This provides a way to customize memory allocation functionality. The required type of memory allocation is passed to the function, allowing you to determine how best to allocate the memory.

Notes

Allocated memory should be 16-byte aligned.

See Also

`SceScreamExternSndMemFree()`

SceScreamExternSndMemFree

Prototype for a Scream memory release callback function.

Definition

```
typedef void (*SceScreamExternSndMemFree)(
    void *mem
);
```

Arguments

mem	(Input) Void pointer to memory allocated by the prototype SceScreamExternSndMemAlloc() function.

Return Values

None

Description

Scream invokes the SceScreamExternSndMemFree() callback function prototype whenever memory release is required. This provides a way to customize memory release functionality.

See Also

SceScreamExternSndMemAlloc()

SceScreamSndDebugHandler

Prototype for a Scream debug text output function.

Definition

```
typedef void (*SceScreamSndDebugHandler)(
    const char *message
);
```

Arguments

message	(Input) A pre-formatted text string containing debug or general information.

Return Values

None

Description

Scream invokes this function prototype whenever text output occurs, enabling applications to funnel all TTY output to a single location (for example, screen, log file, custom application, and so on). Set this function by calling `sceScreamSetDebugHandler()`.

Notes

If no application-defined `SceScreamSndDebugHandler()` is specified, a standard `printf()`-like function is used by default.

You can set this application-defined `SceScreamSndDebugHandler()` prior to initializing Scream. This is recommended, as much information is produced during the initialization process.

See Also

`sceScreamSetDebugHandler()`, `sceScreamOutputHandlerInfoToTTY()`, `sceScreamOutputAllPlayingSoundInfoToTTY()`

Document serial number: 000004892117

SceScreamSndEventCallback

Prototype for the Scream event callback function.

Definition

```
typedef void (*SceScreamSndEventCallback)(
    uint32_t handle,
    void *userCtx,
    uint32_t reason
);
```

Arguments

handle	(Output) Handle of the Sound or Effect for which the callback is being issued.
userCtx	(Output) The user context value most recently set in the Sound's SceScreamSoundParams structure's *userCtx* member, or NULL if the handle is for an Effect instance.
reason	(Output) One of the Callback Constants specifying the reason this callback is being issued.

Return Values

None

Description

This is a prototype for the Scream event callback function. If registered in the SceScreamPlatformInitEx2 *eventCallback* member, Scream invokes this callback function whenever an event takes place for which a given Sound or Effect instance has requested a callback. The Sound or Effect handle, along with a user-specified context value, and a reason value are passed to the client.

For information on using this callback, see "Configuring Sound Event Callbacks" in the "Working with Sounds" chapter, and "Configuring Per-Tick Callbacks" in the "Working with System Globals" chapter of the *Scream Library Overview*.

Notes

WARNING: Because this function may be called from the synthesizer thread, it is critical that processing performed within this callback be kept to an absolute minimum! Otherwise, audio dropouts may occur.

See Also

SceScreamPlatformInitEx2, SceScreamSoundParams

NGS Type Definitions

Summary

NGS type definitions define data types for NGS-specific APIs.

Member	Description
SceScreamIniHandle	Data type for an INI-formatted presets file handle.
SceScreamReverbHandle	Data type for a reverb handle.

SceScreamIniHandle

Data type for an INI-formatted presets file handle.

Definition

```
typedef void *SceScreamIniHandle;
```

Description

Defines the data type used for an INI-formatted presets file handle. You obtain a SceScreamIniHandle by calling sceScreamPresetFileLoad() or sceScreamPresetFileLoadFromMem(). This handle is used by the Presets File (INI) Functions.

See Also

sceScreamPresetFileLoad(), sceScreamPresetFileLoadFromMem(), sceScreamPremasterSubmixSetCustomPreset(), sceScreamReverbSetCustomPresetByName()

SceScreamReverbHandle

Data type for a reverb handle.

Definition

```
typedef uint32_t SceScreamReverbHandle;
```

Description

Defines the data type used for a reverb handle. Obtain a handle with the sceScreamReverbGetHandleByBuss() function. This handle is used by the General Reverb Functions.

See Also

sceScreamReverbGetHandleByBuss()

Scream System Functions

Summary

Scream system functions control system features, set and retrieve system states, and are common to all supported platforms and synthesizers.

Function	Description
sceScreamAddGlobalVariable	Adds a global variable to the system.
sceScreamAddSetGlobalVariable	Sets a global variable value, first adding it to the system if it does not yet exist.
sceScreamDeleteGlobalVariable	Deletes a global variable from the system.
sceScreamFillDefaultScreamPlatformInitArgsEx2	Initializes a SceScreamPlatformInitEx2 data structure with default values.
sceScreamGetAllocatedVoiceCountByType	Retrieves a count of allocated synthesizer voices by type.
sceScreamGetGlobalVariableByHash	Retrieves the value of a global variable.
sceScreamGetGlobalVariableByIndex	Retrieves the value of a global variable.
sceScreamGetGlobalVariableByName	Retrieves the value of a global variable.
sceScreamGetHashFromName	Calculates a 32-bit string hash.
sceScreamGetMasterOutputLevel	Retrieves the current level of a master output channel.
sceScreamGetMaxPolyphony	Retrieves the maximum number of synthesizer voices that can be simultaneously played by Scream.
sceScreamGetNumGlobalVariables	Retrieves the current number of global variables.
sceScreamGetPlaybackMode	Retrieves the current playback mode.
sceScreamGetRandomIndex	Retrieves the current seed index of the Scream random number generator.
sceScreamGetScriptSpeedFactor	Retrieves the global script speed factor for variable speed replays.
sceScreamGetSFXGlobalReg	Retrieves the value of a Scream global register.
sceScreamGetStreamingFileDirectory	Retrieves the associated directory path string for a specified stream path group.
sceScreamGetSynthName	Retrieves the name of the rendering synthesizer with which Scream is linked.
sceScreamGetSystemRunning	Confirms whether Scream has been initialized and is running.
sceScreamGetTick	Retrieves the current Scream tick count.
sceScreamGetVoiceTypeName	Retrieves the name of a synthesizer voice type based on a specified data type.
sceScreamSetDebugHandler	Sets a custom debug text output function.
sceScreamSetGlobalVariableByHash	Sets the value of a global variable.
sceScreamSetGlobalVariableByIndex	Sets the value of a global variable.
sceScreamSetGlobalVariableByName	Sets the value of a global variable.
sceScreamSetMinRipoffTime	Changes the system-wide minimum voice ripoff time.
sceScreamSetPlaybackMode	Sets the playback mode.
sceScreamSetRandomIndex	Sets the seed index of the Scream random number generator.
sceScreamSetScriptSpeedFactor	Sets the global script speed factor for variable speed replays.
sceScreamSetSFXGlobalReg	Sets the value of a Scream global register.
sceScreamSetStreamingFileDirectory	Sets the directory path string for a specified stream path group.

Function	Description
sceScreamStartSoundSystemEx2	Initializes Scream for use by an application.
sceScreamStopAllSounds	Stops all Sounds in the active Scream Sound handlers list.
sceScreamStopAllSoundsByIndex	Stops all instances of a Sound as specified by Bank and index references.
sceScreamStopSoundSystem	Shuts down the Scream runtime engine.

sceScreamAddGlobalVariable

Adds a global variable to the system.

Definition

```
bool sceScreamAddGlobalVariable(
    const char *name,
    uint32_t *outVarIndex
);
```

Arguments

name	(Input) Name assigned to a global variable to add.
outVarIndex	(Output) An optional pointer in which to store the index associated with the global variable. You use this index to reference the associated global variable with the sceScreamSetGlobalVariableByIndex() and sceScreamGetGlobalVariableByIndex() functions. See "Notes" below.

Return Values

Returns TRUE if the specified global variable is created. Returns FALSE if a global variable with the specified name already exists.

Description

This function adds a global variable to the system.

Notes

The *outVarIndex* is valid whether the global variable exists prior to this call or not.

See Also

sceScreamAddSetGlobalVariable(), sceScreamSetGlobalVariableByIndex(), sceScreamGetGlobalVariableByIndex(), sceScreamDeleteGlobalVariable()

sceScreamAddSetGlobalVariable

Sets a global variable value, first adding it to the system if it does not yet exist.

Definition

```
bool sceScreamAddSetGlobalVariable(
    const char *name,
    float val,
    uint32_t *outVarIndex
);
```

Arguments

name	(Input) Name of the global variable to add/set.
val	(Input) Value to set the specified global variable to.
outVarIndex	(Output) An optional pointer in which to store the index associated with the global variable. You use this index to reference the associated global variable with the sceScreamSetGlobalVariableByIndex() and sceScreamGetGlobalVariableByIndex() functions. See "Notes" below.

Return Values

Returns TRUE if the specified global variable is created or if a global variable with the specified name already exists. Returns FALSE if the maximum number of global variables already exists or *name* is NULL.

Description

This function sets a global variable value. If the global variable does not yet exist, the function adds it to the system before setting its value.

Notes

The *outVarIndex* is valid whether the global variable exists prior to this call or not.

See Also

sceScreamAddGlobalVariable(), sceScreamSetGlobalVariableByIndex(), sceScreamGetGlobalVariableByIndex(), sceScreamDeleteGlobalVariable()

sceScreamDeleteGlobalVariable

Deletes a global variable from the system.

Definition

```
bool sceScreamDeleteGlobalVariable(
    const char *name
);
```

Arguments

name (Input) Name of the global variable to delete.

Return Values

Returns TRUE if the global is successfully deleted. Returns FALSE otherwise.

Description

This function deletes a global variable from the system.

See Also

sceScreamAddSetGlobalVariable(), sceScreamAddGlobalVariable()

sceScreamFillDefaultScreamPlatformInitArgsEx2

Initializes a SceScreamPlatformInitEx2 data structure with default values.

Definition

```
int32_t sceScreamFillDefaultScreamPlatformInitArgsEx2(
    SceScreamPlatformInitEx2 *platformInit
);
```

Arguments

platformInit	(Input) Pointer to a SceScreamPlatformInitEx2 structure to be initialized.

Return Values

Returns SCE_SCREAM_SS_ERROR_OK if the operation was successful. Returns SCE_SCREAM_SS_ERROR_INVALID_PARAMETER if *platformInit* is NULL or if the *size* member of the SceScreamPlatformInitEx2 structure specified for the *platformInit* parameter is invalid.

Description

SceScreamPlatformInitEx2 is a data structure used to store initialization values for the Scream platform. This function initializes a SceScreamPlatformInitEx2 structure – including the SceScreamSystemParams structure embedded in its *synthParams* member – with the following Scream and synthesizer-specific default values.

size	No default
initFlags	SCE_SCREAM_SND_DEFAULT_INIT_FLAGS
playbackMode	SCE_SCREAM_SND_DEFAULT_PLAYBACK_MODE
maxLFOs	SCE_SCREAM_SND_DEFAULT_MAX_LFOS
lfoUpdateRate	SCE_SCREAM_SND_DEFAULT_LFO_UPDATE
duckerUpdateRate	SCE_SCREAM_SND_DEFAULT_DUCKER_UPDATE
groupMixerUpdateRate	SCE_SCREAM_SND_DEFAULT_GROUP_MIXER_UPDATE
ccSoundUpdateRate	SCE_SCREAM_SND_DEFAULT_CCSOUND_UPDATE
minRipoffTime	SCE_SCREAM_SND_DEFAULT_MIN_RIPOFF_TIME
memAlloc	SCE_SCREAM_SND_DEFAULT_MEM_ALLOC
memFree	SCE_SCREAM_SND_DEFAULT_MEM_FREE
eventCallback	NULL
maxBanks	SCE_SCREAM_SND_DEFAULT_MAX_BANKS
maxPolyphony	SCE_SCREAM_SND_DEFAULT_MAX_POLYPHONY
maxActiveSnapshots	SCE_SCREAM_SND_DEFAULT_MAX_SNAPSHOTS
maxGlobalVariables	SCE_SCREAM_SND_DEFAULT_MAX_GLOBAL_VARIABLES
maxCCSounds	SCE_SCREAM_SND_DEFAULT_MAX_CCSOUNDS
dopplerSlewRate	SCE_SCREAM_SND_DEFAULT_DOPPLER_SLEW_RATE
pGroupMixerFile	NULL
groupMixerFileSize	0
pBussConfigFile	NULL
bussConfigFileSize	0
pDistanceModelFile	NULL
synthParams	
tickThreadPriority	SCE_SCREAM_SND_DEFAULT_THREAD_PRIORITY
tickThreadAffinity	SCE_SCREAM_SND_DEFAULT_THREAD_AFFINITY
tickThreadStackSize	SCE_SCREAM_SND_DEFAULT_THREAD_STACK_SIZE

voiceTypeCount	[SCE_SCREAM_SND_DEFAULT_NUM_VAG_MONO_VOICES, SCE_SCREAM_SND_DEFAULT_NUM_PCM_MONO_VOICES, SCE_SCREAM_SND_DEFAULT_NUM_AT9_MONO_VOICES, SCE_SCREAM_SND_DEFAULT_NUM_VAG_STEREO_VOICES, SCE_SCREAM_SND_DEFAULT_NUM_PCM_STEREO_VOICES, SCE_SCREAM_SND_DEFAULT_NUM_AT9_STEREO_VOICES]
numReverbs	SCE_SCREAM_SND_DEFAULT_NUM_REVERBS
numPremasterCompSubmixes	SCE_SCREAM_SND_DEFAULT_NUM_PREMASTER_COMP_SUBMIXES
numPremasterScCompSubmixes	SCE_SCREAM_SND_DEFAULT_NUM_PREMASTER_SC_COMP_SUBMIXES
numExternalRacks	0
numExternalVoices	0
initFlags	SCE_SCREAM_SND_DEFAULT_SYNTH_INIT_FLAGS
reserved0	0

See Also

SceScreamPlatformInitEx2, SceScreamSystemParams, sceScreamStartSoundSystemEx2()

sceScreamGetAllocatedVoiceCountByType

Retrieves a count of allocated synthesizer voices by type.

Definition

```
uint32_t sceScreamGetAllocatedVoiceCountByType(
    uint32_t dataType
);
```

Arguments

dataType (Input) One of the NGS Voice Data Type Constants.

Return Values

Returns the number of allocated voices of the specified type.

Description

The number of allocated synthesizer voices of each type depends on specified values for the SceScreamSystemParams structure's *voiceTypeCount* member (an array), which is indexed by the Voice Data Type Constants. The SceScreamSystemParams structure is embedded in the SceScreamPlatformInitEx2 structure's *synthParams* member, and in turn passed to the sceScreamStartSoundSystemEx2() function when initializing Scream.

See Also

SceScreamSystemParams, SceScreamPlatformInitEx2, sceScreamGetMaxPolyphony(), sceScreamGetVoiceTypeName()

sceScreamGetGlobalVariableByHash

Retrieves the value of a global variable.

Definition

```
bool sceScreamGetGlobalVariableByHash(
    uint32_t nameHash,
    float *outVal
);
```

Arguments

nameHash	(Input) Hash of a global variable from which to retrieve a value.
outVal	(Output) A pointer to a floating-point variable in which to store the retrieved value.

Return Values

Returns `TRUE` if the variable specified by *nameHash* is valid. Returns `FALSE` otherwise.

Description

This function retrieves the value of a global variable, specified by hash.

See Also

sceScreamGetGlobalVariableByIndex(), sceScreamAddGlobalVariable(), sceScreamAddSetGlobalVariable(), sceScreamGetGlobalVariableByName(), sceScreamGetHashFromName()

sceScreamGetGlobalVariableByIndex

Retrieves the value of a global variable.

Definition

```
bool sceScreamGetGlobalVariableByIndex(
    uint32_t varIndex,
    float *outVal
);
```

Arguments

varIndex	(Input) Index of a global variable from which to retrieve a value. A value stored in the *outVarIndex* parameter resulting from calls to any of the following functions: sceScreamAddGlobalVariable(), sceScreamAddSetGlobalVariable().
outVal	(Output) A pointer to a floating-point variable in which to store the retrieved value.

Return Values

Returns TRUE if the specified *index* is valid. Returns FALSE otherwise.

Description

This function retrieves the value of a global variable, specified by index.

See Also

sceScreamSetGlobalVariableByIndex(), sceScreamAddGlobalVariable(), sceScreamAddSetGlobalVariable()

sceScreamGetGlobalVariableByName

Retrieves the value of a global variable.

Definition

```
bool sceScreamGetGlobalVariableByName(
    const char *name,
    float *outVal
);
```

Arguments

name	(Input) Name of a global variable from which to retrieve a value.
outVal	(Output) A pointer to a floating-point variable in which to store the retrieved value.

Return Values

Returns `TRUE` if the variable specified by *name* is valid. Returns `FALSE` otherwise.

Description

This function retrieves the value of a global variable, specified by name.

See Also

sceScreamGetGlobalVariableByIndex(), sceScreamGetGlobalVariableByHash(), sceScreamAddGlobalVariable(), sceScreamAddSetGlobalVariable()

sceScreamGetHashFromName

Calculates a 32-bit string hash.

Definition

```
uint32_t sceScreamGetHashFromName(
    const char *name
);
```

Arguments

name	(Input) String from which to calculate a hash.

Return Values

Returns the 32-bit hash of the specified string.

Description

This function calculates a 32-bit hash based on a specified string.

See Also

sceScreamSetLocalVariableByHash(), sceScreamGetLocalVariableByHash()

sceScreamGetMasterOutputLevel

Retrieves the current level of a master output channel.

Definition

```
float sceScreamGetMasterOutputLevel(
    uint32_t channel,
    bool rms,
    bool linear
);
```

Arguments

channel	(Input) The channel output level to retrieve. One of the Polar-Pan Speaker Channel Indices Constants.
rms	(Input) Currently, the result is an instantaneous peak level only (FALSE). Attempting to set to TRUE (for an averaged RMS level) is ignored.
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 *linear* is set to TRUE, returns a linear value that should fall between SCE_SCREAM_SND_LEVEL_LINEAR_MINIMUM and SCE_SCREAM_SND_LEVEL_LINEAR_NOMINAL, where values ≥ SCE_SCREAM_SND_LEVEL_LINEAR_NOMINAL indicate clipping. Otherwise, returns a value in decibels that should fall between SCE_SCREAM_SND_LEVEL_DB_MINIMUM and SCE_SCREAM_SND_LEVEL_DB_NOMINAL, where values ≥ SCE_SCREAM_SND_LEVEL_DB_NOMINAL indicate clipping.

Description

This function retrieves the instantaneous peak level of a specified master output channel. It returns the result either as a linear value or as a decibel (dB) value relative to full-scale.

Notes

The sceScreamGetMasterOutputLevel() function will not operate if Scream is initialized with the SCE_SCREAM_SND_SYNTH_INIT_FLAG_DISABLE_LEVELS flag in the SceScreamSystemParams *initFlags* member.

See Also

SceScreamSystemParams, SceScreamPlatformInitEx2

sceScreamGetMaxPolyphony

Retrieves the maximum number of synthesizer voices that can be simultaneously played by Scream.

Definition

```
uint32_t sceScreamGetMaxPolyphony(void);
```

Return Values

Returns the maximum polyphony (number of simultaneously addressable voices).

Description

The number of simultaneously playable voices on a synthesizer is known as polyphony. This function retrieves the maximum number of voices that can be simultaneously played by Scream with respect to the NGS synthesizer. You set the maximum polyphony at initialization time using the SceScreamPlatformInitEx2 structure's *maxPolyphony* member. The default maximum polyphony is defined by the constant SCE_SCREAM_SND_DEFAULT_MAX_POLYPHONY. Calling this function after initializing Scream (with a call to sceScreamStartSoundSystemEx2()) returns the value set in the SceScreamPlatformInitEx2 structure's *maxPolyphony* member.

Notes

The number of simultaneously addressable voices (polyphony) is conceptually different from the number of allocated voices of each type on the synthesizer; the sum of which may be greater than the number returned by this function. In fact, to accommodate peak usage of single voice types, it may be desirable to set a total number of allocated synthesizer voices greater than the number of voices that can be simultaneously addressed by Scream. For example, if you allocate 24 VAG, 24 PCM, and 16 ATRAC9™ voices (a total of 64), the synthesizer could not play more than the specified number of any one type simultaneously. Conversely, if you allocate 64 voices of each type (a total of 192), the synthesizer can potentially play all 64 voices of a single type simultaneously (assuming default maximum polyphony). For further details, see "Allocating Synthesizer Voice Types" in the "Configuration, Initialization, and Shutdown" chapter of the *Scream Library Overview*.

See Also

SCE_SCREAM_SND_DEFAULT_MAX_POLYPHONY, sceScreamGetSynthName(), sceScreamSetGroupVoiceRange(), sceScreamGetSoundVoiceCount(), SceScreamPlatformInitEx2, sceScreamGetVoiceTypeName()

sceScreamGetNumGlobalVariables

Retrieves the current number of global variables.

Definition

```
uint32_t sceScreamGetNumGlobalVariables(void);
```

Return Values

Returns the current number of global variables, not to be confused with global registers.

Description

This function retrieves the current number of global variables. The maximum number of global variables is defined at initialization time using the SceScreamPlatformInitEx2.*maxGlobalVariables* member, which defaults to SCE_SCREAM_SND_DEFAULT_MAX_GLOBAL_VARIABLES.

See Also

sceScreamAddGlobalVariable(), sceScreamAddSetGlobalVariable(), sceScreamDeleteGlobalVariable()

sceScreamGetPlaybackMode

Retrieves the current playback mode.

Definition

```
uint32_t sceScreamGetPlaybackMode(void);
```

Return Values

Returns one of the Playback Mode Constants.

Description

Retrieves the current playback (output) mode setting. If the playback mode was set to `SCE_SCREAM_SPEAKER_MODE_BEST` at initialization time (in the *playbackMode* member of the `SceScreamPlatformInitEx2` structure used when calling `sceScreamStartSoundSystemEx2()`), the playback mode that was selected as best is returned.

See Also

`sceScreamSetPlaybackMode()`

sceScreamGetRandomIndex

Retrieves the current seed index of the Scream random number generator.

Definition

```
int16_t sceScreamGetRandomIndex(void);
```

Return Values

Returns a seed index value within the range: -32,768 to +32,767.

Description

This function returns the current seed index of the Scream random number generator. The seed index can be used in relation to variable speed replays, where some audio parameter values are determined by random numbers, and there is a need for identical repetition of the original playback. To use the seed index for this purpose, before initiating a sequence you may later wish to replay, call `sceScreamGetRandomIndex()` and store its return value. Then, before initiating the replay, call `sceScreamSetRandomIndex()` with the previously stored value.

See Also

`sceScreamSetRandomIndex()`, `sceScreamGetScriptSpeedFactor()`, `sceScreamSetScriptSpeedFactor()`

sceScreamGetScriptSpeedFactor

Retrieves the global script speed factor for variable speed replays.

Definition

```
float sceScreamGetScriptSpeedFactor(
    uint32_t *outFlags
);
```

Arguments

outFlags	(Output) Pointer to a `uint32_t` in which to receive any current script speed flag values. By default, variable speed replay affects time domain scripting properties only and does not affect pitch domain or ADSR envelope durations. The `SCE_SCREAM_SND_SCRIPTSPEED_AFFECT_PITCH` flag set means variable speed replay also affects the pitch domain. The `SCE_SCREAM_SND_SCRIPTSPEED_AFFECT_ADSR` flag set indicates variable speed replay scales the durations of ADSR segments. Set to `NULL` if flag values are not required.

Return Values

Returns the current script speed factor and optionally, script speed flags.

Description

This function retrieves the global script speed factor. The script speed factor is used for variable speed replays. A value of 1.0 indicates normal speed. Values greater than 1.0 indicate increasingly faster replays. Values less than 1.0 indicate progressively slower replays.

See Also

`sceScreamSetScriptSpeedFactor()`, `sceScreamGetGroupScriptSpeedFactor()`, `sceScreamSetGroupScriptSpeedFactor()`, `sceScreamGetRandomIndex()`, `sceScreamSetRandomIndex()`

sceScreamGetSFXGlobalReg

Retrieves the value of a Scream global register.

Definition

```
int8_t sceScreamGetSFXGlobalReg(
    int32_t which
);
```

Arguments

which	(Input) One-based index of the global register for which to retrieve the value. Must be between 1 and SCE_SCREAM_SND_MAX_GLOBAL_REGISTERS.

Return Values

Returns the value of the specified global register.

Description

This function retrieves the current value of the specified global register (not to be confused with the global variables used with CCSounds). Scream maintains a set of global registers (the total number of which is SCE_SCREAM_SND_MAX_GLOBAL_REGISTERS). Global registers can be used to communicate game engine state to Scream Sound scripts, either for direct application to parameter values, or for more complex logical operations.

See Also

sceScreamSetSFXGlobalReg()

sceScreamGetStreamingFileDirectory

Retrieves the associated directory path string for a specified stream path group.

Definition

```
const char *sceScreamGetStreamingFileDirectory(
    uint32_t uiDirGroup
);
```

Arguments

uiDirGroup	(Input) Zero-based index of a stream path group to retrieve the directory path for. Stream path group indexes are between 0 and (SCE_SCREAM_SND_MAX_STREAMING_FILE_DIRECTORIES – 1).

Return Values

If successful, returns the directory path string for the specified stream path group. Otherwise, returns NULL.

Description

In Scream, every stream file (that is, a file imported into Scream Tool using the *Stream* Grain) is associated with a stream path group. You reference stream path groups using a zero-based index. Directory paths can be set for each stream path group using the sceScreamSetStreamingFileDirectory() function. This function retrieves the current directory path setting for the specified stream path group.

Notes

Stream path groups are set in the Scream Tool stream inspector panel. For details, see “Setting Stream Properties” in the “Streams” chapter of *Scream Tool Help*.

See Also

sceScreamSetStreamingFileDirectory()

sceScreamGetSynthName

Retrieves the name of the rendering synthesizer with which Scream is linked.

Definition

```
const char *sceScreamGetSynthName(void);
```

Return Values

Returns the name of the rendering synthesizer.

Description

This function retrieves the name of the rendering synthesizer with which Scream is linked. Scream processes control information, but all audio processing takes place on a rendering synthesizer.

See Also

sceScreamGetMaxPolyphony(), sceScreamGetVoiceTypeName()

sceScreamGetSystemRunning

Confirms whether Scream has been initialized and is running.

Definition

```
bool sceScreamGetSystemRunning(void);
```

Return Values

Returns `TRUE` if Scream is running. Returns `FALSE` if Scream is not running.

Description

This is a status function that confirms whether Scream has been initialized and is running.

See Also

`sceScreamStartSoundSystemEx2()`, `sceScreamStopSoundSystem()`

sceScreamGetTick

Retrieves the current Scream tick count.

Definition

```
uint32_t sceScreamGetTick(void);
```

Return Values

Returns the current tick count.

Description

Scream ticks occur at a rate of 240 times per second. Each tick increments an internal counter. This function returns the current value of the internal counter.

sceScreamGetVoiceTypeName

Retrieves the name of a synthesizer voice type based on a specified data type.

