NGS Reference

PlayStation®Vita Programmer Tool Runtime Library Release 3.50

Document serial number: 000004892117

Document serial number: 000004892117

Table of Contents

NGS Reference Overview

About the NGS Reference

Introduction

This document describes the members of the NGS API. Many DSP effect Modules are supplied with NGS and can be used in conjunction with Voice Definition templates to affect the audio output. For more information on the supplied DSP modules, see *NGS Modules Overview* and *NGS Modules Reference*.

NGS Return Codes

All `sceNgs*` functions return one of the following codes:

Define	Value	Description
SCE_NGS_OK	SCE_OK	OK [0x00000000].
SCE_NGS_ERROR	-2142633983	Error [0x804A0001].
SCE_NGS_ERROR_INVALID_PARAM	-2142633982	Invalid parameter [0x804A0002].
SCE_NGS_ERROR_INVALID_ALIGNMENT	-2142633981	Invalid buffer alignment [0x804A0003].
SCE_NGS_ERROR_NOT_IMPL	-2142633980	Not implemented on current platform [0x804A0004].
SCE_NGS_ERROR_DEPENDENCY	-2142633979	Dependency error. For example, a circular dependency list with patching [0x804A0005].
SCE_NGS_ERROR_OUT_OF_ASSETS	-2142633978	Reached the maximum limit of allocated assets (see `sceNgsSystemInitParams` structure) [0x804A0006].
SCE_NGS_ERROR_MODULE_NOT_AVAIL	-2142633977	Module not available [0x804A0007].
SCE_NGS_ERROR_RESOURCE_LOCKED	-2142633976	Resource has already been locked [0x804A0008].
SCE_NGS_ERROR_PARAM_OUT_OF_RANGE	-2142633975	Module parameter out of range [0x804A0009].
SCE_NGS_ERROR_INVALID_VOICE_TYPE	-2142633974	Voice type incorrect [0x804A000A].
SCE_NGS_ERROR_SYSTEM_MISMATCH	-2142633973	Systems do not match [0x804A000B].
SCE_NGS_ERROR_INVALID_HANDLE	-2142633972	Invalid handle supplied to function [0x804A000C].
SCE_NGS_ERROR_SIZE_MISMATCH	-2142633971	Size mismatch between supplied and expected [0x804A000D].
SCE_NGS_ERROR_PATCH_NOT_AVAIL	-2142633970	Patch out of range or not connected [0x804A000E].
SCE_NGS_ERROR_PARAM_TYPE_MISMATCH	-2142633969	Parameter and module type mismatch [0x804A000F].
SCE_NGS_ERROR_INVALID_STATE	-2142633968	Internal state in unexpected or incorrect mode [0x804A0010].
SCE_NGS_ERROR_INTERNAL_ALLOC	-2142633967	System has failed to allocate memory from internal resource pools [0x804A0011].
SCE_NGS_ERROR_INTERNAL_PROCESSING	-2142633966	System has caused an error during internal processing [0x804A0012].

Define	Value	Description
SCE_NGS_ERROR_INVALID_BUFFER	-2142633965	Invalid data buffer (e.g. PCM/VAG/ ATRAC9 ™) supplied to system, check size and pointers [0x804A0013].