Definition

```
const char *sceScreamGetVoiceTypeName(
    uint32_t dataType
);
```

Arguments

dataType	(Input) One of the NGS Voice Data Type Constants.

Return Values

Returns a string representing the name of a synth voice type corresponding with the specified data type.

Description

This function retrieves the name of synthesizer voice type corresponding with a specified voice data type.

See Also

sceScreamGetAllocatedVoiceCountByType(), sceScreamGetSynthName(), sceScreamGetMaxPolyphony()

sceScreamSetDebugHandler

Sets a custom debug text output function.

Definition

```
int32_t sceScreamSetDebugHandler(
    SceScreamSndDebugHandler proc
);
```

Arguments

proc	(Input) Pointer to an application-defined function of type SceScreamSndDebugHandler().

Return Values

Returns SCE_SCREAM_SS_ERROR_OK if the operation was successful, otherwise returns SCE_SCREAM_SS_ERROR_INVALID_PARAMETER if an invalid parameter value was detected.

Description

You can create a custom debug text output function that adheres to the SceScreamSndDebugHandler() prototype. This function allows you to set your custom function to receive all Scream and associated synthesizer debug text output – diverting output that would otherwise go to the default `printf()`-like function.

You can unset the current debug handler by specifying `NULL` as the value of the *proc* argument.

It is recommended to set your custom debug text output function *prior* to initializing Scream, as much information is produced during the initialization process.

See Also

SceScreamSndDebugHandler(), sceScreamOutputHandlerInfoToTTY(), sceScreamOutputAllPlayingSoundInfoToTTY()

sceScreamSetGlobalVariableByHash

Sets the value of a global variable.

Definition

```
bool sceScreamSetGlobalVariableByHash(
    uint32_t nameHash,
    float val
);
```

Arguments

nameHash	(Input) Hash of a global variable to set.
val	(Input) A floating-point value to set the specified global variable to.

Return Values

Returns `TRUE` if the variable specified by *nameHash* exists. Returns `FALSE` otherwise.

Description

This function sets the value of a global variable, specified by hash.

See Also

sceScreamGetGlobalVariableByIndex(), sceScreamAddGlobalVariable(), sceScreamAddSetGlobalVariable(), sceScreamSetGlobalVariableByName(), sceScreamGetHashFromName()

sceScreamSetGlobalVariableByIndex

Sets the value of a global variable.

Definition

```
bool sceScreamSetGlobalVariableByIndex(
    uint32_t varIndex,
    float val
);
```

Arguments

varIndex	(Input) Index of a global variable to set. A value stored in the *outVarIndex* parameter resulting from calls to any of the following functions: sceScreamAddGlobalVariable(), sceScreamAddSetGlobalVariable().
val	(Input) A floating-point value to set the specified global variable to.

Return Values

Returns TRUE if the specified *index* is valid. Returns FALSE otherwise.

Description

This function sets the value of a global variable, specified by index.

See Also

sceScreamGetGlobalVariableByIndex(), sceScreamAddGlobalVariable(), sceScreamAddSetGlobalVariable()

sceScreamSetGlobalVariableByName

Sets the value of a global variable.

Definition

```
bool sceScreamSetGlobalVariableByName(
    const char *name,
    float val
);
```

Arguments

name	(Input) Name of the global variable to set.
val	(Input) A floating-point value to set the specified global variable to.

Return Values

Returns `TRUE` if the variable specified by *name* exists. Returns `FALSE` otherwise.

Description

This function sets the value of a global variable, specified by name.

See Also

sceScreamGetGlobalVariableByIndex(), sceScreamAddGlobalVariable(), sceScreamAddSetGlobalVariable(), sceScreamSetGlobalVariableByHash()

sceScreamSetMinRipoffTime

Changes the system-wide minimum voice ripoff time.

Definition

```
int32_t sceScreamSetMinRipoffTime(
    float minRipoffTimeSeconds
);
```

Arguments

minRipoffTimeSeconds	(Input) The number of seconds a voice must be active before it is eligible for stealing.

Return Values

Returns SCE_SCREAM_SS_ERROR_OK if the operation was successful, otherwise returns SCE_SCREAM_SS_ERROR_INVALID_PARAMETER if an invalid parameter value was detected.

Description

This function changes the system-wide minimum voice ripoff time. The minimum ripoff time is the time a voice must be active before it is eligible for stealing.

You can also set the system wide minimum ripoff time at initialization time using SceScreamPlatformInitEx2.*minRipoffTime*.

See Also

SceScreamPlatformInitEx2

sceScreamSetPlaybackMode

Sets the playback mode.

Definition

```
uint32_t sceScreamSetPlaybackMode(
    uint32_t mode,
    uint32_t customSpeakerAzimuths[SCE_SCREAM_SND_POLPAN_MAX_SPEAKERS]
);
```

Arguments

mode	(Input) One of the Playback Mode Constants, other than SCE_SCREAM_SPEAKER_MODE_BEST, which is only valid at initialization time as a value for the SceScreamPlatformInitEx2 *playbackMode* member.
customSpeakerAzimuths	(Input) **Note:** This parameter is ignored when running Scream on the NGS synthesizer.

Return Values

Returns one of the Playback Mode Constants. If the call was successful, this value represents the new (specified) playback mode. If not successful, the value is that of the current (unchanged) playback mode.

Description

This function sets the playback (output) mode. Available Playback Modes are defined by the Playback Mode Constants.

See Also

sceScreamGetPlaybackMode()

sceScreamSetRandomIndex

Sets the seed index of the Scream random number generator.

Definition

```
int32_t sceScreamSetRandomIndex(
    int16_t iIndex
);
```

Arguments

iIndex	(Input) An `int16_t` within the range: -32,768 to +32,767.

Return Values

Returns 0 upon setting the seed index of the random number generator.

Description

This function sets the seed index of the Scream random number generator. The seed index can be used in relation to variable speed replays, where some audio parameter values are determined by random numbers, and there is a need for identical repetition of the original playback, as in an action replay, for instance. To use the seed index for this purpose, before initiating a sequence you may later wish to replay, call `sceScreamGetRandomIndex()` and store its return value. Then, before initiating the replay, call `sceScreamSetRandomIndex()` with the previously stored value.

See Also

`sceScreamGetRandomIndex()`, `sceScreamGetScriptSpeedFactor()`, `sceScreamSetScriptSpeedFactor()`

sceScreamSetScriptSpeedFactor

Sets the global script speed factor for variable speed replays.

Definition

```
float sceScreamSetScriptSpeedFactor(
    float speedFactor,
    uint32_t flags
);
```

Arguments

speedFactor	(Input) A speed multiplier for variable speed replays. Must be greater than 0. A value of 1.0 indicates normal speed.
flags	(Input) By default, variable speed replay affects time domain scripting properties only and does not affect pitch domain or ADSR envelope durations. Set the SCE_SCREAM_SND_SCRIPTSPEED_AFFECT_PITCH flag if you want variable speed replay to also affect the pitch domain. Set the SCE_SCREAM_SND_SCRIPTSPEED_AFFECT_ADSR flag if you want variable speed replay to scale the durations of ADSR segments.

Return Values

Returns the script speed factor that was set, or the current script speed factor if the new value could not be set.

Description

In Scream you can set both global and volume group-specific script speed factors for variable-speed replays. This function sets the global script speed factor. Use the sceScreamSetGroupScriptSpeedFactor() function to set a group-specific script speed factor.

speedFactor functions similarly to the jog wheel on a video cassette player. It can either be at rest (where playback is at normal speed), rotated clockwise (to speed up playback), or rotated counter-clockwise (to slow down playback). A *speedFactor* value of 1.0 indicates normal speed. Values greater than 1.0 increasingly speed up playback. Values less than 1.0 progressively slow down playback.

speedFactor must be greater than 0.0 (variable speed replay cannot stand still or play backwards). While there are no other direct programmatic constraints on the *speedFactor* value, there are some indirect and practical constraints.

At the upper extremes, pitch shift on the rendering synthesizer cannot exceed two octaves above the original pitch, so the effective maximum *speedFactor* value for the pitch domain is 4.0. The time-based playback parameters can be speeded up by factors in excess of 4×. Note however, that very high speeds with many simultaneous Sounds rapidly consume processor cycles. A practical constraint at the high end of the time domain may be to keep *speedFactor* value less than 20.0. Note also that envelope values (ADSR) are not affected by variable speed replays, so long attacks, decays, and releases remain the same no matter what the *speedFactor* is set to.

At the lower extremes, because *speedFactor* temporarily changes the Scream tick rate, very low *speedFactor* values could cause a noticeable delay following the replay until the tick rate returns to 240 ticks-per-second. A practical constraint at the low end may be to keep the *speedFactor* value greater than 0.05. If you need a slower replay speed, use pause/continue functionality to affect frame-by-frame replays.

For more information, see "Manipulating Script Speed" in the "Working with System Globals" chapter of the *Scream Library Overview*.

See Also

sceScreamGetScriptSpeedFactor(), sceScreamGetGroupScriptSpeedFactor(), sceScreamSetGroupScriptSpeedFactor(), sceScreamGetRandomIndex(), sceScreamSetRandomIndex()

sceScreamSetSFXGlobalReg

Sets the value of a Scream global register.

Definition

```
int32_t sceScreamSetSFXGlobalReg(
    int32_t which,
    int8_t val
);
```

Arguments

which	(Input) One-based index of the global register to set. Must be between 1 and SCE_SCREAM_SND_MAX_GLOBAL_REGISTERS.
val	(Input) Value to set the global register to. Must be between -128 and 127.

Return Values

Returns SCE_SCREAM_SS_ERROR_OK if the operation was successful, otherwise returns SCE_SCREAM_SS_ERROR_INVALID_PARAMETER if an invalid parameter value was detected.

Description

This function sets the value of the specified global register. Scream maintains a set of global registers (the total number of which is SCE_SCREAM_SND_MAX_GLOBAL_REGISTERS). Global registers (not to be confused with the global variables used with CCSounds) can be used to communicate game engine state to Scream Sound scripts, either for direct application to parameter values, or for more complex logical operations.

See Also

sceScreamGetSFXGlobalReg()

Document serial number: 000004892117

sceScreamSetStreamingFileDirectory

Sets the directory path string for a specified stream path group.

Definition

```
int32_t sceScreamSetStreamingFileDirectory(
    uint32_t uiDirGroup,
    const char *pDirectoryString
);
```

Arguments

uiDirGroup	(Input) Zero-based index of the stream path group to set. Stream path group indexes are between 0 and (SCE_SCREAM_SND_MAX_STREAMING_FILE_DIRECTORIES – 1).
pDirectoryString	(Input) Pointer to a directory path to use for the stream path group.

Return Values

Returns 1 if the directory path was set. Returns -1 if the directory path was not set. Returns 0 if any arguments are invalid.

Description

In Scream, every stream file (that is, a file imported into Scream Tool using the *Stream* Grain) is associated with a stream path group. You reference stream path groups using a zero-based index. This function allows you to set the directory path for a specified stream path group.

Notes

Scream does not copy the directory path string, but instead just saves the passed-in pointer. Therefore, the string pointed to by the pointer address must be in static data memory, such that it will exist for the lifetime of the application.

See Also

sceScreamGetStreamingFileDirectory()

sceScreamStartSoundSystemEx2

Initializes Scream for use by an application.

Definition

```
int32_t sceScreamStartSoundSystemEx2(
    const SceScreamPlatformInitEx2 *platformInit
);
```

Arguments

platformInit	(Input) Pointer to an initialized SceScreamPlatformInitEx2 data structure.

Return Values

Returns 0 if initialization was successful. Returns SCE_SCREAM_SS_ERROR_SYSTEM_ALREADY_STARTED if Scream is already initialized. Returns SCE_SCREAM_SS_ERROR_SYNTH_INIT_FAILED if the underlying synthesizer failed to initialize.

Description

Use this function to initialize Scream for use by an application.

See Also

SceScreamPlatformInitEx2, sceScreamFillDefaultScreamPlatformInitArgsEx2(), sceScreamStopSoundSystem(), sceScreamGetSystemRunning()

sceScreamStopAllSounds

Stops all Sounds in the active Scream Sound handlers list.

Definition

```
int32_t sceScreamStopAllSounds(void);
```

Return Values

Returns SCE_SCREAM_SS_ERROR_OK.

Description

This function stops all Sounds being generated by the Scream runtime engine. This function performs a *hard* stop. That is, audio generation stops almost instantaneously, depending on how far ahead the rendering synthesizer has buffered samples for playback.

For more information, see "Stopping All Sounds" in the "Working with System Globals" chapter of the *Scream Library Overview*.

Notes

The scope of this function includes Scream Sounds in both internal and external handlers. The sceScreamStopAllSounds() function does not stop Streams. Use the Sndstream function sceScreamStopAllStreams() to stop all Streams.

See Also

sceScreamStopSound(), sceScreamStopAllSoundsInBank(), sceScreamStopAllSoundsInGroup()

sceScreamStopAllSoundsByIndex

Stops all instances of a Sound as specified by Bank and index references.

Definition

```
int32_t sceScreamStopAllSoundsByIndex(
    SceScreamSFXBlock2 *bank,
    int16_t index,
    int32_t behavior
);
```

Arguments

bank	(Input) SceScreamSFXBlock2 pointer referencing the Bank from which the Sound(s) were instantiated.
index	(Input) Index of the Sound within the specified *bank* from which the Sound(s) were instantiated.
behavior	(Input) A choice of two stop behaviors: SCE_SCREAM_SND_STOP_BEHAVIOR_KEYOFF or SCE_SCREAM_SND_STOP_BEHAVIOR_SILENCE.

Return Values

Returns SCE_SCREAM_SS_ERROR_OK if the operation was successful, otherwise returns SCE_SCREAM_SS_ERROR_INVALID_PARAMETER if an invalid parameter value was detected.

Description

This function stops all instances of a Sound as specified by Bank and index references, without the need to specify Sound handles.

The *behavior* parameter provides a choice of two stop behaviors:

- SCE_SCREAM_SND_STOP_BEHAVIOR_KEYOFF: Performs a graceful stop, triggering any *On Stop Marker* grain events, and issuing key-off messages to active voices with ADSR Release settings.
- SCE_SCREAM_SND_STOP_BEHAVIOR_SILENCE: Performs an instantaneous stop.

This function is primarily used for looping Sounds; one-shot Sounds stop themselves.

See Also

sceScreamStopAllSounds(), sceScreamStopAllSoundsInGroup(), sceScreamStopAllSoundsInBank(), sceScreamStopSound(), sceScreamPlaySoundByIndexEx(), sceScreamPlaySoundByNameEx()

sceScreamStopSoundSystem

Shuts down the Scream runtime engine.

Definition

```
int32_t sceScreamStopSoundSystem(void);
```

Return Values

Returns 0 if shutdown was successful. Returns SCE_SCREAM_SS_ERROR_SYSTEM_NOT_STARTED if Scream is not currently running.

Description

Use this function to completely shut down Scream. This function silences all Scream voices, and releases all Scream-allocated memory.

See Also

sceScreamStartSoundSystemEx2(), sceScreamGetSystemRunning()

Group Functions

Summary

Group functions manipulate groups and the Sounds to which they are assigned. They also set and retrieve group state information.

Function	Description
sceScreamContinueAllSoundsInGroup	Continues paused Sounds belonging to one or more Groups.
sceScreamContinueGroup	Continues one or more paused Groups.
sceScreamGetActiveSoundCountByGroup	Retrieves the current number of active sounds in a Group.
sceScreamGetActiveVoiceCountByGroup	Retrieves the current number of active synthesizer voices in use by a Group.
sceScreamGetGroupsByOutputDest	Retrieves the set of Groups currently routing to a specified output destination.
sceScreamGetGroupScriptSpeedFactor	Retrieves a group-specific script speed factor for variable speed replays.
sceScreamGetMasterVolume	Retrieves the current volume level of a Group, or the level of Scream's global volume.
sceScreamPauseAllSoundsInGroup	Pauses all active Sounds belonging to one or more Groups.
sceScreamPauseGroup	Pauses one or more Groups.
sceScreamSetGroupDistanceModel	Sets a distance model for a Group.
sceScreamSetGroupMute	Mutes one or more Groups.
sceScreamSetGroupScriptSpeedFactor	Sets a group-specific script speed factor for variable speed replays.
sceScreamSetGroupSolo	Solos one or more Groups.
sceScreamSetGroupVoiceOutputDest	Sets voice output destination for a Group.
sceScreamSetGroupVoiceRange	Sets the voice allocation range for a Group.
sceScreamSetMasterVolume	Sets the volume level of a Group, or the level of Scream's global volume.
sceScreamSetMasterVolumeDucker	Activates (or deactivates) a volume ducker.
sceScreamStopAllSoundsInGroup	Stops all Sounds in one or more Groups.

sceScreamContinueAllSoundsInGroup

Continues paused Sounds belonging to one or more Groups.

Definition

```
int32_t sceScreamContinueAllSoundsInGroup(
    uint32_t groups
);
```

Arguments

groups	(Input) A bit field indicating the Group(s) in which to continue paused Sounds. One or more of the Group Flags. Use the bitwise OR operator for multiple selections.

Return Values

Returns SCE_SCREAM_SS_ERROR_OK if the operation was successful, otherwise returns SCE_SCREAM_SS_ERROR_INVALID_PARAMETER if an invalid parameter value was detected.

Description

This function continues paused Sounds – belonging to one or more Groups – that have been paused using the sceScreamPauseAllSoundsInGroup() function. You specify the target Group(s) using the Group Flags. Use the bitwise OR operator to make multiple selections to create a bit field of Groups in which to continue paused Sounds.

See Also

sceScreamPauseAllSoundsInGroup(), sceScreamPauseGroup(), sceScreamContinueGroup(), sceScreamPauseSound(), sceScreamContinueSound(), sceScreamReverbPause(), sceScreamReverbContinue()

sceScreamContinueGroup

Continues one or more paused Groups.

Definition

```
int32_t sceScreamContinueGroup(
    uint32_t groups
);
```

Arguments

groups	(Input) A bit field value indicating the Group(s) to continue. One or more of the Group Flags. Use the bitwise OR operator for multiple selections.

Return Values

Returns SCE_SCREAM_SS_ERROR_OK if the operation was successful, otherwise returns SCE_SCREAM_SS_ERROR_INVALID_PARAMETER if an invalid parameter value was detected.

Description

This function continues one or more Groups that have been paused using the sceScreamPauseGroup() function. You specify the target Group(s) to continue using the Group Flags. Use the bitwise OR operator to make multiple selections to create a bit field of Groups to continue.

Notes

This function continues a Group itself, as distinct from sceScreamContinueAllSoundsInGroup(), which iterates over all paused Sounds belonging to a target Group, and continues their instances.
Note: This function does not continue a paused Sound if that Sound's instance is still paused.

See Also

sceScreamPauseGroup(), sceScreamPauseAllSoundsInGroup(), sceScreamContinueAllSoundsInGroup(), sceScreamPauseSound(), sceScreamContinueSound(), sceScreamReverbPause(), sceScreamReverbContinue()

sceScreamGetActiveSoundCountByGroup

Retrieves the current number of active sounds in a Group.

Definition

```
uint32_t sceScreamGetActiveSoundCountByGroup(
    int32_t which
);
```

Arguments

which	(Input) One of the Volume Groups constants.

Return Values

The function returns the number of active sounds in the specified Group.

Description

This function returns the current number of active sounds in a specified Group.

See Also

sceScreamGetActiveVoiceCountByGroup()

sceScreamGetActiveVoiceCountByGroup

Retrieves the current number of active synthesizer voices in use by a Group.

Definition

```
uint32_t sceScreamGetActiveVoiceCountByGroup(
    int32_t which
);
```

Arguments

which	(Input) One of the Volume Groups constants.

Return Values

The function returns the number of active synthesizer voices in use by the specified Group.

Description

This function returns the current number of active synthesizer voices in use by a specified Group.

See Also

sceScreamGetActiveSoundCountByGroup(), sceScreamGetMaxPolyphony()

Document serial number: 000004892117

sceScreamGetGroupsByOutputDest

Retrieves the set of Groups currently routing to a specified output destination.

Definition

```
int32_t sceScreamGetGroupsByOutputDest(
    int32_t outputDest,
    uint32_t *outGroups
);
```

Arguments

outputDest	(Input) An output destination against which to query for routing Groups. One of the Output Destinations constants.
outGroups	(Output) A pointer to a `uint32_t` output variable containing a bitwise combination of zero or more Group Flags indicating the Groups currently routing to the specified output destination.

Return Values

If successful, returns `SCE_SCREAM_SS_ERROR_OK`. Otherwise, returns `SCE_SCREAM_SS_ERROR_INVALID_PARAMETER`.

Description

This function retrieves the set of Groups currently routing to a specified output destination, and stores corresponding Group Flag(s) in an output variable. You can use the retrieved set of Groups as input to Group-based transport and other functions.

See Also

`sceScreamSetGroupVoiceOutputDest()`, Output Destinations, Group Flags, `sceScreamGetActiveVoiceCountByGroup()`, `sceScreamGetActiveSoundCountByGroup()`, `sceScreamPauseGroup()`, `sceScreamContinueGroup()`, `sceScreamPauseAllSoundsInGroup()`, `sceScreamContinueAllSoundsInGroup()`, `sceScreamSetGroupSolo()`, `sceScreamSetGroupMute()`, `sceScreamStopAllSoundsInGroup()`

sceScreamGetGroupScriptSpeedFactor

Retrieves a group-specific script speed factor for variable speed replays.

Definition

```
float sceScreamGetGroupScriptSpeedFactor(
    uint32_t group,
    uint32_t *outFlags
);
```

Arguments

group	(Input) Index of the target Group from which to retrieve a script speed factor. Any of the Volume Groups constants except SCE_SCREAM_GROUP_MASTER_VOLUME, which is not valid as a script speed target.
outFlags	(Output) Pointer to a uint32_t variable in which to receive any script speed flag values applied to the target group. By default, variable speed replay affects time domain scripting properties only and does not affect pitch domain or ADSR envelope durations. The SCE_SCREAM_SND_SCRIPTSPEED_AFFECT_PITCH flag set means variable speed replay also affects the pitch domain. The SCE_SCREAM_SND_SCRIPTSPEED_AFFECT_ADSR flag set indicates variable speed replay scales the durations of ADSR segments. Set to NULL if flag values are not required.

Return Values

Returns the specified group's script speed factor and optionally, script speed flags.

Description

This function retrieves a group-specific script speed factor. Script speed factors are used for variable speed replays. A value of 1.0 indicates normal speed. Values greater than 1.0 indicate increasingly faster replays. Values less than 1.0 indicate progressively slower replays.

See Also

sceScreamSetGroupScriptSpeedFactor(), sceScreamGetScriptSpeedFactor(), sceScreamSetScriptSpeedFactor(), sceScreamGetRandomIndex(), sceScreamSetRandomIndex()

sceScreamGetMasterVolume

Retrieves the current volume level of a Group, or the level of Scream's global volume.

Definition

```
float sceScreamGetMasterVolume(
    int32_t which
);
```

Arguments

which (Input) One of the Volume Groups constants.

Return Values

The function returns the specified volume level as a floating-point value in the range SCE_SCREAM_SND_MIN_GAIN to SCE_SCREAM_SND_MAX_GAIN.

Description

This function returns the current volume of a Group, or the level of Scream's global volume. The *which* parameter is the index of the Group to retrieve a volume level for, and can be specified using one of the Volume Groups constants.

See Also

sceScreamSetMasterVolume()

sceScreamPauseAllSoundsInGroup

Pauses all active Sounds belonging to one or more Groups.

Definition

```
int32_t sceScreamPauseAllSoundsInGroup(
    uint32_t groups
);
```

Arguments

groups	(Input) A bit field value indicating the Group(s) in which to pause constituent Sounds. One or more of the Group Flags. Use the bitwise `OR` operator for multiple selections.

Return Values

Returns `SCE_SCREAM_SS_ERROR_OK` if the operation was successful, otherwise returns `SCE_SCREAM_SS_ERROR_INVALID_PARAMETER` if an invalid parameter value was detected.

Description

This function pauses all playing Sounds belonging to one or more Groups. Any Child Sounds belonging to the Group(s) are also paused. You specify the target Group(s) using the Group Flags. Use the bitwise `OR` operator to make multiple selections to create a bit field of Groups in which to pause constituent Sounds. Use the `sceScreamContinueAllSoundsInGroup()` function to continue Group-constituent Sounds paused by this function.

For more details, see "Pausing All Sounds in a Group" in the "Working with Groups" chapter of the *Scream Library Overview*.

Notes

The pause mechanism used in this function is not persistent. That is, a Sound started after the Group it belongs to has been paused (using this function) will still play. For a persistent Group pause mechanism, use `sceScreamPauseGroup()`.

See Also

`sceScreamContinueAllSoundsInGroup()`, `sceScreamPauseGroup()`, `sceScreamContinueGroup()`, `sceScreamPauseSound()`, `sceScreamContinueSound()`, `sceScreamReverbPause()`, `sceScreamReverbContinue()`

sceScreamPauseGroup

Pauses one or more Groups.

Definition

```
int32_t sceScreamPauseGroup(
    uint32_t groups
);
```

Arguments

groups	(Input) A bit field value indicating the Group(s) to pause. One or more of the Group Flags. Use the bitwise OR operator for multiple selections.

Return Values

Returns SCE_SCREAM_SS_ERROR_OK if the operation was successful, otherwise returns SCE_SCREAM_SS_ERROR_INVALID_PARAMETER if an invalid parameter value was detected.

Description

This function pauses one or more target Groups, causing all active and inactive Sounds and Child Sounds contained in the Groups to pause. You specify the target Group(s) using the Group Flags. Use the bitwise OR operator to make multiple selections to create a bit field of Groups to pause. Use the sceScreamContinueGroup() function to continue paused Groups.

For more details, see "Pausing a Group" in the "Working with Groups" chapter of the *Scream Library Overview*.

Notes

This function pauses a target Group itself, as opposed to sceScreamPauseAllSoundsInGroup(), which iterates over all playing Sounds belonging to a target Group, and pauses their instances. This Group pause can therefore be considered persistent. Sounds that are started after the Group they belong to is paused, start in a paused state.

See Also

sceScreamContinueGroup(), sceScreamPauseAllSoundsInGroup(), sceScreamContinueAllSoundsInGroup(), sceScreamPauseSound(), sceScreamContinueSound(), sceScreamReverbPause(), sceScreamReverbContinue()

sceScreamSetGroupDistanceModel

Sets a distance model for a Group.

Definition

```
int32_t sceScreamSetGroupDistanceModel(
    int32_t group,
    uint32_t modelNameHash
);
```

Arguments

group	(Input) Index of the target Group for which to set a distance model. One of the Volume Group constants.
modelNameHash	(Input) A hash of the name of a distance model. Your audio designer can supply you with a list of the names of distance models contained in a distance model file. You can use the sceScreamGetHashFromName() function to obtain a hash from a distance model name. You can assign no distance model to a Group by setting this parameter to 0, which is equivalent to setting "2D (none)" in Bank contents. Sounds inheriting this setting do not attenuate based on distance.