#defines

General Defines

General Defines Table

Define	Value	Description
SCE_NGS_MODULE_ALL_CHANNELS	(-1)	Specifies that information can be set on all channels.
SCE_NGS_INVALID_HANDLE	(0)	Invalid handle.
SCE_NGS_MEMORY_ALIGN_SIZE	(16)	Memory align size.
SCE_NGS_MAX_SYSTEM_CHANNELS	(2)	Maximum number of audio channels available to the system.
SCE_NGS_SYSTEM_FLAG_NO_MODULE_PARAM_CHECKING	(0)	Each sceNgsVoiceUnlockParams() and sceNgsVoiceSetParamsBlock() call will *not* validate the parameter ranges.
SCE_NGS_SYSTEM_FLAG_MODULE_PARAM_CHECKING	(1)	Each sceNgsVoiceUnlockParams() and sceNgsVoiceSetParamsBlock() call will validate the parameter ranges.
SCE_NGS_SYSTEM_FLAG_DEFAULT	(SCE_NGS_SYSTEM_FLAG_MODULE_PARAM_CHECKING)	The default mode for module parameter checking.
SCE_NGS_VOICE_INIT_BASE	(0)	Initialize basic, resets play and pause flags.
SCE_NGS_VOICE_INIT_ROUTING	(1)	Initialize routing to default settings (no active patches).
SCE_NGS_VOICE_INIT_PRESET	(2)	Initialize using preset values. This will use the supplied voice preset (see sceNgsVoiceInit()).
SCE_NGS_VOICE_INIT_CALLBACKS	(4)	Initialize the callbacks per Module.
SCE_NGS_VOICE_INIT_ALL	(SCE_NGS_VOICE_INIT_BASE | SCE_NGS_VOICE_INIT_CALLBACKS | SCE_NGS_VOICE_INIT_ROUTING | SCE_NGS_VOICE_INIT_PRESET)	Initialize everything (7).
SCE_NGS_VOICE_STATE_AVAILABLE	(0)	Voice available.
SCE_NGS_VOICE_STATE_ACTIVE	(1)	Voice currently active.
SCE_NGS_VOICE_STATE_FINALIZING	(4)	Voice is in key-off mode, but is still processing.
SCE_NGS_VOICE_STATE_UNLOADING	(8)	Voice is unloading (Rack is being removed from system and therefore the Voice is unavailable).
SCE_NGS_VOICE_STATE_PENDING	(16)	Voice has been requested to play, but has not yet started processing.
SCE_NGS_VOICE_STATE_PAUSED	(32)	Voice has been set to a pause state.
SCE_NGS_VOICE_STATE_KEY_OFF	(64)	Voice has been requested to key-off but not yet started processing.
SCE_NGS_VOICE_PATCH_AUTO_SUBINDEX	(-1)	Setup flag for patching; allocates patch on any free sub-patch for the given output.

Define	Value	Description
SCE_NGS_MODULE_FLAG_BYPASSED	(2)	Specifies that the Module in the Voice is to be bypassed.
SCE_NGS_MODULE_FLAG_NOT_BYPASSED	(0)	Specifies that the Module in the Voice is not to be bypassed.
SCE_NGS_NO_CALLBACK	(0)	Specifies no callback.
SCE_NGS_SAMPLE_OFFSET_FROM_AT9_HEADER	(0x80000000)	Used with sceNgsAT9GetSectionDetails(). This should be bitwise OR'ed with sample offset, if taking loop information from the ATRAC9™ file header.