Return Values

If successful, returns SCE_SCREAM_SS_ERROR_OK. Otherwise, returns SCE_SCREAM_SS_ERROR_INVALID_PARAMETER.

Description

This function sets a distance model on a Group basis. Group-assigned Sounds, upon which the Distance Model property in Bank contents is set to "By Group", inherit their Group's distance model setting. To set a Group's distance model, you specify a target Group, and a hash representing the name of a distance model to set.

Notes

Designers can also set Group distance models in Bank contents, and export these settings as part of a group mixer file. When setting Group distance models using this function, check with your audio designer to make sure you are not unintentionally overwriting their settings.

See Also

sceScreamGetHashFromName(), SceScreamPlatformInitEx2.*pDistanceModelFile*

sceScreamSetGroupMute

Mutes one or more Groups.

Definition

```
int32_t sceScreamSetGroupMute(
    uint32_t groups,
    bool mute
);
```

Arguments

groups	(Input) A bit field indicating the Group(s) to mute. One or more of the Group Flags; see Group Flags. Use the bitwise OR operator for multiple selections.
mute	(Input) A Boolean value expressing mute status. Set to TRUE to mute the specified group(s); set to FALSE to cancel mute of the specified group(s).

Return Values

Returns SCE_SCREAM_SS_ERROR_OK if the operation was successful, otherwise returns SCE_SCREAM_SS_ERROR_INVALID_PARAMETER if an invalid parameter value was detected.

Description

This function mutes audio output from one or more Groups. You specify the target Group(s) using the Group Flags; see Group Flags. Use the bitwise OR operator to make multiple selections to create a bit field of Groups to mute.

See Also

sceScreamSetGroupSolo()

sceScreamSetGroupScriptSpeedFactor

Sets a group-specific script speed factor for variable speed replays.

Definition

```
float sceScreamSetGroupScriptSpeedFactor(
    uint32_t group,
    float speedFactor,
    uint32_t flags
);
```

Arguments

group	(Input) Index of the target Group for which to set a script speed factor. Any of the Volume Groups constants except SCE_SCREAM_GROUP_MASTER_VOLUME, which is not valid as a script speed target.
speedFactor	(Input) A speed multiplier for variable speed replays. Must be greater than 0. A value of 1.0 indicates normal speed.
flags	(Input) By default, variable speed replay affects time domain scripting properties only and does not affect pitch domain or ADSR envelope durations. Set the SCE_SCREAM_SND_SCRIPTSPEED_AFFECT_PITCH flag if you want variable speed replay to also affect the pitch domain. Set the SCE_SCREAM_SND_SCRIPTSPEED_AFFECT_ADSR flag if you want variable speed replay to scale the durations of ADSR segments.

Return Values

If the script speed factor was set correctly, returns the new factor. Otherwise returns the existing script speed factor if a new factor could not be set.

Description

In Scream, you can set both global and volume group-specific script speed factors for variable-speed replays. This function sets a group-specific script speed factor. Use the sceScreamSetScriptSpeedFactor() function to set the global script speed factor.

speedFactor functions similarly to the jog wheel on a video cassette player. It can either be at rest (where playback is at normal speed), rotated clockwise (to speed up playback), or rotated counter-clockwise (to slow down playback). A *speedFactor* value of 1.0 indicates normal speed. Values greater than 1.0 increasingly speed up playback. Values less than 1.0 progressively slow down playback.

speedFactor must be greater than 0.0 (variable speed replay cannot stand still or play backwards). While there are no other direct programmatic constraints on the *speedFactor* value, there are some indirect and practical constraints.

At the upper extremes, pitch shift on the rendering synthesizer cannot exceed two octaves above the original pitch, so the effective maximum *speedFactor* value for the pitch domain is 4.0. The time-based playback parameters can be speeded up by factors in excess of 4×. Note however, that very high speeds with many simultaneous Sounds rapidly consume processor cycles. A practical constraint at the high end of the time domain may be to keep *speedFactor* value less than 20.0. Note also that envelope values (ADSR) are not affected by variable speed replays, so long attacks, decays, and releases remain the same no matter what the *speedFactor* is set to.

At the lower extremes, because *speedFactor* temporarily changes the Scream tick rate, very low *speedFactor* values could cause a noticeable delay following the replay until the tick rate returns to 240 ticks-per-second. A practical constraint at the low end may be to keep the *speedFactor* value

greater than 0.05. If you need a slower replay speed, use pause/continue functionality to affect frame-by-frame replays.

For more information, see "Manipulating Group Script Speed" in the "Working with Groups" chapter of the *Scream Library Overview*.

See Also

`sceScreamGetGroupScriptSpeedFactor()`, `sceScreamGetScriptSpeedFactor()`, `sceScreamSetScriptSpeedFactor()`, `sceScreamGetRandomIndex()`, `sceScreamSetRandomIndex()`

sceScreamSetGroupSolo

Solos one or more Groups.

Definition

```
int32_t sceScreamSetGroupSolo(
    uint32_t groups,
    bool solo
);
```

Arguments

groups	(Input) A bit field indicating the Group(s) to solo. One or more of the Group Flags. Use the bitwise OR operator for multiple selections.
solo	(Input) A Boolean value expressing solo status. Set to TRUE to solo the specified group(s); set to FALSE to cancel solo of the specified group(s).

Return Values

Returns SCE_SCREAM_SS_ERROR_OK if the operation was successful, otherwise returns SCE_SCREAM_SS_ERROR_INVALID_PARAMETER if an invalid parameter value was detected.

Description

This function solos audio output from one or more Groups. You specify the target Group(s) using the Group Flags. Use the bitwise OR operator to make multiple selections to create a bit field of Groups to solo.

See Also

sceScreamSetGroupMute()

sceScreamSetGroupVoiceOutputDest

Sets voice output destination for a Group.

Definition

```
int32_t sceScreamSetGroupVoiceOutputDest(
    int32_t group,
    int32_t outputDest
);
```

Arguments

group	(Input) Index of the target Group for which to set a voice output destination. One of the Volume Groups constants.
outputDest	(Input) Index of the output destination to set. For the master output, use SCE_SCREAM_SND_OUTPUT_DEST_MASTER. To specify an allocated pre-master submix buss, use the number of the desired submix, indexing from zero. Range: SCE_SCREAM_SND_OUTPUT_DEST_PREMASTER_0 to (SCE_SCREAM_SND_MAX_PREMASTER_SUBMIXES – 1).

Return Values

Returns SCE_SCREAM_SS_ERROR_OK if the operation was successful, otherwise returns SCE_SCREAM_SS_ERROR_INVALID_PARAMETER if an invalid parameter value was detected.

Description

This function sets voice output destination on a Group basis. It sets the output destination of all voices associated with all Sounds assigned to a Group.

To set a Group's output destination to master output, set *outputDest* to SCE_SCREAM_SND_OUTPUT_DEST_MASTER. To set output destination to one of the pre-master submixes, set *outputDest* to the (zero-based) index of the desired submix; counting from SCE_SCREAM_SND_OUTPUT_DEST_PREMASTER_0 for submix buss number 1.

Notes

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

See Also

sceScreamSetGroupVoiceRange(), sceScreamPlaySoundByIndexEx(), sceScreamPlaySoundByNameEx()

sceScreamSetGroupVoiceRange

Sets the voice allocation range for a Group.

Definition

```
int32_t sceScreamSetGroupVoiceRange(
    int32_t group,
    int32_t min,
    int32_t max
);
```

Arguments

group	(Input) Index of the Group to set a voice allocation range for. One of the Volume Groups constants.
min	(Input) Starting index of the voice allocation. Range: 0 to (sceScreamGetMaxPolyphony() – 1).
max	(Input) Ending index of the voice allocation. Range: *min* to (sceScreamGetMaxPolyphony() – 1).

Return Values

Returns SCE_SCREAM_SS_ERROR_OK if the operation was successful, otherwise returns SCE_SCREAM_SS_ERROR_INVALID_PARAMETER if an invalid parameter value was detected.

Description

This function allows you to allocate a range of voices per Group, and provides a way to guarantee voice availability. In the default voice allocation, all Sounds share the full pool of voices available on the synthesizer.

The *min* and *max* parameters specify the lower and upper limits of an inclusive voice allocation range assigned to the specified Group. The upper limit of this range is Scream-specific, and can be determined using the sceScreamGetMaxPolyphony() function. However, because the *min* and *max* parameters specify a zero-based voice allocation index, and the value returned from sceScreamGetMaxPolyphony() is a one-based count of addressable voices, you must subtract 1 from the returned value. For example, if sceScreamGetMaxPolyphony() returns 192, you would use 191 as the upper limit of values for the *min* and *max* parameters.

For more information, see "Setting Group Voices Ranges" in the "Working with Groups" chapter of the *Scream Library Overview*.

See Also

sceScreamGetMaxPolyphony(), sceScreamGetSoundVoiceCount()

sceScreamSetMasterVolume

Sets the volume level of a Group, or the level of Scream's global volume.

Definition

```
int32_t sceScreamSetMasterVolume(
    int32_t which,
    float vol
);
```

Arguments

which	(Input) One of the Volume Group constants; see Volume Groups.
vol	(Input) Volume level to apply. Range: SCE_SCREAM_SND_MIN_GAIN to SCE_SCREAM_SND_MAX_GAIN.

Return Values

Returns SCE_SCREAM_SS_ERROR_OK if the operation was successful, otherwise returns SCE_SCREAM_SS_ERROR_INVALID_PARAMETER if an invalid parameter value was detected.

Description

Sounds in Scream are assigned to a Group specified in Bank contents. Scream scales each Sound based on its Group volume setting. The SCE_SCREAM_GROUP_MASTER_VOLUME setting scales the volumes of all Groups.

This function sets the volume of a Group or the level of Scream's global volume. The *which* parameter is the index of the Group to set a volume level for, and is specified as one of the Volume Group constants; see Volume Groups. Values for the *vol* parameter are expressed as a `float`, and must be within the range SCE_SCREAM_SND_MIN_GAIN to SCE_SCREAM_SND_MAX_GAIN.

Notes

This function fails if the *which* parameter is out of range. Because the function does not return a value, function calls with out-of-range *which* parameter values are ignored, and an error message is shown in standard output.

Setting a volume for SCE_SCREAM_GROUP_VOLUME_EXTERNAL has no affect.

See Also

sceScreamGetMasterVolume()

sceScreamSetMasterVolumeDucker

Activates (or deactivates) a volume ducker.

Definition

```
int32_t sceScreamSetMasterVolumeDucker(
    int32_t which,
    const SceScreamDuckerDef *state
);
```

Arguments

which	(Input) Zero-based index of the volume ducker to activate. Range: 0 to (SCE_SCREAM_SND_MAX_DUCKERS – 1).
state	(Input) Pointer to an initialized SceScreamDuckerDef data structure with which to activate the volume ducker. Set to NULL to deactivate the volume ducker.

Return Values

Returns SCE_SCREAM_SS_ERROR_OK if the operation was successful, otherwise returns SCE_SCREAM_SS_ERROR_INVALID_PARAMETER if an invalid parameter value was detected.

Description

Volume ducking is the technique of reducing the volume of certain Sounds in order to highlight other Sounds. For example, in a sports game, the volume level of crowd Sounds might be reduced during an announcement or commentary. Scream provides for up to SCE_SCREAM_SND_MAX_DUCKERS simultaneous volume duckers.

Use this function to activate a volume ducker. The *which* parameter is a zero-based index referencing one of the volume duckers. The *state* parameter points to an initialized SceScreamDuckerDef data structure defining the volume ducker's parameters. To deactivate a volume ducker, set the *state* parameter to NULL.

See Also

SceScreamDuckerDef, sceScreamGetMasterVolume(), sceScreamSetGroupVoiceRange(), sceScreamSetMasterVolume()

sceScreamStopAllSoundsInGroup

Stops all Sounds in one or more Groups.

Definition

```
int32_t sceScreamStopAllSoundsInGroup(
    uint32_t groups,
    int32_t behavior
);
```

Arguments

groups	(Input) A bit field indicating which Groups to stop. One or more of the Group Flags. Use the bitwise OR operator for multiple selections.
behavior	(Input) A choice of two stop behaviors: SCE_SCREAM_SND_STOP_BEHAVIOR_KEYOFF or SCE_SCREAM_SND_STOP_BEHAVIOR_SILENCE.

Return Values

Returns SCE_SCREAM_SS_ERROR_OK if the operation was successful, otherwise returns SCE_SCREAM_SS_ERROR_INVALID_PARAMETER if an invalid parameter value was detected.

Description

This function stops all Sounds belonging to the specified Group(s).

The *behavior* parameter provides a choice of two stop behaviors:

- SCE_SCREAM_SND_STOP_BEHAVIOR_KEYOFF: Performs a graceful stop, triggering any *On Stop Marker* grain events, and issuing key-off messages to active voices with ADSR Release settings.
- SCE_SCREAM_SND_STOP_BEHAVIOR_SILENCE: Performs an instantaneous stop.

Notes

Because Scream Groups are not persistent, a Sound belonging to a Group that has been stopped, and that is played after the sceScreamStopAllSoundsInGroup() call, will still play.

See Also

sceScreamStopSound(), sceScreamStopAllSounds(), sceScreamStopAllSoundsInBank()

Bank Functions

Summary

Bank functions load and manipulate Banks, and retrieve Bank information.

Function	Description
sceScreamBankGetNumSoundsInBank	Retrieves the number of Sounds in a Bank.
sceScreamBankGetSoundIndexByName	Retrieves the index of a Sound by reference to its name.
sceScreamBankGetSoundNameByIndex	Retrieves the name of a Sound by reference to its index within a Bank.
sceScreamBankIsSafeToDelete	Determines whether it is safe to delete memory allocated for a Bank.
sceScreamBankLoadEx	Loads a Bank file from disk.
sceScreamBankLoadFromMemEx	Loads a Bank from memory.
sceScreamFindLoadedBankByName	Retrieves the pointer to a loaded Bank by reference to its name.
sceScreamFindLoadedBankNameByPointer	Retrieves the name of a loaded Bank by reference to its SceScreamSFXBlock2 pointer.
sceScreamGetLastLoadError	Retrieves the last Bank load error condition.
sceScreamGetNextLoadedBank	Retrieves the next loaded Bank in the linked list of loaded Banks.
sceScreamStopAllSoundsInBank	Stops all Sounds in a Bank.
sceScreamUnloadBank	Unloads a loaded Bank – synchronously or asynchronously.

sceScreamBankGetNumSoundsInBank

Retrieves the number of Sounds in a Bank.

Definition

```
uint32_t sceScreamBankGetNumSoundsInBank(
    const SceScreamSFXBlock2 *bank
);
```

Arguments

bank	(Input) Handle of the Bank for which you wish to obtain the number of Sounds.

Return Values

If successful, returns the number of Sounds in the specified Bank. If not successful, returns 0.

Description

This function retrieves the number of Sounds resident in a specified Bank.

See Also

sceScreamBankGetSoundNameByIndex(), sceScreamBankGetSoundIndexByName()

Document serial number: 000004892117

sceScreamBankGetSoundIndexByName

Retrieves the index of a Sound by reference to its name.

Definition

```
bool sceScreamBankGetSoundIndexByName(
    const SceScreamSFXBlock2 *bank,
    const char *soundName,
    int16_t *outIndex,
    SceScreamSFXBlock2 **outBank
);
```

Arguments

bank	(Input) Handle of a Bank to search for the specified Sound name. Set to NULL to search all loaded Banks.
soundName	(Input) Name of the Sound from which to retrieve the index.
outIndex	(Output) Pointer to an int16_t variable in which to receive the Sound index – if found.
outBank	(Output) Pointer to a SceScreamSFXBlock2 pointer in which to receive the Bank pointer that contains the referenced Sound. Can be NULL if the Bank pointer is not desired.

Return Values

Returns TRUE if successful, otherwise returns FALSE.

Description

This function retrieves the index of a Sound (and its containing Bank) by reference to its name. You can optionally specify a SceScreamSFXBlock2 pointer if you only wish to search a single Bank.

See Also

sceScreamBankGetNumSoundsInBank(), sceScreamBankGetSoundNameByIndex()

sceScreamBankGetSoundNameByIndex

Retrieves the name of a Sound by reference to its index within a Bank.

Definition

```
bool sceScreamBankGetSoundNameByIndex(
    const SceScreamSFXBlock2 *bank,
    int16_t index,
    char soundName[SCE_SCREAM_SND_MAX_NAME_LENGTH]
);
```

Arguments

bank	(Input) Handle of the Bank from which to retrieve the Sound name.
index	(Input) Zero-based index of a Sound from which to retrieve the name.
soundName	(Output) A character array in which to receive the name of the Sound. Minimum length: SCE_SCREAM_SND_MAX_NAME_LENGTH.

Return Values

Returns TRUE if successful, otherwise returns FALSE.

Description

This function retrieves the name of a Sound by reference to its index within a Bank. It stores the name in a user-supplied character array.

See Also

sceScreamBankGetNumSoundsInBank(), sceScreamBankGetSoundIndexByName()

sceScreamBankIsSafeToDelete

Determines whether it is safe to delete memory allocated for a Bank.

Definition

```
bool sceScreamBankIsSafeToDelete(
    const SceScreamSFXBlock2 *bank
);
```

Arguments

bank	(Input) Handle of the Bank for which to determine delete status, as returned by the sceScreamBankLoadFromMemEx() function.

Return Values

This function returns TRUE when it is safe to free (or overwrite) the specified Bank's memory. Otherwise, it returns FALSE.

Description

This function determines whether memory allocated for a Bank can safely be deleted or otherwise overwritten. This function applies only to Banks that were originally loaded using the sceScreamBankLoadFromMemEx() function. If a Bank has been unloaded asynchronously using the sceScreamUnloadBank() function (that is, with the function's *synchronous* parameter set to FALSE), you can poll the Bank's status using this function. Once the function returns TRUE, it is safe to free (or overwrite) the specified Bank's memory. Before it returns TRUE, it must be assumed that the underlying synthesizer is still accessing the memory.

Notes

If the specified Bank was loaded using sceScreamBankLoadEx(), the function always returns FALSE. The allocated memory for Banks loaded using the sceScreamBankLoadEx() function is automatically freed upon unloading the Bank with a call to sceScreamUnloadBank().

See Also

sceScreamUnloadBank(), sceScreamBankLoadFromMemEx(), sceScreamBankLoadEx()

sceScreamBankLoadEx

Loads a Bank file from disk.

Definition

```
SceScreamSFXBlock2 *sceScreamBankLoadEx(
    const char *name,
    int32_t offset
);
```

Arguments

name	(Input) Name of the BNK file to load from disk.
offset	(Input) Offset number of bytes into a container file where the embedded BNK file begins. Set to zero if the BNK file is not embedded in a container file.

Return Values

If the load operation is successful, the function returns a SceScreamSFXBlock2 pointer to the loaded Bank. If the load fails, the function returns NULL.

Description

This function loads a Bank (BNK file) into main memory from the disk.

If a data management system is being used, in which BNK files are embedded into larger container files, the *offset* parameter provides a way to reference an embedded BNK file by specifying the number of bytes into the container file where the BNK file begins.

Notes

Whenever memory is requested for a Bank, the system passes the constant SCE_SCREAM_SND_MEM_USE_BANK to your custom SceScreamExternSndMemAlloc() *use* parameter.

If the Bank load operation fails, you can check the reason for the failure using the sceScreamGetLastLoadError() function.

Banks are platform-specific. Be sure that you are loading a Bank for the intended platform.

See Also

sceScreamBankLoadFromMemEx(), sceScreamFindLoadedBankByName(), sceScreamUnloadBank(), sceScreamGetLastLoadError()

sceScreamBankLoadFromMemEx

Loads a Bank from memory.

Definition

```
SceScreamSFXBlock2 *sceScreamBankLoadFromMemEx(
    void *loc
);
```

Arguments

loc	(Input) Pointer to a memory address where the Bank resides.

Return Values

If the load operation is successful, the function returns a SceScreamSFXBlock2 pointer to the loaded Bank. If the load fails, the function returns NULL.

Description

This function loads a Bank that is already resident in main memory. This has the effect of registering the Bank with Scream, enabling its Sounds to be played. It is important that the Bank remains resident in main memory until it is unloaded, using the sceScreamUnloadBank() function.

This function is useful if you want to page a Scream Bank into memory while the game is in progress. When the paging code has finished loading the Bank (from Blu-ray or HDD), you can register the memory-loaded Bank with Scream using this function. This is generally the preferred method, as the sceScreamBankLoadEx() function blocks the calling thread until the file has completed loading from disk. sceScreamBankLoadFromMemEx(), however, returns much more quickly.

Notes

If the Bank load operation fails, the reason for the failure can be checked using the sceScreamGetLastLoadError() function. If the failure is due to the *loc* parameter being passed an invalid memory pointer, the sceScreamGetLastLoadError() function returns SCE_SCREAM_SND_LOAD_ERROR_MEMORY.

Banks are platform-specific. Be sure that you are loading a Bank for the intended platform.

A small amount of memory (16 bytes or less) is required for Bank data management that is in addition to the actual Bank data. When calling this function, the system passes the constant SCE_SCREAM_SND_MEM_USE_BANK to your custom SceScreamExternSndMemAlloc() *use* parameter in requesting the additional memory, even though the Bank data itself is already in memory.

See Also

sceScreamBankLoadEx(), sceScreamFindLoadedBankByName(), sceScreamUnloadBank(), sceScreamGetLastLoadError()

sceScreamFindLoadedBankByName

Retrieves the pointer to a loaded Bank by reference to its name.

Definition

```
SceScreamSFXBlock2 *sceScreamFindLoadedBankByName(
    const char *bankName
);
```

Arguments

bankName	(Input) Name of the Bank for which to retrieve the pointer.

Return Values

Returns the SceScreamSFXBlock2 pointer for the specified Bank.

Description

This function retrieves the pointer to a loaded Bank that you reference by name. Because both functions for playing a Sound – sceScreamPlaySoundByIndexEx() and sceScreamPlaySoundByNameEx() – require a SceScreamSFXBlock2 pointer as the *bank* argument, this function is helpful in cases where a Bank's name is known, but not its SceScreamSFXBlock2 pointer.

See Also

sceScreamBankLoadEx(), sceScreamBankLoadFromMemEx(), sceScreamUnloadBank(), sceScreamFindLoadedBankNameByPointer(), sceScreamGetNextLoadedBank()

sceScreamFindLoadedBankNameByPointer

Retrieves the name of a loaded Bank by reference to its SceScreamSFXBlock2 pointer.

Definition

```
bool sceScreamFindLoadedBankNameByPointer(
    SceScreamSFXBlock2 *bank,
    char outBankName[SCE_SCREAM_SND_MAX_BANK_NAME_LENGTH]
);
```

Arguments

bank	(Input) Pointer of a Bank for which to retrieve the name.
outBankName	(Output) Pointer to a variable in which to store the Bank name.

Return Values

Returns TRUE if the Bank name was found, otherwise returns FALSE. If the referenced Bank is loaded but not named, the function returns TRUE — but the value stored in the *outBankName* variable is set to all zeros.

Description

This function retrieves the name of a loaded Bank by reference to its SceScreamSFXBlock2 pointer.

See Also

sceScreamBankLoadEx(), sceScreamBankLoadFromMemEx(), sceScreamFindLoadedBankByName(), sceScreamGetNextLoadedBank()

sceScreamGetLastLoadError

Retrieves the last Bank load error condition.

Definition

```
int32_t sceScreamGetLastLoadError(void);
```

Return Values

Returns the error code generated during the last failed Bank load operation.

Description

This function retrieves the last error condition set in relation to a Bank load operation. The resulting error condition will either be an OS-specific file system error code, or one of the following Scream-specific error codes:

SCE_SCREAM_SND_LOAD_ERROR_COULDNT_OPEN_FILE
SCE_SCREAM_SND_LOAD_ERROR_READING_FILE
SCE_SCREAM_SND_LOAD_ERROR_MEMORY
SCE_SCREAM_SND_LOAD_ERROR_ALIGNMENT
SCE_SCREAM_SND_LOAD_ERROR_INVALID_FORMAT
SCE_SCREAM_SND_LOAD_ERROR_ALREADY_LOADED

See Also

sceScreamBankLoadEx(), sceScreamBankLoadFromMemEx(), sceScreamUnloadBank()

sceScreamGetNextLoadedBank

Retrieves the next loaded Bank in the linked list of loaded Banks.

Definition

```
SceScreamSFXBlock2 *sceScreamGetNextLoadedBank(
    SceScreamSFXBlock2 *prevBank
);
```

Arguments

prevBank	(Input) SceScreamSFXBlock2 pointer of the Bank that comes immediately before the SceScreamSFXBlock2 pointer to retrieve. Can be NULL, in which case the pointer to the head of the Bank linked list is returned.

Return Values

Returns the SceScreamSFXBlock2 pointer immediately following the pointer passed in as *prevBank*. If there is no next Bank, the function returns NULL.

Description

This function retrieves the next Bank in the linked list of loaded Banks. It can be used to iterate over all loaded Banks.

See Also

sceScreamBankLoadEx(), sceScreamBankLoadFromMemEx(), sceScreamFindLoadedBankByName(), sceScreamFindLoadedBankNameByPointer()

sceScreamStopAllSoundsInBank

Stops all Sounds in a Bank.

Definition

```
int32_t sceScreamStopAllSoundsInBank(
    const SceScreamSFXBlock2 *bank,
    int32_t behavior
);
```

Arguments

bank	(Input) Handle of the Bank that you wish to stop, as returned by the sceScreamBankLoadEx() or sceScreamBankLoadFromMemEx() functions.
behavior	(Input) A choice of two stop behaviors: SCE_SCREAM_SND_STOP_BEHAVIOR_KEYOFF or SCE_SCREAM_SND_STOP_BEHAVIOR_SILENCE.

Return Values

Returns SCE_SCREAM_SS_ERROR_OK if the operation was successful, otherwise returns SCE_SCREAM_SS_ERROR_INVALID_PARAMETER if an invalid parameter value was detected.

Description

This function stops all Sounds in a specified Bank.

The *behavior* parameter provides a choice of two stop behaviors:

- SCE_SCREAM_SND_STOP_BEHAVIOR_KEYOFF: Performs a graceful stop, triggering any *On Stop Marker* grain events, and issuing key-off messages to active voices with ADSR Release settings.
- SCE_SCREAM_SND_STOP_BEHAVIOR_SILENCE: Performs an instantaneous stop.

Notes

sceScreamStopAllSoundsInBank() is called automatically by the sceScreamUnloadBank() function.

See Also

sceScreamPlaySoundByIndexEx(), sceScreamPlaySoundByNameEx(), sceScreamStopAllSounds(), sceScreamStopSound(), sceScreamStopAllSoundsInGroup()

sceScreamUnloadBank

Unloads a loaded Bank – synchronously or asynchronously.