Voice Template Defines

Voice Template Defines Table

Define	Value	Description
SCE_NGS_ENV_BUSS_ENVELOPE_MODULE	(1)	Envelope Module.
SCE_NGS_EQ_VOICE_PARA_EQ_MODULE	(1)	Parametric EQ Module.
SCE_NGS_MASTER_BUSS_OUTPUT_MODULE	(1)	Output Module.
SCE_NGS_SAS_EMU_VOICE_PCM_PLAYER	(0)	libSAS Emulation Player Module
SCE_NGS_SAS_EMU_VOICE_ENVELOPE	(1)	Envelope Module.
SCE_NGS_SAS_EMU_NUM_MODULES	(2)	Number of modules.
SCE_NGS_SIMPLE_VOICE_PCM_PLAYER	(0)	Simple Voice PCM Module.
SCE_NGS_SIMPLE_VOICE_EQ	(1)	EQ Module.
SCE_NGS_SIMPLE_VOICE_ENVELOPE	(2)	Envelope Module.
SCE_NGS_SIMPLE_VOICE_PAUSER	(3)	Pauser Module.
SCE_NGS_SIMPLE_VOICE_SEND_1_FILTER	(4)	Send 1 Filter enum.
SCE_NGS_SIMPLE_VOICE_SEND_2_FILTER	(5)	Send 2 Filter enum.
SCE_NGS_SIMPLE_VOICE_NUM_MODULES	(6)	Number of modules.
SCE_NGS_SIMPLE_VOICE_AT9_PLAYER	(0)	Simple ATRAC9™ PCM Voice Module.
SCE_NGS_SIMPLE_VOICE_AT9_EQ	(1)	EQ Module.
SCE_NGS_SIMPLE_VOICE_AT9_ENVELOPE	(2)	Envelope Module.
SCE_NGS_SIMPLE_VOICE_AT9_PAUSER	(3)	Pauser Module.
SCE_NGS_SIMPLE_VOICE_AT9_SEND_1_FILTER	(4)	Send 1 Filter enum.
SCE_NGS_SIMPLE_VOICE_AT9_SEND_2_FILTER	(5)	Send 2 Filter enum.
SCE_NGS_SIMPLE_VOICE_AT9_NUM_MODULES	(6)	Number of modules.
SCE_NGS_VOICE_T1_PCM_PLAYER	(0)	PCM Player Module.
SCE_NGS_VOICE_T1_SIGNAL_GENERATOR	(1)	Signal Generator Module.
SCE_NGS_VOICE_T1_MIXER	(2)	Mixer Module.
SCE_NGS_VOICE_T1_EQ	(3)	EQ Module.
SCE_NGS_VOICE_T1_ENVELOPE	(4)	Envelope Module.
SCE_NGS_VOICE_T1_DISTORTION	(5)	Distortion Module.
SCE_NGS_VOICE_T1_SEND_1_EQ	(6)	Send 1 EQ enum.
SCE_NGS_VOICE_T1_SEND_2_EQ	(7)	Send 2 EQ enum.
SCE_NGS_VOICE_T1_SEND_3_EQ	(8)	Send 3 EQ enum.
SCE_NGS_VOICE_T1_SEND_4_EQ	(9)	Send 4 EQ enum.
SCE_NGS_VOICE_T1_NUM_MODULES	(10)	Number of Modules.
SCE_NGS_VOICE_AT9_PLAYER	(0)	PCM Player Module.
SCE_NGS_VOICE_AT9_SIGNAL_GENERATOR	(1)	Signal Generator Module.
SCE_NGS_VOICE_AT9_MIXER	(2)	Mixer Module.
SCE_NGS_VOICE_AT9_EQ	(3)	EQ Module.
SCE_NGS_VOICE_AT9_ENVELOPE	(4)	Envelope Module.
SCE_NGS_VOICE_AT9_DISTORTION	(5)	Distortion Module.
SCE_NGS_VOICE_AT9_SEND_1_EQ	(6)	Send 1 EQ enum.
SCE_NGS_VOICE_AT9_SEND_2_EQ	(7)	Send 2 EQ enum.
SCE_NGS_VOICE_AT9_SEND_3_EQ	(8)	Send 3 EQ enum.
SCE_NGS_VOICE_AT9_SEND_4_EQ	(9)	Send 4 EQ enum.
SCE_NGS_VOICE_AT9_NUM_MODULES	(10)	Number of Modules.

NGS Type Defines

SceNgsModuleID

Global identifier for all Module descriptors.

Definition

```
#include <ngs/ngs_top.h>
typedef SceUInt32 SceNgsModuleID;
```

Description

Global identifier for all Module descriptors. (See the module ID defines list in the *NGS Modules Reference* for more information.)

See Also

`sceNgsModuleGetInfo`, `SceNgsModuleDef`, `SceNgsVoiceModule`

SceNgsParamsID

Global identifier for all Module parameters.

Definition

```
#include <ngs/ngs_top.h>
typedef SceUInt32 SceNgsParamsID;
```

Description

Global identifier for all Module parameters.

See Also

SceNgsParamsDescriptor, SCE_NGS_MAKE_PARAMS_ID

SceNgsHVoice

The Voice Handle Type.

Definition

```
#include <ngs/ngs_top.h>
typedef SceUInt32 SceNgsHVoice;
```

Description

The Voice Handle Type.

See Also

sceNgsRackGetVoiceHandle, sceNgsVoiceInit, sceNgsVoicePlay, sceNgsVoiceKeyOff, sceNgsVoiceKill, sceNgsVoicePause, sceNgsVoiceResume, sceNgsVoiceSetPreset, sceNgsVoiceGetPatchInfo, sceNgsVoiceGetOutputPatch, sceNgsVoiceRemovePatchRouting, sceNgsVoiceLockParams, sceNgsVoiceUnlockParams, sceNgsVoiceBypassModule, sceNgsVoiceGetStateData, sceNgsVoiceGetActiveState, sceNgsVoiceGetPauseState

SceNgsHPatch

The Patch Handle Type.

Definition

```
#include <ngs/ngs_top.h>
typedef SceUInt32 SceNgsHPatch;
```

Description

The Patch Handle Type.