Definition

```
bool sceScreamUnloadBank(
    const SceScreamSFXBlock2 *bank,
    bool synchronous = true
);
```

Arguments

bank	(Input) Handle of the Bank you wish to unload, as returned by the sceScreamBankLoadEx() or sceScreamBankLoadFromMemEx() functions.
synchronous	(Input) A Boolean specifying whether or not the function should execute synchronously. Specify TRUE for synchronous (blocking) execution. With synchronous execution, it is safe to free (or overwrite) the Bank's allocated memory by the time the function returns. Specify FALSE for asynchronous execution. With asynchronous execution, before freeing (or overwriting) the Bank's allocated memory, you must poll the Bank's status using the sceScreamBankIsSafeToDelete() function until that function returns TRUE, indicating it is safe to do so. While the default value is TRUE, it is recommended to set this parameter to FALSE. See "Notes" below.

Return Values

Returns TRUE if successful, otherwise returns FALSE.

Description

This function unloads a loaded Bank. The *synchronous* parameter allows you to specify synchronous or asynchronous unloading.

If the Bank had been loaded using the sceScreamBankLoadEx() function, the memory it occupies is freed automatically with an internal call to the registered memory free function (see sceScreamStartSoundSystemEx2()).

If the Bank had been loaded using the sceScreamBankLoadFromMemEx() function, it is simply unregistered (rather than unloaded). **Note:** In this case, the application is responsible for freeing the memory occupied by the unloaded Bank.

Notes

In the interests of maintaining backwards compatibility, the *synchronous* parameter defaults to TRUE. However, using synchronous unloading can sleep the calling thread for a significant time interval, and is not recommended.

If a Bank was loaded directly from memory (using sceScreamBankLoadFromMemEx()), you must first unload it — to unregister the Bank from Scream — before freeing its allocated memory.

This function fails if a Bank relocation operation is currently in progress.

See Also

sceScreamBankLoadEx(), sceScreamBankLoadFromMemEx(), sceScreamBankIsSafeToDelete(), sceScreamGetLastLoadError()

Sound Functions

Summary

Sound instance functions start and stop Sounds, dynamically manipulate their parameters, and set and retrieve Sound state information.

Function	Description
sceScreamAutoGain	Changes the gain of a Sound over a specified time.
sceScreamAutoPan	Changes the panning azimuth of a Sound over a specified time.
sceScreamAutoPitchBend	Changes the pitchbend factor of a Sound over a specified time.
sceScreamAutoPitchTranspose	Changes the pitch transposition of a Sound over a specified time.
sceScreamContinueSound	Continues a paused Sound.
sceScreamGetActiveStreamHandle	Retrieves an active stream handle associated with a Sound instance.
sceScreamGetAllSoundReg	Gets the values of all local registers specific to a Sound.
sceScreamGetLocalVariableByHash	Retrieves the value of a local variable.
sceScreamGetNumActiveStreamHandles	Retrieves the number of active stream handles associated with a Sound instance.
sceScreamGetSoundGainComponents	Retrieves the gain components that comprise a Sound's aggregate gain level.
sceScreamGetSoundIndexDesignerParams	Retrieves a Sound's associated designer parameters by reference to its index.
sceScreamGetSoundIndexUserDataPtr	Retrieves a Sound's associated user data values by reference to its index.
sceScreamGetSoundIndexVolumeGroup	Retrieves the volume group to which a Sound is assigned by reference to its index.
sceScreamGetSoundInstanceDesignerParams	Retrieves a Sound's associated designer parameters by reference to its instance handle.
sceScreamGetSoundInstanceUserDataPtr	Retrieves a Sound's associated user data values by reference to its instance handle.
sceScreamGetSoundInstanceVolumeGroup	Retrieves the volume group to which a Sound is assigned by reference to its instance handle.
sceScreamGetSoundNameDesignerParams	Retrieves a Sound's associated designer parameters by reference to its name.
sceScreamGetSoundNameUserDataPtr	Retrieves a Sound's associated user data values by reference to its name.
sceScreamGetSoundNameVolumeGroup	Retrieves the volume group to which a Sound is assigned by reference to its name.
sceScreamGetSoundPanAzimuthComponents	Retrieves the pan components that comprise a Sound's aggregate panning azimuth.
sceScreamGetSoundParamsEx	Retrieves an active Sound's parameter values.
sceScreamGetSoundPitchBendFactorComponents	Retrieves the pitchbend factor components that comprise a Sound's aggregate pitchbend factor.
sceScreamGetSoundPitchTransposeComponents	Retrieves the pitch transposition components that comprise a Sound's aggregate pitch transposition.
sceScreamGetSoundReg	Retrieves the value of a local register.
sceScreamGetSoundVoiceCount	Retrieves the number of voices being used by a Sound.
sceScreamIsSoundIndexALooper	Verifies whether an indexed Sound contains looping waveforms.

Function	Description
sceScreamIsSoundIndexAStreamer	Verifies whether an indexed Sound contains streaming content.
sceScreamIsSoundInstanceALooper	Verifies whether a Sound instance is looping.
sceScreamIsSoundInstanceAStreamer	Verifies whether a Sound instance contains streaming content.
sceScreamIsSoundNameALooper	Verifies whether a named Sound contains looping waveforms.
sceScreamIsSoundNameAStreamer	Verifies whether a named Sound contains streaming content.
sceScreamLockAllSoundReg	Locks a Sound's local registers and retrieves their current values.
sceScreamOutputAllPlayingSoundInfoToTTY	Outputs information about all active Sound instances to the TTY.
sceScreamOutputHandlerInfoToTTY	Outputs information about an active Sound instance to the TTY.
sceScreamPauseSound	Pauses a Sound.
sceScreamPlaySoundByIndexEx	Plays a Sound by reference to its index within a Bank.
sceScreamPlaySoundByNameEx	Plays a Sound by reference to its name within a Bank.
sceScreamSetAllSoundReg	Sets the values of all local registers specific to a Sound.
sceScreamSetLocalVariableByHash	Sets the value of a local variable.
sceScreamSetSoundInstanceLFO	Initializes an LFO or updates parameters on a running LFO.
sceScreamSetSoundParamsEx	Sets an active Sound's parameter values.
sceScreamSetSoundReg	Sets the value of a local register.
sceScreamSoundIndexGet3DDesignerParams	Retrieves asset Grain 3D parameter data by reference to a Sound index.
sceScreamSoundIndexHasOnStopMarker	Verifies whether an indexed Sound contains an *On Stop Marker* Grain.
sceScreamSoundInstanceGet3DComponents	Retrieves dynamic 3D components for Grains associated with a Sound instance.
sceScreamSoundInstanceGet3DDesignerParams	Retrieves asset Grain 3D parameter data by reference to a Sound instance handle.
sceScreamSoundInstanceHasOnStopMarker	Verifies whether a Sound instance contains an *On Stop Marker* Grain.
sceScreamSoundIsStillPlaying	Verifies whether a Sound instance is still playing.
sceScreamSoundNameGet3DDesignerParams	Retrieves asset Grain 3D parameter data by reference to a Sound name.
sceScreamSoundNameHasOnStopMarker	Verifies whether a named Sound contains an *On Stop Marker* Grain.
sceScreamStopSound	Stops a Sound.
sceScreamUnlockAllSoundReg	Sets a Sound's local register values and unlocks them.

sceScreamAutoGain

Changes the gain of a Sound over a specified time.

Definition

```
uint32_t sceScreamAutoGain(
    uint32_t handle,
    float targetGain,
    float timeToTarget,
    uint32_t behaviorFlags
);
```

Arguments

handle	(Input) Handle of the Sound to which to apply a volume change.
targetGain	(Input) Target gain. Range: SCE_SCREAM_SND_MIN_GAIN to SCE_SCREAM_SND_MAX_GAIN.
timeToTarget	(Input) Time taken to reach the target volume in seconds.
behaviorFlags	(Input) Zero or more of the Automated Parameter Change constants (bitwise OR multiple selections). See Automated Parameter Change Flags.

Return Values

Returns the specified Sound handle if the automated gain change is successful. Returns 0 if the specified Sound is no longer active.

Description

This function performs an automated change to the gain (volume) of a Sound. It incrementally changes the gain setting from its current value to a target value over a specified time.

Notes

By default, sceScreamAutoGain() targets the *apiGain* component of the SceScreamGainComponents structure filled by sceScreamGetSoundGainComponents(). However, if you set the SCE_SCREAM_SND_AUTO_USE_SEPARATE_FACTOR *behavior* flag when calling sceScreamAutoGain(), the function instead targets the *autoGain* component. For more information, see "Use Separate Factor" in the "Working with Sounds" chapter of *Scream Library Overview*.

See Also

sceScreamAutoPan(), sceScreamAutoPitchTranspose(), sceScreamAutoPitchBend(), SceScreamGainComponents, sceScreamGetSoundGainComponents()

sceScreamAutoPan

Changes the panning azimuth of a Sound over a specified time.

Definition

```
uint32_t sceScreamAutoPan(
    uint32_t handle,
    uint32_t targetPanAzimuth,
    float timeToTarget,
    uint32_t behaviorFlags
);
```

Arguments

handle	(Input) Handle of the Sound to which to apply a panning azimuth change.
targetPanAzimuth	(Input) The target panning azimuth in degrees. Range: 0 to 359.
timeToTarget	(Input) The time taken to reach the target panning azimuth in seconds.
behaviorFlags	(Input) Zero or more of the Automated Parameter Change constants (bitwise OR multiple selections). See Automated Parameter Change Flags.

Return Values

Returns the specified Sound handle if the automated pan change is successful. Returns 0 if the specified Sound is no longer active.

Description

This function performs an automated change to the panning azimuth of a Sound. It incrementally changes the azimuth setting from its current value to a target value over a specified time.

Notes

For a counter-clockwise panning motion, bitwise OR the SCE_SCREAM_SND_AUTO_COUNTER_CLOCKWISE constant with the *behaviorFlags* member.

By default, sceScreamAutoPan() targets the *apiPanAzimuth* component of the SceScreamPanAzimuthComponents structure filled by sceScreamGetSoundPanAzimuthComponents(). However, if you set the SCE_SCREAM_SND_AUTO_USE_SEPARATE_FACTOR *behavior* flag when calling sceScreamAutoPan(), the function instead targets the *autoPanAzimuth* component. For more information, see "Use Separate Factor" in the "Working with Sounds" chapter of *Scream Library Overview*.

The SCE_SCREAM_SND_AUTO_COUNTER_CLOCKWISE and the SCE_SCREAM_SND_AUTO_TAKE_SHORTEST_PATH constants are mutually exclusive.

See Also

sceScreamAutoPitchTranspose(), sceScreamAutoPitchBend(), sceScreamAutoGain(), SceScreamPanAzimuthComponents, sceScreamGetSoundPanAzimuthComponents()

sceScreamAutoPitchBend

Changes the pitchbend factor of a Sound over a specified time.

Definition

```
uint32_t sceScreamAutoPitchBend(
    uint32_t handle,
    float targetPitchBend,
    float timeToTarget,
    uint32_t behaviorFlags
);
```

Arguments

handle	(Input) Handle of the Sound to which to apply a pitchbend factor change.
targetPitchBend	(Input) The target pitchbend factor. Range: SCE_SCREAM_SND_MIN_PITCH_BEND_FACTOR to SCE_SCREAM_SND_MAX_PITCH_BEND_FACTOR.
timeToTarget	(Input) The time taken to reach the target pitchbend factor in seconds.
behaviorFlags	(Input) Zero or more of the Automated Parameter Change constants (bitwise OR multiple selections). See Automated Parameter Change Flags.

Return Values

Returns the specified Sound handle if the automated pitchbend change is successful. Returns 0 if the specified Sound is no longer active.

Description

This function performs an automated change to the pitchbend factor of a Sound. It incrementally changes the pitchbend factor setting from its current value to a target value over a specified time.

Notes

The limits, within which a Sound will bend up or down from its original pitch, are set in Bank contents. If the specified *targetPitchBend* value is positive, it is multiplied by the pitchbend upper limit. If the specified *targetPitchBend* value is negative, it is multiplied by the pitchbend lower limit. For example, suppose a Sound's pitchbend settings in Bank contents are 2 semitones on the upper limit, and 4 semitones on the lower limit. If *targetPitchBend* is 0.5, the pitchbend factor change stops at 1 semitone above the original pitch. If *targetPitchBend* is -0.5, the pitchbend factor change stops at 2 semitones below the original pitch.

By default, sceScreamAutoPitchBend() targets the *apiPitchBendFactor* component of the SceScreamPitchBendFactorComponents structure filled by sceScreamGetSoundPitchBendFactorComponents(). However, if you set the SCE_SCREAM_SND_AUTO_USE_SEPARATE_FACTOR *behavior* flag when calling sceScreamAutoPitchBend(), the function instead targets the *autoPitchBendFactor* component. For more information, see "Use Separate Factor" in the "Working with Sounds" chapter of *Scream Library Overview*.

See Also

sceScreamAutoPan(), sceScreamAutoPitchTranspose(), sceScreamAutoGain(), SceScreamPitchBendFactorComponents, sceScreamGetSoundPitchBendFactorComponents()

sceScreamAutoPitchTranspose

Changes the pitch transposition of a Sound over a specified time.

Definition

```
uint32_t sceScreamAutoPitchTranspose(
    uint32_t handle,
    int32_t targetPitchTranspose,
    float timeToTarget,
    uint32_t behaviorFlags
);
```

Arguments

handle	(Input) Handle of the Sound to which to apply a pitch transposition change.
targetPitchTranspose	(Input) The target pitch transposition in fines. Range: –SCE_SCREAM_SND_MAX_PITCH_TRANSPOSE to +SCE_SCREAM_SND_MAX_PITCH_TRANSPOSE. See "Notes" below.
timeToTarget	(Input) The time taken to reach the target pitch transposition in seconds.
behaviorFlags	(Input) Zero or more of the Automated Parameter Change constants (bitwise OR multiple selections). See Automated Parameter Change Flags.

Return Values

Returns the specified Sound handle if the automated pitch transpose change is successful. Returns 0 if the specified Sound is no longer active.

Description

This function performs an automated change to the pitch transposition of a Sound. It incrementally changes the pitch transposition setting from its current value to a target value over a specified time.

Notes

You specify pitch transposition units in fines. A fine is a 128th microtonal subdivision of a semitone. There are 128 × 12 = 1536 fines per octave, expressed in the constant SCE_SCREAM_SND_FINES_PER_OCTAVE. The maximum pitch transposition amount, up or down, is 5 octaves, expressed in the constant SCE_SCREAM_SND_MAX_PITCH_TRANSPOSE. Some pitch transposition examples:

- To transpose down one octave, set *targetPitchTranspose* to -1536.
- To transpose up one octave, set *targetPitchTranspose* to 1536.
- To transpose up 2 semitones, set *targetPitchTranspose* to 256.

By default, sceScreamAutoPitchTranspose() targets the *apiPitchTranspose* component of the SceScreamPitchTransposeComponents structure filled by sceScreamGetSoundPitchTransposeComponents(). However, if you set the SCE_SCREAM_SND_AUTO_USE_SEPARATE_FACTOR *behavior* flag when calling sceScreamAutoPitchTranspose(), the function instead targets the *autoPitchTranspose* component. For more information, see "Use Separate Factor" in the "Working with Sounds" chapter of *Scream Library Overview*.

See Also

sceScreamAutoPan(), sceScreamAutoPitchBend(), sceScreamAutoGain(), SceScreamPitchTransposeComponents, sceScreamGetSoundPitchTransposeComponents(), sceScreamGetDopplerPitchTranspose()

sceScreamContinueSound

Continues a paused Sound.

Definition

```
uint32_t sceScreamContinueSound(
    uint32_t handle
);
```

Arguments

handle	(Input) Sound handle, as returned by sceScreamPlaySoundByIndexEx() or sceScreamPlaySoundByNameEx().

Return Values

Returns the specified handle if the Sound is continued. Returns 0 if the specified Sound is no longer active.

Description

This function continues a paused Sound and any Child Sounds it may contain. If the Sound is not currently paused, this function will have no effect.

Notes

This function continues a Sound regardless of how it was originally paused, that is, whether using sceScreamPauseSound(), sceScreamPauseAllSoundsInGroup(), and so on.

See Also

sceScreamPauseSound(), sceScreamPauseAllSoundsInGroup(), sceScreamContinueAllSoundsInGroup(), sceScreamPauseGroup(), sceScreamContinueGroup(), sceScreamReverbPause(), sceScreamReverbContinue()

sceScreamGetActiveStreamHandle

Retrieves an active stream handle associated with a Sound instance.

Definition

```
uint32_t sceScreamGetActiveStreamHandle(
    uint32_t handle,
    int16_t index
);
```

Arguments

handle	(Input) Handle of the Sound from which to obtain a stream handle.
index	(Input) Zero-based index of the stream handle to retrieve.

Return Values

Returns the indexed stream handle if it exists, otherwise `NULL`.

Description

This function returns an active stream handle associated with a specified Sound instance.

For details on working with Streams, see *Sndstream Library Overview* and *Sndstream Library Reference*.

See Also

sceScreamGetNumActiveStreamHandles()

sceScreamGetAllSoundReg

Gets the values of all local registers specific to a Sound.

Definition

```
uint32_t sceScreamGetAllSoundReg(
    uint32_t handle,
    int8_t *vals
);
```

Arguments

handle	(Input) Handle of the Sound for which to get all registers.
vals	(Output) An array (of length SCE_SCREAM_SND_MAX_REGISTERS) in which to store the register values. Range for each value: -128 to 127.

Return Values

Returns the specified handle if the Sound is still active. Returns 0 if the specified Sound is no longer active.

Description

This function gets the values of all local registers specific to a Sound.

Notes

To perform atomic read/modify/write operations on a Sound's local registers (that is, without interference from running scripts), use sceScreamLockAllSoundReg() and sceScreamUnlockAllSoundReg().

See Also

sceScreamSetAllSoundReg(), sceScreamSetSoundReg(), sceScreamGetSoundReg(), sceScreamLockAllSoundReg(), sceScreamUnlockAllSoundReg()

sceScreamGetLocalVariableByHash

Retrieves the value of a local variable.

Definition

```
bool sceScreamGetLocalVariableByHash(
    uint32_t handle,
    uint32_t nameHash,
    float *val
);
```

Arguments

handle	(Input) Handle of a Sound from which to retrieve a local variable value.
nameHash	(Input) 32-bit hash of a named local variable from which to retrieve the value.
val	(Output) A pointer to a floating variable in which to store the retrieved value.

Return Values

Returns TRUE if the specified Sound is active and the specified local variable exists. Returns FALSE otherwise.

Description

This function retrieves the value of a local variable associated with a Sound.

The maximum number of local variables per Sound instance is defined by the constant SCE_SCREAM_SND_MAX_LOCAL_VARIABLES.

You can read all a Sound's local variables at the same time by calling sceScreamGetSoundParamsEx() with a SceScreamSoundParams structure. The local variables' values are in the SceScreamSndLocalVarData structure, the *localVariableData* member of the SceScreamSoundParams structure.

See Also

sceScreamGetHashFromName(), sceScreamSetLocalVariableByHash(), SceScreamSndLocalVarData, SceScreamSoundParams, sceScreamGetSoundParamsEx()

sceScreamGetNumActiveStreamHandles

Retrieves the number of active stream handles associated with a Sound instance.

Definition

```
uint32_t sceScreamGetNumActiveStreamHandles(
    uint32_t handle
);
```

Arguments

handle	(Input) Handle of the Sound from which to obtain a stream handle count.

Return Values

Returns the number of active stream handles.

Description

This function returns the number of active stream handles associated with a specified Sound instance.

See Also

sceScreamGetActiveStreamHandle()

Document serial number: 000004892117

sceScreamGetSoundGainComponents

Retrieves the gain components that comprise a Sound's aggregate gain level.

Definition

```
uint32_t sceScreamGetSoundGainComponents(
    uint32_t handle,
    SceScreamGainComponents *components
);
```

Arguments

handle	(Input) Sound handle, as returned by sceScreamPlaySoundByIndexEx() or sceScreamPlaySoundByNameEx().
components	(Output) Pointer to a SceScreamGainComponents structure in which to receive the Sound's gain components.

Return Values

Returns the specified handle if the Sound is still active. Returns 0 if the specified Sound is no longer active.

Description

A Sound's aggregate gain level is comprised of a number of components. This function retrieves all of a Sound's gain components and stores them in a SceScreamGainComponents structure.

Notes

You can determine the Sound's aggregate gain level by multiplying together the values of its constituent gain components.

You may still hear a Sound's output even if its aggregate gain components calculation comes to zero. In such a case, the Sound's auxiliary send(s) may be non-zero, causing signal to feed into an auxiliary buss, and subsequently through an effect and back into the master buss.

See Also

SceScreamGainComponents, sceScreamGetSoundPanAzimuthComponents(), sceScreamGetSoundPitchTransposeComponents(), sceScreamGetSoundPitchBendFactorComponents(), sceScreamGetSoundParamsEx()

sceScreamGetSoundIndexDesignerParams

Retrieves a Sound's associated designer parameters by reference to its index.

Definition

```
bool sceScreamGetSoundIndexDesignerParams(
    const SceScreamSFXBlock2 *bank,
    int16_t index,
    SceScreamDesignerParams *designerParams
);
```

Arguments

bank	(Input) Pointer to a loaded Bank, as returned from sceScreamBankLoadEx(), sceScreamBankLoadFromMemEx(), sceScreamFindLoadedBankByName(), or sceScreamGetNextLoadedBank().
index	(Input) Index of the Sound within the specified *bank* for which to obtain associated designer parameters.
designerParams	(Output) A pointer to a SceScreamDesignerParams structure to be filled out by this function.

Return Values

If successful, returns TRUE. If not successful, returns FALSE.

Description

This function retrieves the designer parameter values associated with a Sound, by reference to its index within a Bank.

See Also

sceScreamGetSoundNameDesignerParams(), sceScreamGetSoundInstanceDesignerParams(), SceScreamDesignerParams

sceScreamGetSoundIndexUserDataPtr

Retrieves a Sound's associated user data values by reference to its index.

Definition

```
const uint32_t *sceScreamGetSoundIndexUserDataPtr(
    const SceScreamSFXBlock2 *bank,
    int16_t index
);
```

Arguments

bank	(Input) Pointer to a loaded Bank, as returned from sceScreamBankLoadEx(), sceScreamBankLoadFromMemEx(), sceScreamFindLoadedBankByName(), or sceScreamGetNextLoadedBank().
index	(Input) Index of the Sound within the Bank for which to obtain associated user data.

Return Values

If successful, returns a pointer to an array of twelve uint32_t values representing the Sound's associated user data. If not successful, returns NULL.

Description

User data values are (up to) twelve values associated with a Sound that can optionally be entered by audio designers using the Scream Tool Sound Properties panel. This function retrieves the user data values associated with a Sound, by reference to its index within a Bank. For more information, see "User Data?" in the "Working with Sounds" chapter of the *Scream Library Overview*.

See Also

sceScreamGetSoundNameUserDataPtr(), sceScreamGetSoundInstanceUserDataPtr()

sceScreamGetSoundIndexVolumeGroup

Retrieves the volume group to which a Sound is assigned by reference to its index.

Definition

```
int32_t sceScreamGetSoundIndexVolumeGroup(
    const SceScreamSFXBlock2 *bank,
    int16_t index
);
```

Arguments

bank	(Input) Pointer to a loaded Bank, as returned by sceScreamBankLoadEx(), sceScreamBankLoadFromMemEx(), sceScreamFindLoadedBankByName(), or sceScreamGetNextLoadedBank().
index	(Input) Index of the Sound within the specified Bank.

Return Values

If successful, returns the volume group to which the Sound is assigned (see Volume Groups). If not successful, returns -1.

Description

This function retrieves the volume group to which a Sound is assigned by reference to the Sound's index within a loaded Bank.

See Also

sceScreamGetSoundNameVolumeGroup(), sceScreamGetSoundInstanceVolumeGroup()

sceScreamGetSoundInstanceDesignerParams

Retrieves a Sound's associated designer parameters by reference to its instance handle.

Definition

```
uint32_t sceScreamGetSoundInstanceDesignerParams(
    uint32_t handle,
    SceScreamDesignerParams *designerParams
);
```

Arguments

handle	(Input) Handle of the Sound for which to obtain associated designer parameters.
designerParams	(Output) A pointer to a SceScreamDesignerParams structure to be filled out by this function.

Return Values

Returns the specified handle if the query is successful. Returns 0 if the specified Sound is no longer active.

Description

This function retrieves associated designer parameter values from a Sound instance, by reference to its handle.

Notes

Unlike sceScreamGetSoundIndexDesignerParams() and sceScreamGetSoundNameDesignerParams(), which operate on a Sound as stored in Bank contents, this function operates on a running instance of a Sound.

See Also

sceScreamGetSoundIndexDesignerParams(), sceScreamGetSoundNameDesignerParams(), SceScreamDesignerParams

sceScreamGetSoundInstanceUserDataPtr

Retrieves a Sound's associated user data values by reference to its instance handle.

Definition

```
const uint32_t *sceScreamGetSoundInstanceUserDataPtr(
    uint32_t handle
);
```

Arguments

handle	(Input) Handle of the Sound for which to obtain associated user data.

Return Values

If successful, returns a pointer to an array of twelve `uint32_t` values representing the Sound's associated user data. If not successful, returns `NULL`.

Description

User data values are (up to) twelve values associated with a Sound that can optionally be entered by audio designers using the Scream Tool Sound Properties panel. This function retrieves the user data values associated with a Sound, by reference to its handle. For more information, see "User Data?" in the "Working with Sounds" chapter of the *Scream Library Overview*.

Notes

Unlike `sceScreamGetSoundIndexUserDataPtr()` and `sceScreamGetSoundNameUserDataPtr()`, which operate on a Sound as stored in Bank contents, this function operates on a running instance of a Sound.

See Also

`sceScreamGetSoundIndexUserDataPtr()`, `sceScreamGetSoundNameUserDataPtr()`

sceScreamGetSoundInstanceVolumeGroup

Retrieves the volume group to which a Sound is assigned by reference to its instance handle.

Definition

```
int32_t sceScreamGetSoundInstanceVolumeGroup(
    uint32_t handle
);
```

Arguments

handle	(Input) Handle of the Sound instance for which to retrieve the assigned volume Group.

Return Values

If successful, returns the volume group to which the Sound is assigned (see Volume Groups). If not successful, returns -1.

Description

This function retrieves the volume group to which a Sound is assigned by reference to the Sound's instance handle.

Notes

Unlike sceScreamGetSoundIndexVolumeGroup() and sceScreamGetSoundNameVolumeGroup(), which operate on a Sound as stored in Bank contents, this function operates on a running instance of a Sound.

See Also

sceScreamGetSoundIndexVolumeGroup(), sceScreamGetSoundNameVolumeGroup()

sceScreamGetSoundNameDesignerParams

Retrieves a Sound's associated designer parameters by reference to its name.

Definition

```
bool sceScreamGetSoundNameDesignerParams(
    const SceScreamSFXBlock2 *bank,
    const char *name,
    SceScreamDesignerParams *designerParams
);
```

Arguments

bank	(Input) Pointer to a loaded Bank, as returned from sceScreamBankLoadEx(), sceScreamBankLoadFromMemEx(), sceScreamFindLoadedBankByName(), or sceScreamGetNextLoadedBank(). Can be NULL. See "Notes" below.
name	(Input) Name of the Sound within the specified *bank* for which to obtain associated designer parameters.
designerParams	(Output) A pointer to a SceScreamDesignerParams structure to be filled out by this function.

Return Values

If successful, returns TRUE. If not successful, returns FALSE.

Description

This function retrieves the designer parameter values associated with a Sound, by reference to its name within a Bank.

Notes

If *bank* is NULL, this function searches through all registered Banks, and returns designer parameters for the first Sound it finds matching the specified *name*. If *name* is not found in any Bank, the function returns FALSE.

See Also

sceScreamGetSoundIndexDesignerParams(), sceScreamGetSoundInstanceDesignerParams(), SceScreamDesignerParams

sceScreamGetSoundNameUserDataPtr

Retrieves a Sound's associated user data values by reference to its name.

Definition

```
const uint32_t *sceScreamGetSoundNameUserDataPtr(
    const SceScreamSFXBlock2 *bank,
    const char *name
);
```

Arguments

bank	(Input) Pointer to a loaded Bank, as returned from sceScreamBankLoadEx(), sceScreamBankLoadFromMemEx(), sceScreamFindLoadedBankByName(), or sceScreamGetNextLoadedBank(). Can be NULL. See "Notes" below.
name	(Input) Name of the Sound within the Bank for which to obtain associated user data.

Return Values

If successful, returns a pointer to an array of twelve uint32_t values representing the Sound's associated user data. If not successful, returns NULL.

Description

User data values are (up to) twelve values associated with a Sound that can optionally be entered by audio designers using the Scream Tool Sound Properties panel. This function retrieves the user data values associated with a Sound, by reference to its name within a Bank. For more information, see "User Data?" in the "Working with Sounds" chapter of the *Scream Library Overview*.

Notes

If *bank* is NULL, this function searches through all registered Banks, and returns user data for the first Sound it finds matching the specified *name*. If *name* is not found in any Bank, the function returns NULL.

See Also

sceScreamGetSoundIndexUserDataPtr(), sceScreamGetSoundInstanceUserDataPtr()

sceScreamGetSoundNameVolumeGroup

Retrieves the volume group to which a Sound is assigned by reference to its name.

Definition

```
int32_t sceScreamGetSoundNameVolumeGroup(
    const SceScreamSFXBlock2 *bank,
    const char *name
);
```

Arguments

bank	(Input) Pointer to a loaded Bank, as returned by sceScreamBankLoadEx(), sceScreamBankLoadFromMemEx(), sceScreamFindLoadedBankByName(), or sceScreamGetNextLoadedBank().
name	(Input) Name of the Sound within the specified Bank.

Return Values

If successful, returns the volume group to which the Sound is assigned (see Volume Groups). If not successful, returns -1.

Description

This function retrieves the volume group to which a Sound is assigned by reference to the Sound's name within a loaded Bank.

See Also

sceScreamGetSoundIndexVolumeGroup(), sceScreamGetSoundInstanceVolumeGroup()

sceScreamGetSoundPanAzimuthComponents

Retrieves the pan components that comprise a Sound's aggregate panning azimuth.

Definition

```
uint32_t sceScreamGetSoundPanAzimuthComponents(
    uint32_t handle,
    SceScreamPanAzimuthComponents *components
);
```

Arguments

handle	(Input) Sound handle, as returned by sceScreamPlaySoundByIndexEx() or sceScreamPlaySoundByNameEx().
components	(Output) Pointer to a SceScreamPanAzimuthComponents structure in which to receive the Sound's pan components.

Return Values

Returns the specified handle if the Sound is still active. Returns 0 if the specified Sound is no longer active.

Description

A Sound's aggregate panning azimuth is comprised of a number of components. This function retrieves all of a Sound's pan components and stores them in a SceScreamPanAzimuthComponents structure.

Notes

You can determine a Sound's aggregate panning azimuth by adding together the values of its constituent pan components modulo 360.

See Also

SceScreamPanAzimuthComponents, sceScreamGetSoundPitchTransposeComponents(), sceScreamGetSoundPitchBendFactorComponents(), sceScreamGetSoundGainComponents(), sceScreamGetSoundParamsEx()

sceScreamGetSoundParamsEx

Retrieves an active Sound's parameter values.

Definition

```
uint32_t sceScreamGetSoundParamsEx(
    uint32_t handle,
    SceScreamSoundParams *params
);
```

Arguments

handle	(Input) Handle of the Sound for which to retrieve parameter values.
params	(Output) Pointer to a SceScreamSoundParams data structure, into which the retrieved parameter values are stored. The SceScreamSynthParams structure in the SceScreamSoundParams's *synthParams* member may also be filled; see "Notes" below. Do not provide a pointer to a SceScreamSynthParams structure in *synthParams*; that can also be filled by the function.

Return Values

Returns the handle of the specified Sound. Returns 0 if the Sound is no longer active or if SceScreamSoundParams.*size* specified in the *params* member is invalid. Even though the supplied SceScreamSoundParams structure is filled by this function, you must set its *size* member appropriately to indicate that the properly size structure is provided.

Description

This function retrieves parameter values for an actively playing Sound. The retrieved values are stored in the SceScreamSoundParams instance pointed to in the *params* argument.

Notes

If the SCE_SCREAM_SND_MASK_SYNTH_PARAMS bit is set in the SceScreamSoundParams *mask* member, this indicates that one or more synthesizer-specific parameters have been set from the Scream API — overriding those settings in Bank contents — and that the SceScreamSynthParams structure in the *synthParams* parameter has the changed values. To see which synthesizer-specific parameters have been set from the Scream API, examine the value of the SceScreamSynthParams *mask* member.

If the SCE_SCREAM_SND_MASK_SYNTH_PARAMS bit is *not* set, then the synthesizer parameters are as specified in Bank contents and have not been changed by the Scream API. In this case, the corresponding SceScreamSoundParams *synthParams* member does not contain data. A Sound can contain multiple Grains, but synthesizer parameter settings in Bank contents are on a per-Grain basis, and therefore Sound settings cannot be represented in single SceScreamSynthParams data structure. For more information, see "Parent and Child Sounds" and "Scream Tool and API Parameter Settings" in the "System Overview" chapter of the *Scream Library Overview*.

See Also

sceScreamSetSoundParamsEx()

sceScreamGetSoundPitchBendFactorComponents

Retrieves the pitchbend factor components that comprise a Sound's aggregate pitchbend factor.

Definition

```
uint32_t sceScreamGetSoundPitchBendFactorComponents(
    uint32_t handle,
    SceScreamPitchBendFactorComponents *components
);
```

Arguments

handle	(Input) Sound handle, as returned by sceScreamPlaySoundByIndexEx() or sceScreamPlaySoundByNameEx().
components	(Output) Pointer to a SceScreamPitchBendFactorComponents structure in which to receive the Sound's pitchbend factor components.

Return Values

Returns the specified handle if the Sound is still active. Returns 0 if the specified Sound is no longer active.

Description

A Sound's aggregate pitchbend factor is comprised of a number of components. This function retrieves all of a Sound's pitchbend factor components and stores them in a SceScreamPitchBendFactorComponents structure.

Notes

You can determine a Sound's pitchbend factor amount by adding together the values of its constituent pitchbend components and clamping between SCE_SCREAM_SND_MIN_PITCH_BEND_FACTOR to SCE_SCREAM_SND_MAX_PITCH_BEND_FACTOR.

See Also

SceScreamPitchBendFactorComponents, sceScreamGetSoundPitchTransposeComponents(), sceScreamGetSoundGainComponents(), sceScreamGetSoundPanAzimuthComponents(), sceScreamGetSoundParamsEx()

sceScreamGetSoundPitchTransposeComponents

Retrieves the pitch transposition components that comprise a Sound's aggregate pitch transposition.

Definition

```
uint32_t sceScreamGetSoundPitchTransposeComponents(
    uint32_t handle,
    SceScreamPitchTransposeComponents *components
);
```

Arguments

handle	(Input) Sound handle, as returned by sceScreamPlaySoundByIndexEx() or sceScreamPlaySoundByNameEx().
components	(Output) Pointer to a SceScreamPitchTransposeComponents structure in which to receive the Sound's pitch transposition components.

Return Values

Returns the specified handle if the Sound is still active. Returns 0 if the specified Sound is no longer active.

Description

A Sound's aggregate pitch transposition is comprised of a number of components. This function retrieves all of a Sound's pitch transposition components and stores them in a SceScreamPitchTransposeComponents structure.

Notes

You can determine a Sound's aggregate pitch transposition by adding together the values of its constituent pitch transposition components and clamping to ±SCE_SCREAM_SND_MAX_PITCH_TRANSPOSE, relative to the original pitch.

See Also

SceScreamPitchTransposeComponents, sceScreamGetSoundPitchBendFactorComponents(), sceScreamGetSoundGainComponents(), sceScreamGetSoundPanAzimuthComponents(), sceScreamGetSoundParamsEx()

sceScreamGetSoundReg

Retrieves the value of a local register.

Definition

```
uint32_t sceScreamGetSoundReg(
    uint32_t handle,
    int32_t which,
    int8_t *val
);
```

Arguments

handle	(Input) Handle of the Sound from which to get the register value.
which	(Input) One-based index of the register to access. Range: 1 to SCE_SCREAM_SND_MAX_REGISTERS.
val	(Output) Pointer to a variable in which to store the register value. Range: -128 to 127.

Return Values

Returns the specified handle if the Sound is still active. Returns 0 if the specified Sound is no longer active.

Description

This function retrieves the current value of a local (Sound-specific) register.

See Also

sceScreamSetSoundReg(), sceScreamGetAllSoundReg(), sceScreamSetAllSoundReg()

sceScreamGetSoundVoiceCount

Retrieves the number of voices being used by a Sound.

Definition

```
uint32_t sceScreamGetSoundVoiceCount(
    uint32_t handle,
    uint32_t *outVoiceCount
);
```

Arguments

handle	(Input) Sound handle, as returned by sceScreamPlaySoundByIndexEx() or sceScreamPlaySoundByNameEx().
outVoiceCount	(Output) Pointer to a uint32_t variable in which to receive the voice count.

Return Values

Returns the specified handle if the Sound is still active. Returns 0 if the specified Sound is no longer active.

Description

This function retrieves the number of voices currently being used by the specified Sound instance. It stores the value in the variable referenced in the *outVoiceCount* parameter.

See Also

sceScreamGetMaxPolyphony(), sceScreamSetGroupVoiceRange()

sceScreamIsSoundIndexALooper

Verifies whether an indexed Sound contains looping waveforms.

Definition

```
int32_t sceScreamIsSoundIndexALooper(
    const SceScreamSFXBlock2 *bank,
    int16_t index
);
```

Arguments

bank	(Input) Pointer to a Bank that contains the referenced Sound, as returned by sceScreamBankLoadEx(), sceScreamBankLoadFromMemEx(), sceScreamFindLoadedBankByName(), or sceScreamGetNextLoadedBank().
index	(Input) Index of the Sound to obtain looping status for.

Return Values

Returns 0 if the Sound has no looping waveforms. Returns 1 if the Sound has one or more looping waveforms.

Description

This function verifies whether a Sound – referenced by its Bank index – contains any looping waveforms and is therefore considered a *looper*. It returns 1 or 0 to indicate looping status.

Notes

This function only tests for looping *Waveform* Grains. It does not detect the loop-oriented scripting Grains: *Loop Start*, *Loop Continue*, or *Loop End*.

See Also

sceScreamIsSoundNameALooper(), sceScreamIsSoundInstanceALooper()

sceScreamIsSoundIndexAStreamer

Verifies whether an indexed Sound contains streaming content.

Definition

```
int32_t sceScreamIsSoundIndexAStreamer(
    const SceScreamSFXBlock2 *bank,
    int16_t index
);
```

Arguments

bank	(Input) Pointer to a Bank that contains the referenced Sound, as returned by sceScreamBankLoadEx(), sceScreamBankLoadFromMemEx(), sceScreamFindLoadedBankByName(), or sceScreamGetNextLoadedBank().
index	(Input) Index of the Sound from which to obtain streaming status.

Return Values

Returns 1 if the Sound has streaming content. Returns 0 if the Sound has no streaming content.

Description

This function verifies whether a Sound – referenced by index within its Bank – contains any streaming content. Returns 1 or 0 to indicate streaming content status.

See Also

sceScreamIsSoundNameAStreamer(), sceScreamIsSoundInstanceAStreamer()

Document serial number: 000004892117

sceScreamIsSoundInstanceALooper

Verifies whether a Sound instance is looping.

Definition

```
int32_t sceScreamIsSoundInstanceALooper(
    uint32_t handle
);
```

Arguments

handle	(Input) Handle of the Sound for which to obtain looping status.

Return Values

Returns 0 if the Sound is not looping. Returns 1 if the Sound is looping.

Description

This function verifies whether a Sound instance is looping. It returns 1 or 0 to indicate looping status.

Notes

Unlike sceScreamIsSoundIndexALooper() and sceScreamIsSoundNameALooper(), which operate on a Sound as stored in Bank contents, this function operates on a running instance of a Sound.

See Also

sceScreamIsSoundIndexALooper(), sceScreamIsSoundNameALooper()

sceScreamIsSoundInstanceAStreamer

Verifies whether a Sound instance contains streaming content.

Definition

```
int32_t sceScreamIsSoundInstanceAStreamer(
    uint32_t handle
);
```

Arguments

handle	(Input) Handle of the Sound from which to obtain streaming status.

Return Values

Returns 1 if the Sound contains streaming content. Returns 0 if the Sound does not contain streaming content.

Description

This function verifies whether a Sound instance contains streaming content. Returns 1 or 0 to indicate streaming status.

Notes

Unlike sceScreamIsSoundIndexAStreamer() and sceScreamIsSoundNameAStreamer(), which operate on a Sound as stored in Bank contents, this function operates on a running instance of a Sound.

See Also

sceScreamIsSoundIndexAStreamer(), sceScreamIsSoundNameAStreamer()

sceScreamIsSoundNameALooper

Verifies whether a named Sound contains looping waveforms.

Definition

```
int32_t sceScreamIsSoundNameALooper(
    const SceScreamSFXBlock2 *bank,
    const char *name
);
```

Arguments

bank	(Input) Pointer to a Bank that contains the referenced Sound, as returned by sceScreamBankLoadEx(), sceScreamBankLoadFromMemEx(), sceScreamFindLoadedBankByName(), or sceScreamGetNextLoadedBank().
name	(Input) Name of the Sound for which to obtain looping status.

Return Values

Returns 0 if the Sound has no looping waveforms. Returns 1 if the Sound has one or more looping waveforms.

Description

This function verifies whether a Sound – referenced by name within its Bank – contains any looping waveforms and is therefore considered a *looper*. It returns 1 or 0 to indicate looping status.

Notes

This function only tests for looping *Waveform* Grains. It does not detect the loop-oriented scripting Grains: *Loop Start*, *Loop Continue*, or *Loop End*.

See Also

sceScreamIsSoundIndexALooper(), sceScreamIsSoundInstanceALooper()

sceScreamIsSoundNameAStreamer

Verifies whether a named Sound contains streaming content.

Definition

```
int32_t sceScreamIsSoundNameAStreamer(
    const SceScreamSFXBlock2 *bank,
    const char *name
);
```

Arguments

bank	(Input) Pointer to a Bank that contains the referenced Sound, as returned by sceScreamBankLoadEx(), sceScreamBankLoadFromMemEx(), sceScreamFindLoadedBankByName(), or sceScreamGetNextLoadedBank().
name	(Input) Name of the Sound from which to obtain streaming status.

Return Values

Returns 1 if the Sound has streaming content. Returns 0 if the Sound has no streaming content.

Description

This function verifies whether a Sound – referenced by name within its Bank – contains any streaming content. It returns 1 or 0 to indicate streaming status.

See Also

sceScreamIsSoundIndexAStreamer(), sceScreamIsSoundInstanceAStreamer()

sceScreamLockAllSoundReg

Locks a Sound's local registers and retrieves their current values.

Definition

```
uint32_t sceScreamLockAllSoundReg(
    uint32_t handle,
    int8_t *vals
);
```

Arguments

handle	(Input) Handle of the Sound from which to lock and retrieve all local register values.
vals	(Output) An array (of size SCE_SCREAM_SND_MAX_REGISTERS) in which to store the local register values. Range of each value: -128 to 127.

Return Values

Returns the specified handle if the Sound is still active. Returns 0 if the specified Sound is no longer active.

Description

This function locks a Sound's local registers, and retrieves their current values. You can then process the register values with the knowledge that a script cannot modify them during processing time.

Note: In order to lock a Sound's local registers, the function effectively locks the entire underlying rendering synthesizer. See warning in "Notes" below.

Notes

WARNING: Because this function locks the underlying rendering synthesizer, it is critical that calls to it are followed by a matching call to sceScreamUnlockAllSoundReg(). Processing performed between these two calls should be kept to an absolute minimum!

If you do not need an atomic read/modify/write capability on a Sound's local registers (that is, without interference from running scripts), use sceScreamGetAllSoundReg() and sceScreamSetAllSoundReg() instead.

See Also

sceScreamUnlockAllSoundReg(), sceScreamGetAllSoundReg(), sceScreamSetAllSoundReg(), sceScreamGetSoundReg()

sceScreamOutputAllPlayingSoundInfoToTTY

Outputs information about all active Sound instances to the TTY.

Definition

```
int32_t sceScreamOutputAllPlayingSoundInfoToTTY(
    uint32_t flags
);
```

Arguments

flags	(Input) Any combination of TTY Output Flags.

Return Values

Returns SCE_SCREAM_SS_ERROR_OK if the operation was successful.

Description

This function outputs information about all active Sound instances to the TTY.

See Also

sceScreamOutputHandlerInfoToTTY(), SceScreamSndDebugHandler(), sceScreamSetDebugHandler(), sceScreamPlaySoundByIndexEx(), sceScreamPlaySoundByNameEx()

sceScreamOutputHandlerInfoToTTY

Outputs information about an active Sound instance to the TTY.

Definition

```
uint32_t sceScreamOutputHandlerInfoToTTY(
    uint32_t handle,
    uint32_t flags
);
```

Arguments

handle	(Input) Sound handle, as returned by sceScreamPlaySoundByIndexEx() or sceScreamPlaySoundByNameEx().
flags	(Input) Any combination of TTY Output Flags.

Return Values

Returns the specified handle if the Sound is still active. Returns 0 if the specified Sound is no longer active.

Description

This function outputs information about an active Sound instance to the TTY.

See Also

sceScreamOutputAllPlayingSoundInfoToTTY(), SceScreamSndDebugHandler(), sceScreamSetDebugHandler(), sceScreamPlaySoundByIndexEx(), sceScreamPlaySoundByNameEx()

sceScreamPauseSound

Pauses a Sound.

Definition

```
uint32_t sceScreamPauseSound(
    uint32_t handle
);
```

Arguments

handle	(Input) Sound handle, as returned by sceScreamPlaySoundByIndexEx() or sceScreamPlaySoundByNameEx().

Return Values

Returns the specified handle if the Sound is paused. Returns 0 if the specified Sound is no longer active.

Description

This function pauses the specified Sound and any Child Sounds it contains. If the Sound is already paused, this function has no effect. Pausing a Sound does not free the voice(s) it is using. If you start the same Sound from a new call, any instance limits still apply – just as if the paused Sound is still playing and using voices.

See Also

sceScreamContinueSound(), sceScreamPauseAllSoundsInGroup(), sceScreamContinueAllSoundsInGroup(), sceScreamPauseGroup(), sceScreamContinueGroup(), sceScreamReverbPause(), sceScreamReverbContinue(), sceScreamPlaySoundByIndexEx(), sceScreamPlaySoundByNameEx()

sceScreamPlaySoundByIndexEx

Plays a Sound by reference to its index within a Bank.

Definition

```
uint32_t sceScreamPlaySoundByIndexEx(
    const SceScreamSFXBlock2 *bank,
    int16_t index,
    const SceScreamSoundParams *params,
    int32_t outputDest = SCE_SCREAM_SND_OUTPUT_DEST_MASTER
);
```

Arguments

bank	(Input) Pointer to a Bank that contains the Sound you wish to play, as returned by sceScreamBankLoadEx(), sceScreamBankLoadFromMemEx(), sceScreamFindLoadedBankByName(), or sceScreamGetNextLoadedBank().
index	(Input) Zero-based index of the requested Sound within the specified *bank*.
params	(Input) Pointer to an initialized SceScreamSoundParams structure containing any Sound parameter values to set from the API. See "Notes" below.
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).

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 Sound by reference to its index within a specified Bank.

The *params* argument takes a pointer to a SceScreamSoundParams structure, in which you can store initial parameter values to set on the Sound.

Notes

Setting Sound parameter values from the API overrides any corresponding settings arising from Bank contents. For further details, see "Scream Tool and API Parameter Settings" in the "System Overview" chapter of the *Scream Library Overview*.

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

See Also

sceScreamPlaySoundByNameEx(), sceScreamStopSound(), sceScreamSetGroupVoiceOutputDest()

sceScreamPlaySoundByNameEx

Plays a Sound by reference to its name within a Bank.

Definition

```
uint32_t sceScreamPlaySoundByNameEx(
    const SceScreamSFXBlock2 *bank,
    const char *name,
    const SceScreamSoundParams *params,
    int32_t outputDest = SCE_SCREAM_SND_OUTPUT_DEST_MASTER
);
```

Arguments

bank	(Input) Pointer to a Bank that contains the Sound you wish to play, as returned by sceScreamBankLoadEx(), sceScreamBankLoadFromMemEx(), sceScreamFindLoadedBankByName(), or sceScreamGetNextLoadedBank(). Can also be NULL. See "Notes" below.
name	(Input) Name of the requested Sound within the specified *bank*.
params	(Input) Pointer to an initialized SceScreamSoundParams structure containing any Sound parameter values to set from the API. See "Notes" below.
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).

Return Values

Returns the handle of the requested Sound. Returns 0 if the specified *name* is not valid or if SceScreamSoundParams.*size* specified in the *params* member is invalid.

Description

This function plays a Sound by reference to its name within a specified Bank.

The *params* argument takes a pointer to a SceScreamSoundParams structure, in which you can store initial parameter values to set on the Sound.

Notes

If *bank* is NULL, this function searches through all registered Banks, then plays the first Sound it finds having the specified *name*. If no Sound with that name is found, the function returns 0.

Setting Sound parameter values from the API overrides any corresponding settings arising from Bank contents. For further details, see "Scream Tool and API Parameter Settings" in the "System Overview" chapter of the *Scream Library Overview*.

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

See Also

sceScreamPlaySoundByIndexEx(), sceScreamStopSound(), sceScreamSetGroupVoiceOutputDest()

sceScreamSetAllSoundReg

Sets the values of all local registers specific to a Sound.

Definition

```
uint32_t sceScreamSetAllSoundReg(
    uint32_t handle,
    const int8_t *vals
);
```

Arguments

handle	(Input) Handle of the Sound for which to set all registers.
vals	(Input) An array of register values. Range of each value: -128 to 127. Length: SCE_SCREAM_SND_MAX_REGISTERS.

Return Values

Returns the specified handle if the Sound is still active. Returns 0 if the specified Sound is no longer active.

Description

This function sets the values of all local registers specific to a Sound.

Notes

To perform atomic read/modify/write operations on a Sound's local registers (that is, without interference from running scripts), use sceScreamLockAllSoundReg() and sceScreamUnlockAllSoundReg().

See Also

sceScreamGetAllSoundReg(), sceScreamGetSoundReg(), sceScreamSetSoundReg(), sceScreamLockAllSoundReg(), sceScreamUnlockAllSoundReg()

sceScreamSetLocalVariableByHash

Sets the value of a local variable.

Definition

```
bool sceScreamSetLocalVariableByHash(
    uint32_t handle,
    uint32_t nameHash,
    float val
);
```

Arguments

handle	(Input) Handle of a Sound upon which to set a local variable.
nameHash	(Input) 32-bit hash of a named local variable to set.
val	(Input) A floating-point value to set the specified local variable to.

Return Values

Returns `TRUE` if the specified Sound is active and the specified local variable exists. Returns `FALSE` otherwise.

Description

This function sets the value of a local variable associated with a Sound. You reference a local variable by its hash. You can obtain a hash for a variable name using the `sceScreamGetHashFromName()` function.

The maximum number of local variables per Sound instance is defined by the constant `SCE_SCREAM_SND_MAX_LOCAL_VARIABLES`.

You can set all a Sound's local variables at the same time by placing their values in a `SceScreamSndLocalVarData` structure and point to it as the *localVariableData* member of a `SceScreamSoundParams` structure. Call `sceScreamSetSoundParamsEx()` using this `SceScreamSoundParams` to write the variables' values.

See Also

`sceScreamGetHashFromName()`, `sceScreamGetLocalVariableByHash()`, `SceScreamSndLocalVarData`, `SceScreamSoundParams`, `sceScreamSetSoundParamsEx()`

sceScreamSetSoundInstanceLFO

Initializes an LFO or updates parameters on a running LFO.

Definition

```
uint32_t sceScreamSetSoundInstanceLFO(
    uint32_t handle,
    const SceScreamLFOParameters *params
);
```

Arguments

handle	(Input) Handle of a Sound upon which to apply LFO settings.
params	(Input) A SceScreamLFOParameters structure, appropriately initialized with LFO parameter values.

Return Values

Returns the specified Sound handle if the Sound is still active and LFO parameters were successfully applied. Returns 0 if the specified Sound is no longer active or if there was an error.

Description

This function either initializes a new LFO on a Sound instance or, if the specified LFO already exists, updates its parameters.

For more information on setting up an LFO, see "Initializing and Controlling Sound LFOs" in the "Working with Sounds" chapter of the *Scream Library Overview*.

Notes

When modifying the parameters of a running LFO (whether set up from Bank contents or from the Scream API), only changes to rate and depth can be performed seamlessly. Changes to all other parameters (for example, *shape*, *targetParam*, *startOffset*, and so on) cause the LFO to restart from the beginning of its shape.

When initializing an LFO on a Sound, you must specify the full set of LFO parameters, setting the SceScreamLFOParameters *paramMask* member to:
(SCE_SCREAM_SND_LFO_MASK_TARGET_PARAM | SCE_SCREAM_SND_LFO_MASK_SHAPE | SCE_SCREAM_SND_LFO_MASK_RATE | SCE_SCREAM_SND_LFO_MASK_DEPTH).

See Also

SceScreamLFOParameters

sceScreamSetSoundParamsEx

Sets an active Sound's parameter values.