See Also

sceNgsVoiceCreatePatchRouting, sceNgsVoiceGetPatchInfo, sceNgsVoiceGetOutputPatch, sceNgsVoiceRemovePatchRouting, sceNgsVoicePatchSetVolume, sceNgsVoicePatchSetVolumes, sceNgsVoicePatchSetVolumesMatrix

SceNgsHSynSystem

The Synth System Handle Type.

Definition

```
#include <ngs/ngs_top.h>
typedef SceUInt32 SceNgsHSynSystem;
```

Description

The Synth System Handle Type.

See Also

sceNgsSystemInit, sceNgsSystemUpdate, sceNgsSystemRelease, sceNgsSystemLock, sceNgsSystemUnlock, sceNgsSystemSetParamErrorCallback, sceNgsRackInit, sceNgsRackGetRequiredMemorySize

SceNgsHRack

The Rack Handle Type.

Definition

```
#include <ngs/ngs_top.h>
typedef SceUInt32 SceNgsHRack;
```

Description

The Rack Handle Type.

See Also

sceNgsRackInit, sceNgsRackGetVoiceHandle, sceNgsRackRelease, sceNgsRackSetParamErrorCallback

Macros

SCE_NGS_MAKE_PARAMS_ID

Macro for creating Params header IDs.

Definition

```
#include <ngs/ngs_top.h>
#define SCE_NGS_MAKE_PARAMS_ID(
        module_id,
        index,
        ver
) ( ((module_id) & 0x0000ffff) | (((ver) & 0x000000ff) << 16) | (((index) &
0x000000ff) << 24) )
```

Arguments

module_id	Identifies the module.
index	Index.
ver	Version.

Description

Creates parameter header IDs for the given module ID.

SCE_NGS_FAILED

Macro to define operation failed condition `(r) != SCE_NGS_OK`.

Definition

```
#include <ngs/ngs_top.h>
#define SCE_NGS_FAILED(
        r
) ((r) != SCE_NGS_OK)
```

Arguments

r	Macro returns `!= SCE_NGS_OK`.

Description

Defines operation failed condition based on `SCE_NGS_OK`.

SCE_NGS_SUCCESS

Macro to define operation success condition `(r) == SCE_NGS_OK`.

Definition

```
#include <ngs/ngs_top.h>
#define SCE_NGS_SUCCESS(
        r
) ((r) == SCE_NGS_OK)
```

Arguments

r	Macro returns == `SCE_NGS_OK`.

Description

Defines operation success condition based on `SCE_NGS_OK`.

Structures

SceNgsModuleParamHeader

Structure used to precede SceNgsParamsDescriptor in data passed to sceNgsVoiceSetParamsBlock() function.

Definition

```
#include <ngs/ngs_top.h>
typedef struct {
        SceInt32 moduleId;
        SceInt32 chan;
} SceNgsModuleParamHeader;
```

Members

moduleId	Module to set the parameters on.
chan	Channel to set the parameters on.

Description

Structure used to precede SceNgsParamsDescriptor (such as any module's parameter structure) in data passed to sceNgsVoiceSetParamsBlock() function. Multiple blocks can be held in a continuous segment of memory referenced by a module. Using this module ID allows for the process to run through each block one after the other, rather than individually referencing a single block. Keeping them all in the same place means for quicker processing of block data; it is not necessary to continually lock and unlock parameter block structures in order to change specific parameter data.

See Also

SceNgsParamsDescriptor, sceNgsVoiceSetParamsBlock

SceNgsParamsDescriptor

Structure to define the description of module parameters.

Definition

```
#include <ngs/ngs_top.h>
typedef struct {
        SceNgsParamsID id;
        SceUInt32 size;
} SceNgsParamsDescriptor;
```

Members

id	The unique identifier for this interface.
size	Size of entire params data, including this header.

Description

Structure to define the description of module parameters. This is a standard header appended to each module's parameter set for use with sceNgsVoiceSetParamsBlock().

See Also

SceNgsModuleParamHeader, sceNgsVoiceSetParamsBlock

SceNgsBufferInfo

Structure used to pass or receive buffer information.

Definition

```
#include <ngs/ngs_top.h>
typedef struct {
        void *data;
        SceUInt32 size;
} SceNgsBufferInfo;
```

Members

data	The pointer to the data.
size	Size of buffer.