Definition

```
uint32_t sceScreamSetSoundParamsEx(
    uint32_t handle,
    const SceScreamSoundParams *params
);
```

Arguments

handle	(Input) Handle of the Sound for which to set parameter values.
params	(Input) Pointer to a SceScreamSoundParams data structure containing the parameter values to set.

Return Values

If successful, returns the handle of the specified Sound. Returns 0 if not successful or if SceScreamSoundParams.*size* specified in the *params* member is invalid.

Description

This function sets the parameters of an actively playing Sound. Sound parameter settings made from the Scream API override any settings in Bank contents. For further details, see "Notes" below.

Notes

If the pointed to SceScreamSoundParams data structure includes SceScreamSynthParams settings, these will override all synthesizer-specific parameter settings in Bank contents, which may include settings in multiple Grains and in Child Sounds. Therefore, it is generally good practice to coordinate synthesizer parameter settings with your audio design team.

For more information on how setting these parameters interacts with the original Bank contentsScream Tool settings, see "Setting Parameter Values" in the "Working with Sounds" chapter of the *Scream Library Overview*.

See Also

sceScreamGetSoundParamsEx(), sceScreamGetSoundGainComponents()

sceScreamSetSoundReg

Sets the value of a local register.

Definition

```
uint32_t sceScreamSetSoundReg(
    uint32_t handle,
    int32_t which,
    int8_t val
);
```

Arguments

handle	(Input) Handle of the Sound on which to set a register value.
which	(Input) One-based index of the register to set. Range: 1 to SCE_SCREAM_SND_MAX_REGISTERS.
val	(Input) Value to set the register to. Range: -128 to 127.

Return Values

Returns the specified handle if the Sound is still active. Returns 0 if the specified Sound is no longer active.

Description

This function sets the value of a local (Sound-specific) register.

See Also

sceScreamGetSoundReg(), sceScreamGetAllSoundReg(), sceScreamSetAllSoundReg()

sceScreamSoundIndexGet3DDesignerParams

Retrieves asset Grain 3D parameter data by reference to a Sound index.

Definition

```
int32_t sceScreamSoundIndexGet3DDesignerParams(
    SceScreamSFXBlock2 *bank,
    int16_t index,
    uint32_t maxCount,
    uint32_t *outNum3dGrains,
    SceScreamSnd3DGrainData out3dGrainData[]
);
```

Arguments

bank	(Input) Pointer to a Bank containing a Sound to query, as returned by the sceScreamBankLoadEx(), sceScreamBankLoadFromMemEx(), sceScreamFindLoadedBankByName(), or sceScreamGetNextLoadedBank() functions.
index	(Input) Index of a Sound within the specified *bank* to query for 3D parameter data.
maxCount	(Input) The maximum number of asset Grains to query for 3D parameter data. Constrains the number of SceScreamSnd3DGrainData structures to store in the *out3DGrainData* array.
outNum3dGrains	(Output) A pointer to a uint32_t variable in which to hold the actual number of SceScreamSnd3DGrainData structures stored in the *out3DGrainData* array. Can be less than *maxCount* if there are fewer asset Grains associated with a referenced Sound.
out3dGrainData	(Output) A pointer to an array of SceScreamSnd3DGrainData structures, large enough to store at least *maxCount* records.

Return Values

If successful, returns SCE_SCREAM_SS_ERROR_OK. Otherwise, returns SCE_SCREAM_SS_ERROR_INVALID_PARAMETER.

Description

This function retrieves 3D parameter data from scripted Sounds and CCSounds, as referenced by an index within a Bank. Technically, 3D parameter data is not the property of a Sound, but actually belongs to asset Grains (*Waveform* and *Stream* Grains) contained in a Sound's script. For CCSounds, the function retrieves 3D parameter data from asset Grains contained in the scripts of Sounds controlled by a CCSound.

The function retrieves 3D parameter data from a maximum of *outNum3dGrains* associated asset Grains. If the output value of *outNum3dGrains* is less than the value you specified for *maxCount*, you can be sure that data was retrieved from all associated asset Grains. However, if *outNum3dGrains* is equal to the value you specified for *maxCount*, the retrieved 3D data may or may not represent all associated asset Grains.

Notes

Using this function, you can also determine whether a Sound can be designated as a *3D Sound*. That is, whether a Sound has an assigned distance model (or inherits a distance model from its assigned Group). The function only queries asset Grains that possess 3D settings. So if you set *maxCount* to 1,

and the output value of *outNum3dGrains* is 1, it is safe to assume that at least one Grain contained 3D settings, and therefore that the Sound can be considered a 3D Sound. Conversely, if the output value of *outNum3dGrains* is zero, you can assume the Sound is not a 3D Sound.

See Also

sceScreamSoundNameGet3DDesignerParams(), sceScreamSoundInstanceGet3DDesignerParams(), SceScreamSnd3DGrainData, sceScreamSoundInstanceGet3DComponents()

sceScreamSoundIndexHasOnStopMarker

Verifies whether an indexed Sound contains an *On Stop Marker* Grain.

Definition

```
int32_t sceScreamSoundIndexHasOnStopMarker(
    const SceScreamSFXBlock2 *bank,
    int16_t index
);
```

Arguments

bank	(Input) Pointer to a Bank that contains the referenced Sound, as returned by sceScreamBankLoadEx(), sceScreamBankLoadFromMemEx(), sceScreamFindLoadedBankByName(), or sceScreamGetNextLoadedBank().
index	(Input) Index of the Sound to examine for an *On Stop Marker* Grain.

Return Values

Returns 1 if the Sound has an *On Stop Marker* Grain. Returns 0 otherwise.

Description

This function verifies whether a Sound – referenced by its Bank index – contains an *On Stop Marker* Grain in its script. It returns 1 or 0 to indicate the presence of the marker.

See Also

sceScreamSoundNameHasOnStopMarker(), sceScreamSoundInstanceHasOnStopMarker()

sceScreamSoundInstanceGet3DComponents

Retrieves dynamic 3D components for Grains associated with a Sound instance.

Definition

```
int32_t sceScreamSoundInstanceGet3DComponents(
    uint32_t handle,
    uint32_t maxCount,
    uint32_t *outNum3dComponents,
    SceScreamSnd3DComponents out3dComponents[]
);
```

Arguments

handle	(Input) Handle of an active Sound instance to query.
maxCount	(Input) The maximum number of asset Grains to query for 3D component data. Constrains the number of SceScreamSnd3DComponents structures to store in the *out3dComponents* array.
outNum3dComponents	(Output) A pointer to a uint32_t variable in which to hold the actual number of SceScreamSnd3DComponents structures stored in the *out3dComponents* array. Can be less than *maxCount* if there are fewer asset Grains associated with a referenced Sound instance.
out3dComponents	(Output) A pointer to an array of SceScreamSnd3DComponents structures, large enough to store at least *maxCount* records.

Return Values

If successful, returns SCE_SCREAM_SS_ERROR_OK. Otherwise, returns SCE_SCREAM_SS_ERROR_INVALID_PARAMETER.

Description

This function retrieves dynamic 3D attenuation components resulting from an active distance model on a Sound instance. The function stores its results in an array of SceScreamSnd3DComponents structures, specified in the *out3dComponents* parameter, one structure per Grain queried.

Technically, 3D attenuation components are not the properties of a Sound, but actually belong to asset Grains (*Waveform* and *Stream* Grains) contained in a Sound's script. For CCSounds, the function retrieves 3D components from asset Grains contained in the scripts of Sounds controlled by a CCSound.

The function retrieves 3D attenuation components from a maximum of *maxCount* associated asset Grains. If the actual number of Grains from which 3D components were retrieved (that is, the output value of *outNum3dComponents*) is less than the value you specified for *maxCount*, you can be sure that data was retrieved from all associated asset Grains. However, if *outNum3dGrains* is equal to the value you specified for *maxCount*, the retrieved 3D data may or may not represent all associated asset Grains.

See Also

SceScreamSnd3DComponents, sceScreamSoundIndexGet3DDesignerParams(), sceScreamSoundNameGet3DDesignerParams(), sceScreamSoundInstanceGet3DDesignerParams(), SceScreamSnd3DGrainData

sceScreamSoundInstanceGet3DDesignerParams

Retrieves asset Grain 3D parameter data by reference to a Sound instance handle.

Definition

```
int32_t sceScreamSoundInstanceGet3DDesignerParams(
    uint32_t handle,
    uint32_t maxCount,
    uint32_t *outNum3dGrains,
    SceScreamSnd3DGrainData out3dGrainData[]
);
```

Arguments

handle	(Input) Handle of an active Sound instance to query.
maxCount	(Input) The maximum number of asset Grains to query for 3D parameter data. Constrains the number of SceScreamSnd3DGrainData structures to store in the *out3DGrainData* array.
outNum3dGrains	(Output) A pointer to a uint32_t variable in which to hold the actual number of SceScreamSnd3DGrainData structures stored in the *out3DGrainData* array. Can be less than *maxCount* if there are fewer asset Grains associated with a referenced Sound.
out3dGrainData	(Output) A pointer to an array of SceScreamSnd3DGrainData structures, large enough to store at least *maxCount* records.

Return Values

If successful, returns SCE_SCREAM_SS_ERROR_OK. Otherwise, returns SCE_SCREAM_SS_ERROR_INVALID_PARAMETER.

Description

This function retrieves 3D parameter data from active scripted Sounds and CCSounds, as referenced by an instance handle. Technically, 3D parameter data is not the property of a Sound, but actually belongs to asset Grains (*Waveform* and *Stream* Grains) contained in a Sound's script. For CCSounds, the function retrieves 3D parameter data from asset Grains contained in the scripts of Sounds controlled by a CCSound.

The function retrieves 3D parameter data from a maximum of *outNum3dGrains* associated asset Grains. If the output value of *outNum3dGrains* is less than the value you specified for *maxCount*, you can be sure that data was retrieved from all associated asset Grains. However, if *outNum3dGrains* is equal to the value you specified for *maxCount*, the retrieved 3D data may or may not represent all associated asset Grains.

Notes

Using this function, you can also determine whether a Sound can be designated as *3D Sound*. That is, whether a Sound has an assigned distance model (or inherits a distance model from its assigned Group). The function only queries asset Grains that possess 3D settings. So if you set *maxCount* to 1, and the output value of *outNum3dGrains* is 1, it is safe to assume that at least one Grain contained 3D settings, and therefore that the Sound can be considered a 3D Sound. Conversely, if the output value of *outNum3dGrains* is zero, you can assume the Sound is not a 3D Sound.

Unlike sceScreamSoundIndexGet3DDesignerParams() and sceScreamSoundNameGet3DDesignerParams(), which operate on a Sound as stored in Bank contents, this function operates on a running instance of a Sound.

See Also

sceScreamSoundIndexGet3DDesignerParams(), sceScreamSoundNameGet3DDesignerParams(), SceScreamSnd3DGrainData, sceScreamSoundInstanceGet3DComponents()

sceScreamSoundInstanceHasOnStopMarker

Verifies whether a Sound instance contains an *On Stop Marker* Grain.

Definition

```
int32_t sceScreamSoundInstanceHasOnStopMarker(
    uint32_t handle
);
```

Arguments

handle	(Input) Handle of the Sound to examine for an *On Stop Marker* Grain.

Return Values

Returns 1 if the Sound contains an *On Stop Marker* Grain. Returns 0 otherwise.

Description

This function verifies whether a Sound instance contains an *On Stop Marker* Grain. It returns 1 or 0 to indicate the presence of the marker.

Notes

Unlike sceScreamSoundIndexHasOnStopMarker() and sceScreamSoundNameHasOnStopMarker(), which operate on a Sound as stored in Bank contents, this function operates on a running instance of a Sound.

See Also

sceScreamSoundIndexHasOnStopMarker(), sceScreamSoundNameHasOnStopMarker()

sceScreamSoundIsStillPlaying

Verifies whether a Sound instance is still playing.

Definition

```
uint32_t sceScreamSoundIsStillPlaying(
    uint32_t handle
);
```

Arguments

handle	(Input) Sound handle, as returned by sceScreamPlaySoundByIndexEx() or sceScreamPlaySoundByNameEx().

Return Values

Returns the passed in Sound handle if the Sound is still playing. Returns 0 if the Sound has stopped.

Description

This status function verifies whether a Sound is still playing.

See Also

sceScreamPlaySoundByIndexEx(), sceScreamPlaySoundByNameEx()

sceScreamSoundNameGet3DDesignerParams

Retrieves asset Grain 3D parameter data by reference to a Sound name.

Definition

```
int32_t sceScreamSoundNameGet3DDesignerParams(
    SceScreamSFXBlock2 *bank,
    const char *name,
    uint32_t maxCount,
    uint32_t *outNum3dGrains,
    SceScreamSnd3DGrainData out3dGrainData[]
);
```

Arguments

bank	(Input) Pointer to a Bank containing a Sound to query, as returned by the sceScreamBankLoadEx(), sceScreamBankLoadFromMemEx(), sceScreamFindLoadedBankByName(), or sceScreamGetNextLoadedBank() functions.
name	(Input) Name of a Sound within the specified *bank* to query for 3D parameter data.
maxCount	(Input) The maximum number of asset Grains to query for 3D parameter data. Constrains the number of SceScreamSnd3DGrainData structures to store in the *out3DGrainData* array.
outNum3dGrains	(Output) A pointer to a uint32_t variable in which to hold the actual number of SceScreamSnd3DGrainData structures stored in the *out3DGrainData* array. Can be less than *maxCount* if there are fewer asset Grains associated with a referenced Sound.
out3dGrainData	(Output) A pointer to an array of SceScreamSnd3DGrainData structures, large enough to store at least *maxCount* records.

Return Values

If successful, returns SCE_SCREAM_SS_ERROR_OK. Otherwise, returns SCE_SCREAM_SS_ERROR_INVALID_PARAMETER.

Description

This function retrieves 3D parameter data from scripted Sounds and CCSounds, as referenced by a name within a Bank. Technically, 3D parameter data is not the property of a Sound, but actually belongs to asset Grains (*Waveform* and *Stream* Grains) contained in a Sound's script. For CCSounds, the function retrieves 3D parameter data from asset Grains contained in the scripts of Sounds controlled by a CCSound.

The function retrieves 3D parameter data from a maximum of *outNum3dGrains* associated asset Grains. If the output value of *outNum3dGrains* is less than the value you specified for *maxCount*, you can be sure that data was retrieved from all associated asset Grains. However, if *outNum3dGrains* is equal to the value you specified for *maxCount*, the retrieved 3D data may or may not represent all associated asset Grains.

Notes

Using this function, you can also determine whether a Sound can be designated as *3D Sound*. That is, whether a Sound has an assigned distance model (or inherits a distance model from its assigned Group). The function only queries asset Grains that possess 3D settings. So if you set *maxCount* to 1,

and the output value of *outNum3dGrains* is 1, it is safe to assume that at least one Grain contained 3D settings, and therefore that the Sound can be considered a 3D Sound. Conversely, if the output value of *outNum3dGrains* is zero, you can assume the Sound is not a 3D Sound.

See Also

sceScreamSoundIndexGet3DDesignerParams(), sceScreamSoundInstanceGet3DDesignerParams(), SceScreamSnd3DGrainData, sceScreamSoundInstanceGet3DComponents()

sceScreamSoundNameHasOnStopMarker

Verifies whether a named Sound contains an *On Stop Marker* Grain.

Definition

```
int32_t sceScreamSoundNameHasOnStopMarker(
    const SceScreamSFXBlock2 *bank,
    const char *name
);
```

Arguments

bank	(Input) Pointer to a Bank that contains the referenced Sound, as returned by sceScreamBankLoadEx(), sceScreamBankLoadFromMemEx(), sceScreamFindLoadedBankByName(), or sceScreamGetNextLoadedBank().
name	(Input) Name of the Sound to examine for an *On Stop Marker* Grain.

Return Values

Returns 1 if the Sound contains an *On Stop Marker* Grain. Returns 0 otherwise.

Description

This function verifies whether a Sound – referenced by name within its Bank – contains an *On Stop Marker* Grain. It returns 1 or 0 to indicate the presence of the marker.

See Also

sceScreamSoundIndexHasOnStopMarker(), sceScreamSoundInstanceHasOnStopMarker()

sceScreamStopSound

Stops a Sound.

Definition

```
uint32_t sceScreamStopSound(
    uint32_t handle,
    int32_t behavior
);
```

Arguments

handle	(Input) Handle of the Sound to stop, as returned by sceScreamPlaySoundByIndexEx() or sceScreamPlaySoundByNameEx().
behavior	(Input) A choice of two stop behaviors: SCE_SCREAM_SND_STOP_BEHAVIOR_KEYOFF or SCE_SCREAM_SND_STOP_BEHAVIOR_SILENCE.

Return Values

Returns the specified handle if the Sound is stopped. Returns 0 if the specified Sound is no longer active.

Description

This function stops the specified Sound.

The *behavior* parameter provides a choice of two stop behaviors:

- SCE_SCREAM_SND_STOP_BEHAVIOR_KEYOFF: Performs a graceful stop, triggering any *On Stop Marker* grain events, and issuing key-off messages to active voices with ADSR Release settings.
- SCE_SCREAM_SND_STOP_BEHAVIOR_SILENCE: Performs an instantaneous stop.

This function is primarily used for looping Sounds; one-shot Sounds stop themselves.

See Also

sceScreamStopAllSounds(), sceScreamStopAllSoundsInGroup(), sceScreamStopAllSoundsInBank(), sceScreamPlaySoundByIndexEx(), sceScreamPlaySoundByNameEx()

sceScreamUnlockAllSoundReg

Sets a Sound's local register values and unlocks them.

Definition

```
uint32_t sceScreamUnlockAllSoundReg(
    uint32_t handle,
    const int8_t *vals
);
```

Arguments

handle	(Input) Handle of the Sound for which to set all local register values.
vals	(Input) An array (of size SCE_SCREAM_SND_MAX_REGISTERS) containing the local register values to set. Range of each value: -128 to 127.

Return Values

Returns the specified handle if the Sound is still active. Returns 0 if the specified Sound is no longer active.

Description

This function sets new values for a Sound's local registers after they have been locked with a call to sceScreamLockAllSoundReg(). After setting the local register values, the function unlocks them, re-exposing them to script modification.

Notes

WARNING: This function must be used in conjunction with a preceding call to sceScreamLockAllSoundReg(). Because sceScreamLockAllSoundReg() effectively locks the entire underlying rendering synthesizer, it is critical that it is followed by a matching call to this function to unlock the rendering synthesizer. Processing performed between these two calls should be kept to an absolute minimum!

If you do not need an atomic read/modify/write capability on a Sound's local registers (that is, without interference from running scripts), use sceScreamGetAllSoundReg() and sceScreamSetAllSoundReg() instead.

See Also

sceScreamLockAllSoundReg(), sceScreamGetAllSoundReg(), sceScreamSetAllSoundReg(), sceScreamSetSoundReg()

Reverb Functions

Summary

Reverb functions set stock and custom presets, set properties, and pause/continue I3DL2 reverb instances.

Function	Description
sceScreamReverbContinue	Continues a (paused) reverb instance.
sceScreamReverbGetHandleByBuss	Returns the handle of a reverb instance currently assigned to a specified buss.
sceScreamReverbPause	Pauses a reverb instance.
sceScreamReverbSetAllProperties	Sets the properties of a reverb instance.
sceScreamReverbSetCustomPreset	Sets a custom preset, referenced by index, on a specified reverb instance.
sceScreamReverbSetCustomPresetByName	Sets a custom preset, referenced by name, on a specified reverb instance.
sceScreamReverbSetDirectPathOutputDest	Sets the output destination of a reverb instance.
sceScreamReverbSetStockPreset	Sets a stock preset on a specified reverb instance.
sceScreamReverbSetVolumePolar	Sets reverb instance volume and pan properties.

sceScreamReverbContinue

Continues a (paused) reverb instance.

Definition

```
int32_t sceScreamReverbContinue(
    SceScreamReverbHandle handle
);
```

Arguments

handle	(Input) Handle of the reverb instance to continue.

Return Values

Returns SCE_SCREAM_SS_ERROR_OK.

Description

This function continues a reverb instance that has been paused using the sceScreamReverbPause() function.

Notes

Continuing an instance that is not paused has no effect.

See Also

sceScreamReverbPause(), sceScreamPauseGroup(), sceScreamContinueGroup(), sceScreamPauseAllSoundsInGroup(), sceScreamContinueAllSoundsInGroup(), sceScreamPauseSound(), sceScreamContinueSound()

sceScreamReverbGetHandleByBuss

Returns the handle of a reverb instance currently assigned to a specified buss.

Definition

```
SceScreamReverbHandle sceScreamReverbGetHandleByBuss(
    uint32_t buss
);
```

Arguments

buss	(Input) Zero-based index of an auxiliary send buss from which to retrieve the reverb instance handle. Range: 0 to (SCE_SCREAM_NUM_AUX_BUSSES – 1).

Return Values

Returns the handle of a reverb instance assigned to the specified buss. If the specified buss is unoccupied, the function returns zero.

Description

This function returns the handle of a reverb instance currently assigned to a specified auxiliary send buss.

Notes

To route an external voice to a Scream-managed reverb instance, with respect to synthesizer, cast the return value from a SceScreamReverbHandle to a `SceNgsHVoice` type.

sceScreamReverbPause

Pauses a reverb instance.

Definition

```
int32_t sceScreamReverbPause(
    SceScreamReverbHandle handle
);
```

Arguments

handle	(Input) Handle of the reverb instance to pause.

Return Values

Returns SCE_SCREAM_SS_ERROR_OK.

Description

This function suspends processing of a reverb instance. This means that active reverb tails are preserved, and are resumed immediately upon calling sceScreamReverbContinue().

Notes

Pausing a reverb instance that is already paused has no effect.

See Also

sceScreamReverbContinue(), sceScreamPauseGroup(), sceScreamContinueGroup(), sceScreamPauseAllSoundsInGroup(), sceScreamContinueAllSoundsInGroup(), sceScreamPauseSound(), sceScreamContinueSound()

sceScreamReverbSetAllProperties

Sets the properties of a reverb instance.

Definition

```
int32_t sceScreamReverbSetAllProperties(
    SceScreamReverbHandle handle,
    const SceScreamSndReverbProps *properties
);
```

Arguments

handle	(Input) Handle of a reverb instance upon which to set properties.
properties	(Input) A SceScreamSndReverbProps structure, storing reverb property values.

Return Values

Returns SCE_SCREAM_SS_ERROR_OK if the operation was successful, otherwise returns SCE_SCREAM_SS_ERROR_INVALID_PARAMETER if an invalid parameter value was detected.

Description

This function sets the properties (that is, parameter values) of a reverb instance. You store the desired reverb property values in a SceScreamSndReverbProps structure, and reference it in the *properties* parameter.

A simpler way to set reverb properties is to set a stock or custom preset using the sceScreamReverbSetStockPreset() or sceScreamReverbSetCustomPreset() functions respectively.

This function may be useful if you are working with a custom data format for storing reverb settings or programmatically generating reverb parameter values based on game geometries.

See Also

SceScreamSndReverbProps, sceScreamReverbSetStockPreset(), sceScreamReverbSetCustomPreset()

sceScreamReverbSetCustomPreset

Sets a custom preset, referenced by index, on a specified reverb instance.

Definition

```
int32_t sceScreamReverbSetCustomPreset(
    SceScreamReverbHandle handle,
    SceScreamIniHandle iniFile,
    uint32_t presetIndex
);
```

Arguments

handle	(Input) Handle of a reverb instance upon which to set a custom preset.
iniFile	(Input) Handle of a presets file that contains the desired custom preset.
presetIndex	(Input) Zero-based index of the desired custom preset within the specified presets file. Range: 0 to (*numPresets* – 1), where *numPresets* represents the number of presets in the specified presets file, and can be determined by calling sceScreamPresetFileGetPresetCount(), passing the appropriate SceScreamIniHandle.

Return Values

Returns SCE_SCREAM_SS_ERROR_OK if the operation was successful, otherwise returns SCE_SCREAM_SS_ERROR_INVALID_PARAMETER if an invalid parameter value was detected.

Description

This function sets a custom preset on a specified reverb instance. You load custom presets files (in INI format) into Scream using the sceScreamPresetFileLoad() function, which returns a SceScreamIniHandle. You specify the desired preset by its index within the presets file.

Notes

To set a preset by name, use the sceScreamReverbSetCustomPresetByName() function.

See Also

sceScreamReverbSetCustomPresetByName(), sceScreamReverbSetStockPreset(), sceScreamPresetFileLoad(), sceScreamPresetFileGetPresetCount()

sceScreamReverbSetCustomPresetByName

Sets a custom preset, referenced by name, on a specified reverb instance.

Definition

```
int32_t sceScreamReverbSetCustomPresetByName(
    SceScreamReverbHandle handle,
    SceScreamIniHandle iniFile,
    const char *presetName
);
```

Arguments

handle	(Input) Handle of a reverb instance upon which to set the custom preset.
iniFile	(Input) Handle of a presets file that contains the desired custom preset.
presetName	(Input) A string matching one of the preset names within the specified presets file.

Return Values

Returns SCE_SCREAM_SS_ERROR_OK if the operation was successful, otherwise returns SCE_SCREAM_SS_ERROR_INVALID_PARAMETER if an invalid parameter value was detected.

Description

This function sets a custom preset on a specified reverb instance. You load custom presets files (in INI format) into Scream using the sceScreamPresetFileLoad() function. The sceScreamReverbSetCustomPresetByName() function allows you specify the desired preset by name – rather than by its index – within the presets file.

Notes

To specify a preset by its index within the file, use the sceScreamReverbSetCustomPreset() function.

See Also

sceScreamReverbSetCustomPreset(), sceScreamReverbSetStockPreset(), sceScreamPresetFileLoad(), sceScreamPresetFileGetPresetCount(), sceScreamPremasterSubmixSetCustomPresetByName()

sceScreamReverbSetDirectPathOutputDest

Sets the output destination of a reverb instance.

Definition

```
int32_t sceScreamReverbSetDirectPathOutputDest(
    SceScreamReverbHandle handle,
    int32_t outputDest
);
```

Arguments

handle	(Input) Handle of a reverb instance for which to set the output destination.
outputDest	(Input) Index of an output destination. 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). To specify the master buss, use SCE_SCREAM_SND_OUTPUT_DEST_MASTER. See "Notes" below.

Return Values

Returns SCE_SCREAM_SS_ERROR_OK if the operation was successful, otherwise returns SCE_SCREAM_SS_ERROR_INVALID_PARAMETER if an invalid parameter value was detected.

Description

This function sets the output destination of a reverb instance. Without calling this function, reverb instance output destination defaults to the master buss. However, using this function you can optionally route reverb output to one of the pre-master submix busses.

Notes

Pre-master submix busses must be allocated at initialization time using the *numPremasterCompSubmixes* and *numPremasterScCompSubmixes* members of the SceScreamSystemParams structure. Make sure that you do not set a pre-master submix output destination in *outputDest* that has not been allocated. For further details, see "Setting NGS/NGS2 Pre-Master Submix Indices" in the "Working with Pre-Master and Master Signal Processors" chapter of the *Scream Library Overview*.

See Also

sceScreamSetGroupVoiceOutputDest()

sceScreamReverbSetStockPreset

Sets a stock preset on a specified reverb instance.

Definition

```
int32_t sceScreamReverbSetStockPreset(
    SceScreamReverbHandle handle,
    uint32_t presetIndex
);
```

Arguments

handle	(Input) Handle of a reverb instance upon which to set the stock preset.
presetIndex	(Input) Index of the desired stock preset. One of the SceScreamI3DL2StockPresets constants.

Return Values

Returns SCE_SCREAM_SS_ERROR_OK if the operation was successful, otherwise returns SCE_SCREAM_SS_ERROR_INVALID_PARAMETER if an invalid parameter value was detected.

Description

This function sets one of the stock presets on a specified reverb instance. For a description of the reverb stock presets, see SceScreamI3DL2StockPresets.

See Also

sceScreamReverbSetCustomPresetByName(), sceScreamReverbSetCustomPreset()

sceScreamReverbSetVolumePolar

Sets reverb instance volume and pan properties.

Definition

```
int32_t sceScreamReverbSetVolumePolar(
    SceScreamReverbHandle handle,
    float gain,
    float lfeGain,
    uint32_t azimuth,
    uint32_t focus,
    uint32_t mode,
    uint32_t excludeTargets
);
```

Arguments

handle	(Input) Handle of the reverb instance for which to set volume and pan properties.
gain	(Input) Overall output gain level of the reverb instance. This parameter scales all output channels except the LFE channel. Range: SCE_SCREAM_SND_MIN_GAIN to SCE_SCREAM_SND_MAX_GAIN.
lfeGain	(Input) Gain on the LFE channel. **Note**: this parameter is ignored when running Scream on the NGS synthesizer.
azimuth	(Input) Panning azimuth of reverb instance, expressed in degrees clockwise. Range: 0 to 359. A value of 0 specifies that reverb is straight ahead; 90 specifies directly to the right; 180 specifies behind, and so on.
focus	(Input) Width of panning focus of reverb instance, expressed in degrees. Range: 0 to 360. A value of 0 specifies a point source; 360 specifies reverb coming from all directions. **Note:** Setting *focus* to 360 makes *azimuth* irrelevant.
mode	(Input) **Note**: This parameter is ignored when running Scream on the NGS synthesizer.
excludeTargets	(Input) **Note**: This parameter is ignored when running Scream on the NGS synthesizer.

Return Values

Returns SCE_SCREAM_SS_ERROR_OK if the operation was successful, otherwise returns SCE_SCREAM_SS_ERROR_INVALID_PARAMETER if an invalid parameter value was detected.

Description

This function sets output volume and panning properties collectively for all channels of a reverb instance.

For further information about reverb azimuth and focus, see the "Setting Reverb Instance Output Gain and Panning" in the "Working with the I3DL2 Reverb" chapter of the *Scream Library Overview*.

Auxiliary Buss Functions

Summary

Auxiliary buss functions allow you to manipulate auxiliary busses.

Function	Description
sceScreamSetAuxBussOutputDest	Sets an auxiliary buss output destination.

sceScreamSetAuxBussOutputDest

Sets an auxiliary buss output destination.

Definition

```
int32_t sceScreamSetAuxBussOutputDest(
    uint32_t auxBuss,
    int32_t outputDest
);
```

Arguments

auxBuss	(Input) Zero-based index of an auxiliary buss for which to set the output destination.
outputDest	(Input) Index of an output destination to set. For the master output, use SCE_SCREAM_SND_OUTPUT_DEST_MASTER. To specify an allocated pre-master submix buss, use the number of the desired submix, indexing from zero. Range: SCE_SCREAM_SND_OUTPUT_DEST_PREMASTER_0 to (SCE_SCREAM_SND_MAX_PREMASTER_SUBMIXES – 1). See "Notes" below.

Return Values

If successful, returns SCE_SCREAM_SS_ERROR_OK. Otherwise, returns SCE_SCREAM_SS_ERROR_INVALID_PARAMETER.

Description

This function sets the output destination for an auxiliary buss. Without calling this function, auxiliary buss output destination defaults to the master buss. However, using this function you can optionally route auxiliary buss output to one of the pre-master submix busses.

Notes

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

See Also

sceScreamSetGroupVoiceOutputDest()

Presets File (INI) Functions

Summary

Preset file functions load and unload presets (INI) files, and retrieve information from them.

Function	Description
sceScreamPresetFileGetPresetCount	Retrieves the count of individual presets contained in a presets file.
sceScreamPresetFileGetPresetName	Retrieves the name of a preset contained in a presets file.
sceScreamPresetFileLoad	Loads a presets file from disk.
sceScreamPresetFileLoadFromMem	Loads a presets file from memory.
sceScreamPresetFileUnload	Unloads a presets file.

sceScreamPresetFileGetPresetCount

Retrieves the count of individual presets contained in a presets file.

Definition

```
uint32_t sceScreamPresetFileGetPresetCount(
    SceScreamIniHandle handle
);
```

Arguments

handle	(Input) Handle of the presets file from which to obtain a count of presets. Returned by the sceScreamPresetFileLoad() or sceScreamPresetFileLoadFromMem() functions.

Return Values

Returns the total count of presets contained in the specified presets file.

Description

This function retrieves the total count of presets contained in a presets INI file.

Notes

Use this function to determine the maximum value plus one for the *presetIndex* parameter in the sceScreamReverbSetCustomPreset() and sceScreamPremasterSubmixSetCustomPreset() functions. Because these functions' *presetIndex* parameter expects a zero-based index of presets, subtract one (1) from the value returned by this function to get the maximum index.

See Also

sceScreamReverbSetCustomPreset(), sceScreamPremasterSubmixSetCustomPreset()

sceScreamPresetFileGetPresetName

Retrieves the name of a preset contained in a presets file.

Definition

```
int32_t sceScreamPresetFileGetPresetName(
    SceScreamIniHandle handle,
    uint32_t presetIndex,
    char *buffer,
    uint32_t bufferLength
);
```

Arguments

handle	(Input) Handle of the presets file from which to retrieve the name of a preset. Returned by the sceScreamPresetFileLoad() or sceScreamPresetFileLoadFromMem() functions.
presetIndex	(Input) Zero-based index of a preset from which to retrieve the name.
buffer	(Input) Pointer to a character buffer in which to store the preset name. Must be at least the size specified in *bufferLength*.
bufferLength	(Input) Size, in bytes, of the buffer pointed to by *buffer*.

Return Values

Returns SCE_SCREAM_SS_ERROR_OK if the operation was successful, otherwise returns SCE_SCREAM_SS_ERROR_INVALID_PARAMETER if an invalid parameter value was detected.

Description

This function retrieves the name of a preset, specified by its index within a presets file.

See Also

sceScreamReverbSetCustomPresetByName(),
sceScreamPremasterSubmixSetCustomPresetByName(),
sceScreamPresetFileGetPresetCount()

sceScreamPresetFileLoad

Loads a presets file from disk.

Definition

```
SceScreamIniHandle sceScreamPresetFileLoad(
    const char *name
);
```

Arguments

name	(Input) Path to a presets file to load from disk.

Return Values

Returns a SceScreamIniHandle presets file handle.

Description

This function loads a presets file in INI format from disk, and returns a handle to it.

Notes

Whenever memory is requested for a presets file, the system passes the constant SCE_SCREAM_SND_MEM_USE_PRESETS to your custom SceScreamExternSndMemAlloc() *use* parameter.

See Also

sceScreamPresetFileLoadFromMem(), sceScreamPresetFileUnload()

sceScreamPresetFileLoadFromMem

Loads a presets file from memory.

Definition

```
SceScreamIniHandle sceScreamPresetFileLoadFromMem(
    const void *loc,
    uint32_t size
);
```

Arguments

loc	(Input) Location of a presets file in memory.
size	(Input) Size, in bytes, of the memory-resident presets file pointed to in *loc*.

Return Values

Returns a SceScreamIniHandle presets file handle.

Description

This function loads a presets file in INI format from memory, and returns a handle to it.

Notes

The preset file must remain resident in memory until released with a call to the sceScreamPresetFileUnload() function.

See Also

sceScreamPresetFileLoad(), sceScreamPresetFileUnload()

sceScreamPresetFileUnload

Unloads a presets file.

Definition

```
int32_t sceScreamPresetFileUnload(
    SceScreamIniHandle handle
);
```

Arguments

handle	(Input) Handle of the presets file to unload. Returned from the sceScreamPresetFileLoad() or sceScreamPresetFileLoadFromMem() functions.

Return Values

Returns SCE_SCREAM_SS_ERROR_OK if the operation was successful, otherwise returns SCE_SCREAM_SS_ERROR_INVALID_PARAMETER if an invalid parameter value was detected.

Description

This function unloads a presets file, freeing any associated memory.

Notes

This function should be called regardless of the method used to load the presets file – from disk or from memory. There is no asynchronous presets file unload function.

See Also

sceScreamPresetFileLoad(), sceScreamPresetFileLoadFromMem()

Buss Configuration Functions

Summary

Buss configuration functions allow you to set and query buss presets.

Function	Description
sceScreamApplyBussPreset	Applies a preset to a buss effect.
sceScreamGetBussPresetCount	Retrieves the total count of buss presets.
sceScreamGetBussPresetName	Retrieves the name of a buss preset.
sceScreamGetBussPresetType	Retrieves the buss type associated with a named buss preset.

sceScreamApplyBussPreset

Applies a preset to a buss effect.

Definition

```
bool sceScreamApplyBussPreset(
    const char *name,
    int32_t bussIndex
);
```

Arguments

name	(Input) Name of a preset to apply.
bussIndex	(Input) Index of a premaster submix buss or auxiliary buss upon which to apply the preset. Defaults to -1, in which case the buss index value is specified by the preset itself. Not applicable to presets for which there is only one corresponding buss, such as the master buss. See "Notes" below.

Return Values

Returns `TRUE` if the preset was successfully applied, otherwise returns `FALSE`.

Description

This function applies a preset to an effect. The preset, specified by name, must be contained in a buss configuration file, loaded into Scream at initialization time, specified in the `SceScreamPlatformInitEx2.pBussConfigFile` member. The index of the target buss upon which to apply the preset can be specified either by preset data or by the programmer.

Notes

The buss index specified in the *bussIndex* parameter, if other than the default, overrides any preset data-specified buss index.

On the NGS synthesizer, the number of auxiliary busses is `SCE_SCREAM_NUM_AUX_BUSSES` (3), and auxiliary buss indices range from 0 to 2.

Pre-master submix busses must be allocated at initialization time using the *numPremasterCompSubmixes* and *numPremasterScCompSubmixes* members of the `SceScreamSystemParams` structure. Make sure that you do not apply a buss preset to a premaster submix buss that has not been allocated.

See Also

`SceScreamPlatformInitEx2.pBussConfigFile`

sceScreamGetBussPresetCount

Retrieves the total count of buss presets.

Definition

```
bool sceScreamGetBussPresetCount(
    uint32_t *count
);
```

Arguments

count	(Output) Pointer to a `uint32_t` variable in which to receive the buss preset count.

Return Values

Returns `TRUE` if successful, otherwise returns `FALSE`.

Description

This function retrieves the total count of loaded effect presets. Presets must be contained in a buss presets file, loaded into Scream at initialization time, specified in the `SceScreamPlatformInitEx2.pBussConfigFile` member.

See Also

`SceScreamPlatformInitEx2.pBussConfigFile`

sceScreamGetBussPresetName

Retrieves the name of a buss preset.

Definition

```
bool sceScreamGetBussPresetName(
    uint32_t presetIndex,
    const char **presetName
);
```

Arguments

presetIndex	(Input) Index of a preset for which to retrieve the name.
presetName	(Output) Pointer to a `const char` pointer in which to receive the name of the specified preset.

Return Values

Returns `TRUE` if successful, otherwise returns `FALSE`.

Description

This function retrieves the name of an effect preset specified by index. Presets must be contained in an effect presets file, loaded into Scream at initialization time, specified in the `SceScreamPlatformInitEx2.pBussConfigFile` member.

See Also

`SceScreamPlatformInitEx2.pBussConfigFile`

sceScreamGetBussPresetType

Retrieves the buss type associated with a named buss preset.

Definition

```
int32_t sceScreamGetBussPresetType(
    const char *name
);
```

Arguments

name	(Input) Name of a preset to query for its buss type.

Return Values

Returns one of the Buss Types.

Description

This function retrieves the buss type associated with a named effect preset. The preset, specified by name, must be contained in a buss presets file, loaded into Scream at initialization time, specified in the SceScreamPlatformInitEx2.*pBussConfigFile* member.

See Also

SceScreamPlatformInitEx2.*pBussConfigFile*

Document serial number: 000004892117

Group Mix Functions

Summary

Group mix functions allow you to activate, deactivate, and query mix snapshots. And to set the mixer base level.

Function	Description
sceScreamActivateMixSnapshot	Activates a mix snapshot.
sceScreamDeactivateAllMixSnapshots	Deactivates all active mix snapshots.
sceScreamDeactivateMixSnapshot	Deactivates a mix snapshot.
sceScreamGetActiveMixSnapshotCount	Retrieves a count of the number of active mix snapshots.
sceScreamGetActiveMixSnapshotNames	Retrieves a priority ordered list of active mix snapshot names.
sceScreamGetMixSnapshotCount	Retrieves a count of the total number of mix snapshots.
sceScreamGetMixSnapshotName	Retrieves the name of a mix snapshot, specified by index.
sceScreamGetMixSnapshotPriority	Retrieves the priority setting for a mix snapshot.
sceScreamIsMixSnapshotActive	Queries whether a mix snapshot is active.
sceScreamSetGroupMixerBaseLevel	Sets the group mixer base level.

sceScreamActivateMixSnapshot

Activates a mix snapshot.

Definition

```
bool sceScreamActivateMixSnapshot(
    const char *name,
    float mixScalar,
    float fadeTimeOverride
);
```

Arguments

name	(Input) Name of a snapshot to activate.
mixScalar	(Input) A normalized percentage of the snapshot to apply. Defaults to 1.0 (or 100%) to use the designer's setting. Setting this parameter to lower than 1.0 reduces the level-setting impact of a snapshot.
fadeTimeOverride	(Input) A programmer override on the transition-in time of any mix changes that occur as a result of activating a snapshot. Expressed in seconds. Defaults to -1.0 (or no override), allowing the snapshot's transition-in time to be as defined by the designer.

Return Values

Returns `TRUE` if the snapshot is successfully activated, otherwise returns `FALSE`.

Description

This function activates a mix snapshot. You specify the name of a snapshot to activate, as defined in a loaded group mixer file. You pre-load a group mixer file into memory, and then provide the corresponding memory pointer as a value for the `SceScreamPlatformInitEx2` structure's *pGroupMixerFile* member when initializing Scream.

The mix snapshots mechanism supports activation of multiple snapshots at the same time. You can set an upper limit for the number of simultaneously active snapshots at initialization time using the `SceScreamPlatformInitEx2` *maxActiveSnapshots* member.

Snapshots contain level settings for one or more Groups, which are applied using either of two modes: Adjustment or Absolute. In Absolute mode, Group volume level is set by a single snapshot. If multiple snapshots contain Absolute settings for a Group, the snapshot with the highest priority prevails. In Adjustment mode, Group volume level may be set by a combination of multiple snapshots. If multiple snapshots contain Adjustment settings for a Group, their settings add together to produce a combined snapshot level for the Group.

For further details, see "Understanding Mix Snapshots" in the "Working with Mix Snapshots" of the *Scream Library Overview*.

See Also

`sceScreamDeactivateMixSnapshot()`, `sceScreamDeactivateAllMixSnapshots()`, `SceScreamPlatformInitEx2.pGroupMixerFile`, `sceScreamSetMasterVolume()`

sceScreamDeactivateAllMixSnapshots

Deactivates all active mix snapshots.

Definition

```
int32_t sceScreamDeactivateAllMixSnapshots(
    float fadeTimeSeconds
);
```

Arguments

fadeTimeSeconds	(Input) Transition-out time, over which all Group volumes return to foundation mix settings, that is, initial Group levels combined with the mixer base level. Expressed in seconds.

Return Values

Returns SCE_SCREAM_SS_ERROR_OK if the operation was successful, otherwise returns SCE_SCREAM_SS_ERROR_INVALID_PARAMETER if an invalid parameter value was detected.

Description

This function deactivates all active mix snapshots.

See Also

sceScreamActivateMixSnapshot(), sceScreamDeactivateMixSnapshot(), sceScreamSetMasterVolume()

sceScreamDeactivateMixSnapshot

Deactivates a mix snapshot.

Definition

```
bool sceScreamDeactivateMixSnapshot(
    const char *name,
    float fadeTimeOverride
);
```

Arguments

name	(Input) Name of a snapshot to deactivate.
fadeTimeOverride	(Input) A programmer override on the transition-out time of any mix changes that occur as a result of deactivating a snapshot. Expressed in seconds. Defaults to -1.0 (or no override), allowing the snapshot's transition-out time to be as defined by the designer.

Return Values

Returns `TRUE` if the snapshot is successfully deactivated, otherwise returns `FALSE`.

Description

This function deactivates a mix snapshot, specified by name.

See Also

`sceScreamActivateMixSnapshot()`, `sceScreamDeactivateAllMixSnapshots()`, `sceScreamSetMasterVolume()`

sceScreamGetActiveMixSnapshotCount

Retrieves a count of the number of active mix snapshots.

Definition

```
uint32_t sceScreamGetActiveMixSnapshotCount(void);
```

Return Values

Returns a count of the number of active mix snapshots. The returned value is in the range 0 to SceScreamPlatformInitEx2.*maxActiveSnapshots*.

Description

This function retrieves a count of the number of active mix snapshots.

See Also

sceScreamIsMixSnapshotActive(), sceScreamGetMixSnapshotPriority()

Document serial number: 000004892117

sceScreamGetActiveMixSnapshotNames

Retrieves a priority ordered list of active mix snapshot names.

Definition

```
uint32_t sceScreamGetActiveMixSnapshotNames(
    char **names,
    uint32_t maxCount
);
```

Arguments

names	(Output) An array of `char` pointers in which to receive snapshot names.
maxCount	(Input) Maximum number of active snapshot names to retrieve. Range: 0 to SceScreamPlatformInitEx2.*maxActiveSnapshots*.

Return Values

Returns the number of active mix snapshot names retrieved. The returned value is in the range: 0 to *maxCount*.

Description

This function retrieves a priority ordered list of active mix snapshot names.

See Also

sceScreamGetActiveMixSnapshotCount()

sceScreamGetMixSnapshotCount

Retrieves a count of the total number of mix snapshots.

Definition

```
uint32_t sceScreamGetMixSnapshotCount(void);
```

Return Values

Returns a count of the number of mix snapshots contained in your group mixer file.

Description

This function retrieves a count of the total number of mix snapshots contained in a group mixer file with which Scream was initialized.

You can use the returned count value to determine the upper limit of snapshot indices when retrieving snapshot names using the `sceScreamGetMixSnapshotName()` function.

See Also

`sceScreamGetMixSnapshotName()`, `SceScreamPlatformInitEx2.pGroupMixerFile`, `sceScreamGetActiveMixSnapshotCount()`

Document serial number: 000004892117

sceScreamGetMixSnapshotName

Retrieves the name of a mix snapshot, specified by index.

Definition

```
int32_t sceScreamGetMixSnapshotName(
    uint32_t snapshotIndex,
    char *snapshotName,
    uint32_t maxLength
);
```

Arguments

snapshotIndex	(Input) The index of a snapshot for which to retrieve the name. Range 0 to (sceScreamGetMixSnapshotCount() – 1).
snapshotName	(Output) A char array into which the name of the specified mix snapshot is copied.
maxLength	(Input) Maximum number of characters to copy into the *snapshotName* array.

Return Values

If successful, returns SCE_SCREAM_SS_ERROR_OK. Otherwise, returns SCE_SCREAM_SS_ERROR_INVALID_PARAMETER.

Description

This function retrieves the name of a mix snapshot. Mix snapshots are contained in a group mixer file, which must be loaded into Scream at initialization time. You identify a mix snapshot to query by its index within your group mixer file. You can use the sceScreamGetMixSnapshotCount() function to retrieve a count the number of mix snapshots contained in your group mixer file. The *snapshotIndex* parameter takes a zero-based index. So you must subtract 1 from the value returned by sceScreamGetMixSnapshotCount() to determine the index of the last snapshot.

See Also

sceScreamGetMixSnapshotCount(), SceScreamPlatformInitEx2.*pGroupMixerFile*, sceScreamGetActiveMixSnapshotNames()

Document serial number: 000004892117

sceScreamGetMixSnapshotPriority

Retrieves the priority setting for a mix snapshot.

Definition

```
bool sceScreamGetMixSnapshotPriority(
    const char *name,
    uint32_t *priority
);
```

Arguments

name	(Input) Name of a snapshot for which to retrieve the priority setting.
priority	(Output) Pointer to a `uint32_t` variable in which to receive the snapshot's priority.

Return Values

Returns `TRUE` if the priority retrieval operation was successful, returns `FALSE` if not.

Description

This function retrieves the priority setting for a mix snapshot.

See Also

`sceScreamIsMixSnapshotActive()`, `sceScreamGetActiveMixSnapshotCount()`

sceScreamIsMixSnapshotActive

Queries whether a mix snapshot is active.

Definition

```
bool sceScreamIsMixSnapshotActive(
    const char *name
);
```

Arguments

name	(Input) Name of a snapshot to query.

Return Values

Returns `TRUE` if the specified snapshot is active; returns `FALSE` if the snapshot is inactive.

Description

This function queries whether a mix snapshot, specified by name, is active.

See Also

sceScreamGetActiveMixSnapshotCount(), sceScreamGetMixSnapshotPriority()

sceScreamSetGroupMixerBaseLevel

Sets the group mixer base level.

Definition

```
int32_t sceScreamSetGroupMixerBaseLevel(
    float dbLevel,
    float fadeTimeSeconds
);
```

Arguments

dbLevel	(Input) Group mixer base level. Expressed in dB.
fadeTimeSeconds	(Input) Fade time for the group mixer to reach the target base level from its current level. Expressed in seconds.

Return Values

Returns SCE_SCREAM_SS_ERROR_OK if the operation was successful, otherwise returns SCE_SCREAM_SS_ERROR_INVALID_PARAMETER if an invalid parameter value was detected.

Description

This function sets the group mixer base level. The group mixer base level provides an adjustment control on all Group volume levels, before further scaling by the Group Master level. It is conceived as an additional headroom control, and is part of the foundation mix settings included in a group mixer file. The initial base level setting is defined in a group mixer file, which can be loaded into Scream at initialization time. See the SceScreamPlatformInitEx2 *pGroupMixerFile* member.

Notes

Setting the group mixer base level from the API overrides the initial base level set by a group mixer file. This essentially alters the foundation mix, and thus may offset the effects of activating mix snapshots in unpredictable ways. When setting the mixer base level, it is prudent to store its initial value, so that you can reset to this value later, allowing your audio designer's foundation mix settings to remain intact. For further details, see the “Working with Mix Snapshots” chapter in the *Scream Library Overview*.

See Also

SceScreamPlatformInitEx2.*pGroupMixerFile*

Pre-Master Submix Functions

Summary

Pre-master submix functions set pre-master submix buss effects individually and collectively, by setting presets and setting all properties.

Function	Description
sceScreamPremasterSubmixSetAllProperties	Sets the properties of all effects on a specified pre-master submix buss.
sceScreamPremasterSubmixSetCustomPreset	Sets a custom effects preset, referenced by index, on a specified pre-master submix buss.
sceScreamPremasterSubmixSetCustomPresetByName	Sets a custom effects preset, referenced by name, on a specified pre-master submix buss.
sceScreamSynthPremasterSubmixConnectSideChainInput	Connects a pre-master submix side-chain compression input to another pre-master submix.
sceScreamSynthPremasterSubmixSetOutputGain	Sets a pre-master submix output gain.
sceScreamSynthPremasterSubmixSetupCompressor	Sets up a pre-master submix compressor.

sceScreamPremasterSubmixSetAllProperties

Sets the properties of all effects on a specified pre-master submix buss.

Definition

```
int32_t sceScreamPremasterSubmixSetAllProperties(
    uint32_t premasterSubmixID,
    const SceScreamSndPremasterSubmixProps *properties
);
```

Arguments

premasterSubmixID	(Input) Zero-based index of the target pre-master submix upon which to set all effect properties.
properties	(Input) Pointer to a SceScreamSndPremasterSubmixProps structure, appropriately initialized with effect property values.

Return Values

Returns SCE_SCREAM_SS_ERROR_OK if the operation was successful, otherwise returns SCE_SCREAM_SS_ERROR_INVALID_PARAMETER if an invalid parameter value was detected.

Description

This function sets the properties of all effects on a specified pre-master submix buss.

Notes

To set properties for individual pre-master submix effects, you can use the sceScreamSynthPremasterSubmixSetupCompressor() function.

See Also

SceScreamSndPremasterSubmixProps, sceScreamPremasterSubmixSetCustomPreset(), sceScreamPremasterSubmixSetCustomPresetByName(), sceScreamSynthPremasterSubmixSetupCompressor()

sceScreamPremasterSubmixSetCustomPreset

Sets a custom effects preset, referenced by index, on a specified pre-master submix buss.

Definition

```
int32_t sceScreamPremasterSubmixSetCustomPreset(
    uint32_t premasterSubmixID,
    SceScreamIniHandle iniFile,
    uint32_t presetIndex
);
```

Arguments

premasterSubmixID	(Input) Zero-based index of the target pre-master submix upon which to set a custom effect preset.
iniFile	(Input) Handle of a presets file that contains the desired custom preset.
presetIndex	(Input) Zero-based index of the desired custom preset within the specified INI file. Range: 0 to (*numPresets* – 1), where *numPresets* represents the number of presets in the specified presets file, and can be determined by calling sceScreamPresetFileGetPresetCount(), passing the appropriate SceScreamIniHandle.

Return Values

Returns SCE_SCREAM_SS_ERROR_OK if the operation was successful, otherwise returns SCE_SCREAM_SS_ERROR_INVALID_PARAMETER if an invalid parameter value was detected.

Description

This function sets a custom effects preset on a specified pre-master submix buss. You load custom presets files (in INI format) into Scream using the sceScreamPresetFileLoad() function. The sceScreamPremasterSubmixSetCustomPreset() function allows you specify the desired preset by index – rather than by its name – within the presets file.

Notes

To specify a preset by its name within the file, use the sceScreamPremasterSubmixSetCustomPresetByName() function.

See Also

sceScreamPremasterSubmixSetCustomPresetByName(), sceScreamPremasterSubmixSetAllProperties()

sceScreamPremasterSubmixSetCustomPresetByName

Sets a custom effects preset, referenced by name, on a specified pre-master submix buss.