Description

Structure used for passing or receiving buffer information.

See Also

sceNgsModuleGetPreset, sceNgsVoiceLockParams, sceNgsRackInit

SceNgsVoicePreset

Structure used to describe a Voice preset.

Definition

```
#include <ngs/ngs_top.h>
typedef struct {
        SceInt32 nNameOffset;
        SceUInt32 uNameLength;
        SceInt32 nPresetDataOffset;
        SceUInt32 uSizePresetData;
        SceInt32 nBypassFlagsOffset;
        SceUInt32 uNumBypassFlags;
} SceNgsVoicePreset;
```

Members

nNameOffset	Byte offset (from this structure) to `char*` null-terminated string for the preset name or 0 (none).
uNameLength	Length of name array.
nPresetDataOffset	Byte offset (from this structure) to `SceNgsModuleParamHeader*` header for module data in block data format or 0 (none).
uSizePresetData	Size of preset data.
nBypassFlagsOffset	Byte offset (from this structure) to `SceUInt32*` array of module IDs to bypass or 0 (none)
uNumBypassFlags	Number of elements in array of module IDs to bypass.

Description

Structure used to describe a Voice preset.

See Also

`sceNgsVoiceInit`, `sceNgsVoiceSetPreset`

SceNgsSystemInitParams

Structure required to initialize the NGS system.

Definition

```
#include <ngs/ngs_top.h>
typedef struct {
        SceInt32 nMaxRacks;
        SceInt32 nMaxVoices;
        SceInt32 nGranularity;
        SceInt32 nSampleRate;
        SceInt32 nMaxModules;
} SceNgsSystemInitParams;
```

Members

nMaxRacks	Maximum number of Racks within the System.
nMaxVoices	Maximum number of Voices within the System.
nGranularity	PCM sample granularity (NGS will process and output PCM sample packets of this size per channel). See "System Caveats, NGS Granularity" in the *NGS Overview*.
nSampleRate	Base sample rate.
nMaxModules	Maximum number of Module types that are available for the whole system. This parameter is left for future extension, and on the PlayStation®Vita system, this should be specified as any value >= 0.

Description

Structure required to initialize the NGS system.

The information set within this structure will determine the memory size that the user must allocate for the system to perform as required.

See Also

sceNgsSystemGetRequiredMemorySize, sceNgsSystemInit, "System Caveats" in the *NGS Overview*

Document serial number: 000004892117

SceNgsRackDescription

Structure to describe Rack information.

Definition

```
#include <ngs/ngs_top.h>
typedef struct {
        struct SceNgsVoiceDefinition *pVoiceDefn;
        SceInt32 nVoices;
        SceInt32 nChannelsPerVoice;
        SceInt32 nMaxPatchesPerInput;
        SceInt32 nPatchesPerOutput;
        void *pUserReleaseData;
} SceNgsRackDescription;
```

Members

pVoiceDefn	Pointer to Voice Definition information.
nVoices	Maximum number of Voices within the Rack.
nChannelsPerVoice	Maximum number of audio channels for each Voice (1=Mono, 2=Stereo).
nMaxPatchesPerInput	Maximum number of Voices (processed from other Racks) that can be patched to each Voice.
nPatchesPerOutput	Maximum patches from each Voice output.
pUserReleaseData	Pointer to user release data (returned when callback release method used).

Description

Structure to describe Rack information.

Each Rack within the system contains a number of Voices, where each of these Voices is defined using the same Voice Definition information.

See Also

sceNgsRackGetRequiredMemorySize, sceNgsRackInit, sceNgsRackRelease

SceNgsPatchSetupInfo

Structure to describe Patch information.

Definition

```
#include <ngs/ngs_top.h>
typedef struct {
        SceNgsHVoice hVoiceSource;
        SceInt32 nSourceOutputIndex;
        SceInt32 nSourceOutputSubIndex;
        SceNgsHVoice hVoiceDestination;
        SceInt32 nTargetInputIndex;
} SceNgsPatchSetupInfo;
```

Members

hVoiceSource	Source Voice handle.
nSourceOutputIndex	Specifies the output number for the source (0 to n-1, where n = max outputs of the source). So for a Voice with 4 outputs the value (n) is 0 <= n < 4.
nSourceOutputSubIndex	Specifies the sub index when each physical output has multiple patches. Where sub index n is 0 <= n < *SceNgsRackDescription.nPatchesPerOutput*,
hVoiceDestination	Destination Voice handle.
nTargetInputIndex	Destination input index (0 to n-1, where n=max inputs of the target Voice). If there is only one input mixer in the Voice this will always be 0.

Description

Structure to describe Patch information.

Each Rack within the system contains a number of Voices, where each of these Voices is defined using the same Voice Definition information.

When creating a Rack, it is possible to have multiple patches from each physical output. Resulting in the audio data being the same, but patch volume and destination being different. `SCE_NGS_VOICE_PATCH_AUTO_SUBINDEX` automatically finds the first '`sub output`' that is free and assigns the patch. Otherwise, with *nSourceOutputSubIndex*, you can explicitly set the source output sub index.

For example, a Reverb Buss has 1 output and 2 patches per output.

```
Reverb Buss (output 0, sub output 0) -> Delay Buss
Reverb Buss (output 0, sub output 1) -> EQ Buss
```

In this example the data is split 2 ways, with the same reverb signal being sent to both Delay Buss and EQ Buss.

When multiple mixers are present you can specify the destination input index using *nTargetInputIndex*. For example, the side-chain compressor has 2 input mixers, so you specify either 0 or 1 for either the input or control signal.

See Also

`sceNgsPatchCreateRouting`, `sceNgsPatchGetInfo`, `SceNgsRackDescription`

SceNgsVolumeMatrix

Structure to describe the volume levels for each channel to channel connection within a Patch.

Definition

```
#include <ngs/ngs_top.h>
typedef struct {
        SceFloat32
m[SCE_NGS_MAX_SYSTEM_CHANNELS][SCE_NGS_MAX_SYSTEM_CHANNELS];
} SceNgsVolumeMatrix;
```

Members

m	Specifies the values used to define the matrix, [source channels][destination channels]. The volume data is arranged as [s][d]=f. Where: s = source channel (output from voice / input to patch). d = destination channel (input to next voice / output from patch). f = linear volume scalar value (float).

Description

Structure to describe the volume levels for each channel to channel connection within a Patch.

See Also

SceNgsPatchRouteInfo

SceNgsPatchRouteInfo

Structure to describe the information of a Patch.

Definition

```
#include <ngs/ngs_top.h>
typedef struct {
        SceInt32 nOutputChannels;
        SceInt32 nInputChannels;
        SceNgsVolumeMatrix vols;
} SceNgsPatchRouteInfo;
```

Members

nOutputChannels	Number of output channels from source Voice(0-n).
nInputChannels	Number of input channels to destination Voice(0-n).
vols	Volume Matrix for Patch.

Description

Structure to describe the information of a Patch.

Shows the information of a given Patch in the system - the number of inputs and outputs as well as a volume matrix containing each input channel to output channel volume.

See Also

sceNgsPatchGetInfo

SceNgsVoiceInfo

Structure to describe the Voice status and other Voice information.

Definition

```
#include <ngs/ngs_top.h>
typedef struct {
        SceUInt32 uVoiceState;
        SceUInt32 uNumModules;
        SceUInt32 uNumInputs;
        SceUInt32 uNumOutputs;
        SceUInt32 uNumPatchesPerOutput;
        SceUInt32 uUpdateCallsActive;
} SceNgsVoiceInfo;
```

Members

uVoiceState	Bitmask, see SCE_NGS_VOICE_STATE_* defines.
uNumModules	Number of Modules in Voice definition.
uNumInputs	Number of inputs in Voice definition.
uNumOutputs	Number of outputs in Voice definition.
uNumPatchesPerOutput	Number of Patch routes per output.
uUpdateCallsActive	Number of update calls the voice has been active.

Description

Structure to describe the Voice status and other Voice information. Possible values returned in the *uVoiceState* parameter are as follows:

- SCE_NGS_VOICE_STATE_AVAILABLE
- SCE_NGS_VOICE_STATE_ACTIVE
- SCE_NGS_VOICE_STATE_KEY_OFF
- SCE_NGS_VOICE_STATE_FINALIZING
- SCE_NGS_VOICE_STATE_UNLOADING
- SCE_NGS_VOICE_STATE_PENDING
- SCE_NGS_VOICE_STATE_PAUSED

Note that *uVoiceState* is a bitmask, so may return a mixture of these return values.