Definition

```
int32_t sceScreamPremasterSubmixSetCustomPresetByName(
    uint32_t premasterSubmixID,
    SceScreamIniHandle iniFile,
    const char *presetName
);
```

Arguments

premasterSubmixID	(Input) Zero-based index of the target pre-master submix upon which to set a custom effects preset.
iniFile	(Input) Handle of a presets file that contains the desired custom preset.
presetName	(Input) A string matching one of the preset names within the specified presets file.

Return Values

Returns SCE_SCREAM_SS_ERROR_OK if the operation was successful, otherwise returns SCE_SCREAM_SS_ERROR_INVALID_PARAMETER if an invalid parameter value was detected.

Description

This function sets a custom effects preset on a specified pre-master submix buss. You load custom presets files (in INI format) into Scream using the sceScreamPresetFileLoad() function. The sceScreamPremasterSubmixSetCustomPresetByName() function allows you specify the desired preset by name – rather than by its index – within the presets file.

Notes

To specify a preset by its index within the file, use the sceScreamPremasterSubmixSetCustomPreset() function.

See Also

sceScreamPremasterSubmixSetCustomPreset(), sceScreamPremasterSubmixSetAllProperties()

sceScreamSynthPremasterSubmixConnectSideChainInput

Connects a pre-master submix side-chain compression input to another pre-master submix.

Definition

```
int32_t sceScreamSynthPremasterSubmixConnectSideChainInput(
    uint32_t premasterCompSubmixID,
    uint32_t premasterScCompSubmixID
);
```

Arguments

premasterCompSubmixID	(Input) Zero-based index of the destination pre-master submix, that is, the submix containing the compressor into which to receive side-chain input signal.
premasterScCompSubmixID	(Input) Zero-based index of the source pre-master submix, that is, the side-chain input signal to drive the destination compressor.

Return Values

Returns SCE_SCREAM_SS_ERROR_OK if the operation was successful, otherwise returns SCE_SCREAM_SS_ERROR_INVALID_PARAMETER if an invalid parameter value was detected.

Description

This function sets output signal from a pre-master submix as side-chain compression input signal on another pre-master submix. You specify the index of the pre-master submix to serve as side-chain input signal in the *premasterScCompSubmixID* parameter, as well as the index of the pre-master submix into which it feeds in the *premasterCompSubmixID* parameter.

You specify values for the *premasterCompSubmixID* and *premasterScCompSubmixID* parameters using zero-based indices. The range of index values you can specify depends on the number of pre-master submix busses allocated in the *numPremasterCompSubmixes* member of the SceScreamSystemParams structure. The first pre-master submix index is always SCE_SCREAM_SND_OUTPUT_DEST_PREMASTER_0 (zero). The last pre-master submix index is (n – 1), where n is the number of allocated pre-master submix busses.

See Also

sceScreamSynthPremasterSubmixSetupCompressor(),
sceScreamSynthPremasterSubmixSetOutputGain()

sceScreamSynthPremasterSubmixSetOutputGain

Sets a pre-master submix output gain.

Definition

```
int32_t sceScreamSynthPremasterSubmixSetOutputGain(
    uint32_t premasterSubmixID,
    float gainLinear
);
```

Arguments

premasterSubmixID	(Input) Zero-based index of the pre-master submix upon which to set output gain level.
gainLinear	(Input) Output gain, expressed on a linear scale. Range: SCE_SCREAM_SND_MIN_GAIN to SCE_SCREAM_SND_MAX_GAIN.

Return Values

Returns SCE_SCREAM_SS_ERROR_OK if the operation was successful, otherwise returns SCE_SCREAM_SS_ERROR_INVALID_PARAMETER if an invalid parameter value was detected or SCE_SCREAM_SS_ERROR_SYNTH_INIT_FAILED if the call cannot proceed due to a synth configuration error.

Description

This function sets a pre-master submix output gain to the master buss.

The *premasterSubmixID* parameter identifies a pre-master submix buss using a zero-based index. The range of index values you can specify depends on the number of pre-master submix busses allocated in the *numPremasterCompSubmixes* member of the SceScreamSystemParams structure. The first pre-master submix index is always SCE_SCREAM_SND_OUTPUT_DEST_PREMASTER_0 (zero). The last pre-master submix index is (n – 1), where n is the number of allocated pre-master submix busses.

See Also

sceScreamSynthPremasterSubmixConnectSideChainInput(),
sceScreamSynthPremasterSubmixSetupCompressor()

sceScreamSynthPremasterSubmixSetup Compressor

Sets up a pre-master submix compressor.

Definition

```
int32_t sceScreamSynthPremasterSubmixSetupCompressor(
    uint32_t premasterSubmixID,
    bool effectOn,
    bool linkedChannels,
    bool peakMode,
    float thresholdDB,
    float ratio,
    float attackTimeMS,
    float releaseTimeMS,
    float makeupGainDB,
    float softKneeDB
);
```

Arguments

premasterSubmixID	(Input) Zero-based index of the pre-master submix upon which to set up a compressor.
effectOn	(Input) A Boolean value that determines whether the compressor is on or off. Set to `TRUE` for on; set to `FALSE` for off.
linkedChannels	(Input) A Boolean value that determines whether to link the input channels to retain the original panning image. Set to `TRUE` for channel linking; set to `FALSE` for channel independence. See "Notes" below.
peakMode	(Input) A Boolean value that determines whether to use peak mode. Set to `TRUE` for peak mode; set to `FALSE` for RMS mode. See "Notes" below.
thresholdDB	(Input) Operation threshold of the compressor. Expressed in dB. Range: -100.0 to 0.0.
ratio	(Input) Compression ratio. Must be greater than 0.0. For input signal compression use values greater than 1.0; for expansion use values less than 1.0. For example, for a compression ratio of 2:1, use 2.0.
attackTimeMS	(Input) Attack time of the compressor. Expressed in milliseconds. Must be greater than or equal to 0.0. Typically within the range of around 0 to 5 milliseconds.
releaseTimeMS	(Input) Release time of the compressor. Expressed in milliseconds. Must be greater than or equal to 0.0. Typically within the range of around 500 to 1000 milliseconds.
makeupGainDB	(Input) Output make-up gain. Expressed in dB. Range: -100.0 to 100.0.
softKneeDB	(Input) Defines the limits of an amplitude range, centered around the *thresholdDB* level, over which the compression response curve operates. Range: -100.0 to 0.0 dB, where 0 dB produces no softening of the compression response curve (that is, a 'hard knee').

Return Values

Returns `SCE_SCREAM_SS_ERROR_OK` if the operation was successful, otherwise returns `SCE_SCREAM_SS_ERROR_INVALID_PARAMETER` if an invalid parameter value was detected.

Description

This function sets up a pre-master submix compressor.

The *premasterSubmixID* parameter identifies a pre-master submix buss using a zero-based index. The range of index values you can specify depends on the number of pre-master submix busses allocated in the *numPremasterCompSubmixes* member of the SceScreamSystemParams structure. The first pre-master submix index is always SCE_SCREAM_SND_OUTPUT_DEST_PREMASTER_0 (zero). The last pre-master submix index is (*n* – 1), where *n* is the number of allocated pre-master submix busses.

You can also set up parameters in a SceScreamSndPremasterSubmixProps structure and call sceScreamPremasterSubmixSetAllProperties() to set pre-master submix compressor values. The members in SceScreamSndPremasterSubmixProps are nearly identical to this function's parameters.

Notes

Channel linking preserves the panning image, but is a more intrusive compression mode because each channel is compressed equally based on the *peakMode* setting.

In peak mode, the compressor responds to the instantaneous level of the input signal. Peak mode can produce more quick-reacting and obvious results. In RMS mode, the compressor responds to an averaged level of the input signal. RMS mode can produce more relaxed and subtle results.

See Also

sceScreamSynthPremasterSubmixConnectSideChainInput(), sceScreamSynthPremasterSubmixSetOutputGain(), SceScreamSndPremasterSubmixProps, sceScreamPremasterSubmixSetAllProperties(), sceScreamSynthMasterSetupCompressor()

Master Speakers Buss Functions

Summary

Master speakers buss functions set master speakers buss effects and retrieve levels.

Function	Description
sceScreamSynthMasterSetupCompressor	Sets up the master buss compressor.

sceScreamSynthMasterSetupCompressor

Sets up the master buss compressor.

Definition

```
int32_t sceScreamSynthMasterSetupCompressor(
    bool effectOn,
    bool linkedChannels,
    bool peakMode,
    float thresholdDB,
    float ratio,
    float attackTimeMS,
    float releaseTimeMS,
    float makeupGainDB,
    float softKneeDB
);
```

Arguments

effectOn	(Input) A Boolean value that determines whether the compressor is on or off. Set to TRUE for on; set to FALSE for off.
linkedChannels	(Input) A Boolean value that determines whether to link the input channels to retain the original panning image. Set to TRUE for channel linking; set to FALSE for channel independence. See "Notes" below.
peakMode	(Input) A Boolean value that determines whether to use peak mode. Set to TRUE for peak mode; set to FALSE for RMS mode. See "Notes" below.
thresholdDB	(Input) Operation threshold of the compressor. Expressed in dB. Range: -100.0 to 0.0.
ratio	(Input) Compression ratio. Must be greater than 0.0. For input signal compression use values greater than 1.0; for expansion use values less than 1.0. For example, for a compression ratio of 2:1, use 2.0.
attackTimeMS	(Input) Attack time of the compressor. Expressed in milliseconds. Must be greater than or equal to 0.0. Typically within the range of around 0 to 5 milliseconds.
releaseTimeMS	(Input) Release time of the compressor. Expressed in milliseconds. Must be greater than or equal to 0.0. Typically within the range of around 500 to 1000 milliseconds.
makeupGainDB	(Input) Output make-up gain. Expressed in dB. Range: -100.0 to 100.0.
softKneeDB	(Input) Defines the limits of an amplitude range, centered around the *thresholdDB* level, over which the compression response curve operates. Range: -100.0 to 0.0 dB, where 0 dB produces no softening of the compression response curve (that is, a 'hard knee').

Return Values

Returns SCE_SCREAM_SS_ERROR_OK if the operation was successful, otherwise returns SCE_SCREAM_SS_ERROR_INVALID_PARAMETER if an invalid parameter value was detected.

Description

This function sets up the master buss compressor.

Notes

Channel linking preserves the panning image but is a more intrusive compression mode because each channel is compressed equally based on the RMS averaging of all channels.

In peak mode, the compressor responds to the instantaneous level of the input signal. Peak mode can produce more quick-reacting and obvious results. In RMS mode, the compressor responds to an averaged level of the input signal. RMS mode can produce more relaxed and subtle results.

See Also

sceScreamSynthPremasterSubmixSetupCompressor()

NGS Direct Access Functions

Summary

NGS direct access functions allow you to route external NGS voices to the Scream master buss and pre-master submix busses.

Function	Description
sceScreamSynthGetMasterVoiceHandle	Retrieves the Scream master voice handle.
sceScreamSynthGetNGSSystemHandle	Retrieves the NGS system handle.
sceScreamSynthGetPremasterSubmixVoiceHandle	Retrieves a pre-master submix voice handle.

sceScreamSynthGetMasterVoiceHandle

Retrieves the Scream master voice handle.

Definition

```
void *sceScreamSynthGetMasterVoiceHandle(void);
```

Return Values

Returns the master voice handle.

Description

This function retrieves the Scream master voice handle. You use the master voice handle to route external NGS voices to the Scream master voice.

Notes

Cast the return value to a `SceNgsHSynSystem` type.

See Also

`sceScreamSynthGetNGSSystemHandle()`,
`sceScreamSynthGetPremasterSubmixVoiceHandle()`

sceScreamSynthGetNGSSystemHandle

Retrieves the NGS system handle.

Definition

```
void *sceScreamSynthGetNGSSystemHandle(void);
```

Return Values

Returns the NGS system handle.

Description

This function retrieves the NGS system handle.

Notes

Cast the return value to a `SceNgsHSynSystem` type. This handle is necessary in order to create NGS voice racks.

See Also

`sceScreamSynthGetMasterVoiceHandle()`,
`sceScreamSynthGetPremasterSubmixVoiceHandle()`

sceScreamSynthGetPremasterSubmixVoiceHandle

Retrieves a pre-master submix voice handle.

Definition

```
void *sceScreamSynthGetPremasterSubmixVoiceHandle(
    uint32_t premasterSubmixID
);
```

Arguments

premasterSubmixID	(Input) Zero-based index of a pre-master submix from which to retrieve a handle.

Return Values

Returns the specified pre-master submix voice handle.

Description

This function retrieves a pre-master submix voice handle. You use pre-master submix voice handles to route to the corresponding Scream pre-master submixes.

Notes

Cast the return value to a `SceNgsHSynSystem` type.

See Also

`sceScreamSynthGetNGSSystemHandle()`, `sceScreamSynthGetMasterVoiceHandle()`

Utility Functions

Summary

Utility functions calculate Doppler pitch transpose and 3D sound spatialization parameters. For more information on spatialization, see “Sound Spatialization” in the “Working with Distance Models, Doppler, and Spatialization” chapter of the *Scream Library Overview*.

Function	Description
sceScreamCalcSoundAngles	Calculates azimuth and elevation angles for 3D sound spatialization.
sceScreamCreateListener	Creates a three-dimensional (3D) sound spatialization listener.
sceScreamDeleteListener	Deletes a 3D sound spatialization listener.
sceScreamGetDopplerPitchTranspose	Calculates a pitch transposition amount used for the Doppler effect.
sceScreamGetListener	Gets the location and orientation of a 3D sound spatialization listener.
sceScreamGetWorldUnitsPerMeter	Retrieves the currently assigned number of game-world units per meter.
sceScreamSetListener	Sets the location and orientation of a 3D sound spatialization listener.
sceScreamSetWorldUnitsPerMeter	Sets the number of game-world units per meter.

sceScreamCalcSoundAngles

Calculates azimuth and elevation angles for 3D sound spatialization.

Definition

```
int32_t sceScreamCalcSoundAngles(
    uint32_t handle,
    const SceScreamSnd3DVector *location,
    uint32_t *azimuth,
    int32_t *elevation
);
```

Arguments

handle	(Input) The handle of the listener for which to calculate azimuth/elevation data, a value returned by sceScreamCreateListener().
location	(Input) Pointer to a SceScreamSnd3DVector structure containing the location of the sound-emitting object in game world space.
azimuth	(Output) Pointer to a uint32_t variable in which to receive the azimuth angle for the sound-emitting object.
elevation	(Output) Pointer to an int32_t variable in which to receive the elevation angle for the sound-emitting object.

Return Values

Returns 0 if successful. Returns SCE_SCREAM_SS_ERROR_INVALID_PARAMETER if an invalid parameter value was detected. Returns SCE_SCREAM_SS_ERROR_SYSTEM_NOT_STARTED if Scream is not currently running.

Description

This function calculates the azimuth and elevation angles between a specified listener and sound-emitting object. You specify the location of the sound-emitting object as a SceScreamSnd3DVector structure in the *location* parameter.

The output *azimuth* and *elevation* values from this function can be utilized for 3D sound spatialization. The azimuth value is expressed in degrees, 0 being directly in front of, and 180 being directly behind the listener. You can apply the *azimuth* value directly to the SceScreamSoundParams *azimuth* member.

The *elevation* value is also expressed in degrees, 0 being at listener height, 90 directly above, and -90 directly below the listener. The SceScreamSoundParams structure, however, does not include an *elevation* member. Instead, Scream provides a way to manage distance cues through panning focus. Closer sounds – which tend to be more dispersed – are given wider focus. Conversely, more distant sounds are given narrower focus. Developers are free to choose from a variety of potential approaches for mapping *elevation* to focus, or even to ignore this value. One approach, for example, could be a simple linear calculation such as:

```
focus = ( abs(elevation) / 90 ) * 360 )
```

For an example of using this function, see "Sound Spatialization" in the "Working with Distance Models, Doppler, and Spatialization" chapter of the *Scream Library Overview*.

See Also

SceScreamSoundParams, sceScreamCreateListener(), sceScreamDeleteListener(), sceScreamGetListener(), sceScreamSetListener(), SceScreamSnd3DVector

sceScreamCreateListener

Creates a three-dimensional (3D) sound spatialization listener.

Definition

```
uint32_t sceScreamCreateListener(void);
```

Return Values

If successful, returns a `uint32_t` representing the listener handle. Otherwise returns zero.

Description

This function creates a listener used for 3D sound spatialization. Along with `sceScreamSetListener()` and `sceScreamCalcSoundAngles()`, it is used to calculate the polar coordinates of sound-emitting object(s) relative to a listener, based on game world Cartesian coordinates.

A listener is a proxy for a listening point in game world space. For example, this might be a game character's ears, or in a sports game, the location of a camera.

The function finds an available listener (from a pool of `SCE_SCREAM_SND_MAXLISTENERS`), and allocates it.

For more information on using listeners, see the "Working with Distance Models, Doppler, and Spatialization" chapter of the *Scream Library Overview*.

See Also

`sceScreamDeleteListener()`, `sceScreamSetListener()`, `sceScreamGetListener()`, `sceScreamCalcSoundAngles()`

sceScreamDeleteListener

Deletes a 3D sound spatialization listener.

Definition

```
int32_t sceScreamDeleteListener(
    uint32_t handle
);
```

Arguments

handle	(Input) The handle of the listener you wish to delete, a value returned by sceScreamCreateListener().

Return Values

Returns 0 if successful. Returns SCE_SCREAM_SS_ERROR_INVALID_PARAMETER if an invalid parameter value was detected. Returns SCE_SCREAM_SS_ERROR_SYSTEM_NOT_STARTED if Scream is not currently running.

Description

This function deallocates a 3D sound spatialization listener and returns it to a pool of SCE_SCREAM_SND_MAXLISTENERS.

See Also

sceScreamCreateListener(), sceScreamSetListener(), sceScreamGetListener(), sceScreamCalcSoundAngles()

sceScreamGetDopplerPitchTranspose

Calculates a pitch transposition amount used for the Doppler effect.

Definition

```
int32_t sceScreamGetDopplerPitchTranspose(
    float32_t approachingMps
);
```

Arguments

approachingMps	(Input) Speed of an approaching sound-producing object in meters per second. Use negative amounts for objects that are moving away from the listener.

Return Values

Returns a pitch transposition value expressed in fines. See SCE_SCREAM_SND_FINES_PER_OCTAVE.

Description

This function calculates pitch transposition amounts used for the Doppler pitch shift effect. Return values are expressed in fines, and can be applied directly to the SceScreamSoundParams.*pitchTranspose* member.

Doppler is the name given to the phenomenon of pitch shifting that occurs with a moving sound source (or a moving listener). An everyday example would be an ambulance or fire truck siren that moves towards and then away from a stationary listener. The pitch of the sound source appears to increase as the sound becomes closer; and to decrease as the sound gets further away. The effect of moving sound sources is sometimes exaggerated in games for dramatic effect.

For further discussion, see "Doppler Pitch Shifting" in the "Working with Distance Models, Doppler, and Spatialization" chapter of the *Scream Library Overview*.

See Also

SceScreamSoundParams, sceScreamAutoPitchTranspose()

sceScreamGetListener

Gets the location and orientation of a 3D sound spatialization listener.

Definition

```
int32_t sceScreamGetListener(
    uint32_t handle,
    SceScreamSnd3DVector *location,
    SceScreamSnd3DVector *front,
    SceScreamSnd3DVector *top
);
```

Arguments

handle	(Input) The handle of the listener for which to retrieve location/orientation data, a value returned by sceScreamCreateListener().
location	(Output) Pointer to a SceScreamSnd3DVector structure in which to receive the location of the listener in game world space. Set to NULL if this information is not required.
front	(Output) Pointer to a SceScreamSnd3DVector structure in which to receive the listener's front orientation vector. Set to NULL if this information is not required.
top	(Output) Pointer to a SceScreamSnd3DVector structure in which to receive the listener's top orientation vector. Set to NULL if this information is not required.

Return Values

Returns 0 if successful. Returns SCE_SCREAM_SS_ERROR_INVALID_PARAMETER if an invalid parameter value was detected. Returns SCE_SCREAM_SS_ERROR_SYSTEM_NOT_STARTED if Scream is not currently running.

Description

This function retrieves the location and orientation of a 3D sound spatialization listener. Values for the listener's location, front orientation vector, and top orientation vector are stored, respectively, in SceScreamSnd3DVector structures pointed by the *location*, *front*, and *top* parameters.

See Also

sceScreamCreateListener(), sceScreamDeleteListener(), sceScreamSetListener(), sceScreamCalcSoundAngles(), SceScreamSnd3DVector

sceScreamGetWorldUnitsPerMeter

Retrieves the currently assigned number of game-world units per meter.

Definition

```
float sceScreamGetWorldUnitsPerMeter(void);
```

Return Values

The current number of game-world units per meter.

Description

This function retrieves the currently assigned number of game-world units per meter. Without setting the units by calling sceScreamSetWorldUnitsPerMeter(), game-world units per meter defaults to 1.0.

See Also

sceScreamSetWorldUnitsPerMeter(), sceScreamGetDopplerPitchTranspose(), SceScreamSnd3DGrainData

sceScreamSetListener

Sets the location and orientation of a 3D sound spatialization listener.

Definition

```
int32_t sceScreamSetListener(
    uint32_t handle,
    const SceScreamSnd3DVector *location,
    const SceScreamSnd3DVector *front,
    const SceScreamSnd3DVector *top,
    bool cameraCut
);
```

Arguments

handle	(Input) The handle of the listener you wish to set, a value returned by sceScreamCreateListener().
location	(Input) Pointer to a SceScreamSnd3DVector structure containing the location of the listener in game world space.
front	(Input) Pointer to a SceScreamSnd3DVector structure containing the front orientation vector relative to the location of the listener.
top	(Input) Pointer to a SceScreamSnd3DVector structure containing the top orientation vector relative to the location of the listener.
cameraCut	(Input) A Boolean value indicating, for purpose of Doppler pitch shift calculation, whether there is a discontinuity in listener location. Set to TRUE if a camera cut is occurring this frame. Otherwise, set to FALSE. See "Notes" below.

Return Values

Returns 0 if successful. Returns SCE_SCREAM_SS_ERROR_INVALID_PARAMETER if an invalid parameter value was detected. Returns SCE_SCREAM_SS_ERROR_SYSTEM_NOT_STARTED if Scream is not currently running.

Description

This function sets the location and orientation of a 3D sound spatialization listener. The *location*, *front*, and *top* parameters specify, respectively, the location, front, and top orientation vectors of the listener.

Notes

When using Scream's Doppler effect, you can set the *cameraCut* parameter to TRUE to indicate a discontinuity in listener (or camera) location. This avoids the potential for large instantaneous Doppler pitch shifts that might otherwise result. A discontinuity in listener position is the more common camera cut scenario. You can also specify a camera cut when there has been a discontinuity in emitter location. For further details, see the SCE_SCREAM_SND_FLAG_DOPPLER_CAMERA_CUT Sound flag. Specification of both listener *and* emitter camera cuts within the same frame is unnecessary.

See Also

sceScreamCreateListener(), sceScreamDeleteListener(), sceScreamGetListener(), sceScreamCalcSoundAngles(), SceScreamSnd3DVector, SCE_SCREAM_SND_FLAG_DOPPLER_CAMERA_CUT

sceScreamSetWorldUnitsPerMeter

Sets the number of game-world units per meter.

Definition

```
int32_t sceScreamSetWorldUnitsPerMeter(
    float unitsPerMeter
);
```

Arguments

unitsPerMeter (Input) The number of game-world units per meter.

Return Values

If successful, returns SCE_SCREAM_SS_ERROR_OK. Otherwise, returns SCE_SCREAM_SS_ERROR_INVALID_PARAMETER.

Description

This function sets the number of game-world units per meter. Scream uses meters as its unit of distance. Distance calculations for distance model input, and velocity calculations for Scream's Doppler effect, are based on values specified for Sound and listener 3D positional coordinates. These values are assumed to be expressed in meters. In your game, if distance is expressed in units other than meters, you can use this function to instruct Scream to convert your distance units to meters. Thereafter, you can specify Sound and listener positions in your game-world units, and Scream adjusts its calculations accordingly. For example, if your game uses feet as its unit measure of distance, you should set this value to approximately 3.281, which is the number of feet in a meter.

See Also

sceScreamGetWorldUnitsPerMeter(), sceScreamGetDopplerPitchTranspose(), SceScreamSnd3DVector, sceScreamSetListener(), SceScreamSoundParams.*position*,

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.

Error Codes

Error codes used with the sceScreamGetLastLoadError() function.

Define	Value	Description
SCE_SCREAM_SS_ERROR_OK	(0)	Successful operation. No error detected. Returned by numerous functions.
SCE_SCREAM_SND_LOAD_ERROR_COULDNT_OPEN_FILE	SCE_SCREAM_MAKE_ERROR (0x000)	A file could not be opened. Returned by the sceScreamGetLastLoadError() function.
SCE_SCREAM_SND_LOAD_ERROR_READING_FILE	SCE_SCREAM_MAKE_ERROR (0x001)	A problem occurred reading a file. Returned by the sceScreamGetLastLoadError() function.
SCE_SCREAM_SND_LOAD_ERROR_MEMORY	SCE_SCREAM_MAKE_ERROR (0x002)	Memory could not be allocated to load a file. Returned by the sceScreamGetLastLoadError() function.
SCE_SCREAM_SND_LOAD_ERROR_ALIGNMENT	SCE_SCREAM_MAKE_ERROR (0x003)	A Bank is not aligned on a 16-byte boundary. Returned by the sceScreamGetLastLoadError() function.
SCE_SCREAM_SND_LOAD_ERROR_INVALID_FORMAT	SCE_SCREAM_MAKE_ERROR (0x004)	A file has an invalid format. Returned by the sceScreamGetLastLoadError() function.
SCE_SCREAM_SND_LOAD_ERROR_ALREADY_LOADED	SCE_SCREAM_MAKE_ERROR (0x005)	A Bank at the specified memory address is already loaded. Returned by the sceScreamGetLastLoadError() function.
SCE_SCREAM_SS_ERROR_SYSTEM_ALREADY_STARTED	SCE_SCREAM_MAKE_ERROR (0x101)	Scream is already initialized, or that initialization failed. Returned by the sceScreamStartSoundSystemEx2() function.
SCE_SCREAM_SS_ERROR_SYSTEM_NOT_STARTED	SCE_SCREAM_MAKE_ERROR (0x102)	Scream is not currently running. Returned by the sceScreamStopSoundSystem() function.
SCE_SCREAM_SS_ERROR_SYNTH_INIT_FAILED	SCE_SCREAM_MAKE_ERROR (0x103)	Underlying synthesizer failed to initialize. Returned by the sceScreamStartSoundSystemEx2() function.
SCE_SCREAM_SS_ERROR_INVALID_PARAMETER	SCE_SCREAM_MAKE_ERROR (0x104)	Invalid parameter detected. Returned by numerous functions.
SCE_SCREAM_SS_ERROR_UNSUPPORTED	SCE_SCREAM_MAKE_ERROR (0x105)	Indicates that a requested operation is not supported.