See Also

sceNgsVoiceGetInfo, "Voice State Transitions" in the *NGS Overview*.

SceNgsCallbackInfo

Structure to describe callback information.

Definition

```
#include <ngs/ngs_top.h>
typedef struct {
        SceNgsHVoice hVoiceHandle;
        SceNgsHRack hRackHandle;
        SceNgsModuleID uModuleID;
        SceInt32 nCallbackData;
        SceInt32 nCallbackData2;
        void *pCallbackPtr;
        void *pUserData;
} SceNgsCallbackInfo;
```

Members

hVoiceHandle	Handle to the Voice that generated the callback.
hRackHandle	Handle to the Rack that generated the callback.
uModuleID	Module ID within the Voice that generated the callback.
nCallbackData	Callback type-specific data.
nCallbackData2	Callback type-specific data.
pCallbackPtr	Callback type-specific data.
pUserData	User data specified on callback registration.

Description

Structure returned with each callback giving information about the callbacks trigger.

SceSulphaNgsConfig

Specifies the NGS agent configuration to calculate memory requirements.

Definition

```
#include <ngs/sulpha_ngs.h>
typedef struct {
        SceUInt32 maxNamedObjects;
        SceUInt32 maxTraceBufferBytes;
} SceSulphaNgsConfig;
```

Members

maxNamedObjects	The maximum number of named objects that can be set.
maxTraceBufferBytes	The maximum size of the message buffer in bytes.

Description

Structure to describe the NGS agent configuration to calculate memory requirements.

The user can obtain the default configuration by calling `sceSulphaNgsGetDefaultConfig()`. This structure should then be passed to the `sceSulphaNgsGetNeededMemory()` function in order to calculate the memory requirements. Finally this structure should be passed to `sceSulphaNgsInit()` in order to initialize the agent.

Specifies the NGS agent configuration to calculate memory requirements. The user can obtain the default configuration by calling `sceSulphaNgsGetDefaultConfig`. This structure should then be passed to the `sceSulphaNgsGetNeededMemory()` function in order to calculate the memory requirements. Finally this structure should be passed to `sceSulphaNgsInit` in order to initialize the agent.

System Functions

sceNgsSystemGetRequiredMemorySize

Query function for the size in bytes required for system initialization.

Definition

```
#include <ngs/ngs_top.h>
SceInt32 sceNgsSystemGetRequiredMemorySize(
        const SceNgsSystemInitParams *pSynthParams,
        SceUInt32 *pnSize
);
```

Calling Conditions

Can be called from an interrupt handler.

Multithread safe.

Arguments

pSynthParams	Pointer to an SceNgsSystemInitParams structure.
pnSize	Pointer to a SceUInt32 to return size if memory required.

Return Values

See SCE_NGS_OK and other NGS return codes for more information.

Description

Returns the memory size (in bytes) required for the system to operate using the information set in an SceNgsSystemInitParams structure. The user must allocate a buffer of this size before they can call the sceNgsSystemInit() function.

See Also

sceNgsSystemInit, sceNgsSystemInit, SceNgsSystemInitParams

sceNgsSystemInit

Initialization function for an NGS system instance.

Definition

```
#include <ngs/ngs_top.h>
SceInt32 sceNgsSystemInit(
        void *pSynthSysMemory,
        const SceUInt32 uMemSize,
        const SceNgsSystemInitParams *pSynthParams,
        SceNgsHSynSystem *pSystemHandle
);
```

Calling Conditions

Can be called from an interrupt handler.

Multithread safe.

Arguments

pSynthSysMemory	Pointer to a system memory buffer allocated by the user.
uMemSize	Size of memory, returned from sceNgsSystemGetRequiredMemorySize().
pSynthParams	Pointer to a SceNgsSystemInitParams structure.
pSystemHandle	Pointer to a SceNgsHSynSystem handle which returns the synth system handle.

Return Values

See SCE_NGS_OK and other NGS return codes for more information.

Description

Initializes the NGS system using the information set in an SceNgsSystemInitParams structure. Note that only one instance is permitted per system process.

See Also

sceNgsSystemGetRequiredMemorySize

sceNgsSystemUpdate

Updates the NGS system.

Definition

```
#include <ngs/ngs_top.h>
SceInt32 sceNgsSystemUpdate(
        SceNgsHSynSystem hSystemHandle
);
```

Calling Conditions

Can be called from an interrupt handler.

Multithread safe.

Arguments

hSystemHandle System handle, returned from sceNgsSystemInit().

Return Values

See SCE_NGS_OK and other NGS return codes for more information.

Description

Updates the NGS system. This will process all active Voices within all Racks, causing a packet of PCM data to be generated. Callbacks are called from this function, after voice processing and only after the full procedure is complete.

Note that this is a blocking function and should be called from a separate thread to the gameplay engine.

See Also

sceNgsSystemInit

sceNgsSystemRelease

Requests for the system to be released.

Definition

```
#include <ngs/ngs_top.h>
SceInt32 sceNgsSystemRelease(
        SceNgsHSynSystem hSystemHandle
);
```

Calling Conditions

Can be called from an interrupt handler.

Multithread safe.

Arguments

hSystemHandle	System handle, returned from sceNgsSystemInit().

Return Values

See SCE_NGS_OK and other NGS return codes for more information.

Description

Requests for the system to be released. Releasing the system will stop audio generation of the NGS system. Any racks still present in the system will have their associated release functions called. Once released, it is possible for the user to free any allocated system memory. This function will return only after sceNgsSystemUpdate() has finished being processed.

See Also

sceNgsSystemInit, sceNgsSystemUpdate

sceNgsSystemLock

Requests the system is locked.

Definition

```
#include <ngs/ngs_top.h>
SceInt32 sceNgsSystemLock(
        SceNgsHSynSystem hSystemHandle
);
```

Calling Conditions

Can be called from an interrupt handler.

Multithread safe.

Arguments

hSystemHandle System handle, returned from sceNgsSystemInit().

Return Values

See SCE_NGS_OK and other NGS return codes for more information.

Description

Requests the system is locked. Any changes made to Voice parameters and similar objects will not be registered until an unlock; however, the system will still be updated (sceNgsSystemUpdate()) during a SystemLock period. This allows for synchronization across the NGS system (where pausing Voices, starting multiple Voices or modifying the frequency of multiple Voices could otherwise be problematic). Note that this function returns straight away unless the system is already locked. In which case, it will wait until it is unlocked

See Also

sceNgsSystemInit, sceNgsSystemUnlock, sceNgsSystemUpdate

sceNgsSystemUnlock

Unlocks the system.

Definition

```
#include <ngs/ngs_top.h>
SceInt32 sceNgsSystemUnlock(
        SceNgsHSynSystem hSystemHandle
);
```

Calling Conditions

Can be called from an interrupt handler.

Multithread safe.

Arguments

hSystemHandle	System handle, returned from sceNgsSystemInit().

Return Values

See SCE_NGS_OK and other NGS return codes for more information.

Description

Attempts to unlock the system. Any changes made to Voice parameters during a system lock will be processed on the next NGS update.

See Also

sceNgsSystemInit, sceNgsSystemLock, sceNgsSystemUpdate

sceNgsSystemSetParamErrorCallback

Permits the user to setup a system level error callback.

Definition

```
#include <ngs/ngs_top.h>
SceInt32 sceNgsSystemSetParamErrorCallback(
        SceNgsHSynSystem hSystemHandle,
        const SceNgsParamsErrorCallbackFunc callbackFuncPtr
);
```

Calling Conditions

Can be called from an interrupt handler.

Multithread safe.

Arguments

hSystemHandle	System handle, returned from sceNgsSystemInit().
callbackFuncPtr	Callback function.

Return Values

See SCE_NGS_OK and other NGS return codes for more information.

Description

Permits the user to setup a system level error callback. If an error is returned during the asynchronous processing of a Module's parameter handling, a callback is registered. The callbacks are batched and called at the end of sceNgsSystemUpdate(). If the Rack from which the error callback originated does not have its own error callback, the system error callback will be called, if one has been registered by this function.

See Also

sceNgsSystemInit, sceNgsSystemUpdate, sceNgsRackSetParamErrorCallback

sceNgsSystemSetFlags

Sets the mode for module parameter checking.

Definition

```
#include <ngs/ngs_top.h>
SceInt32 sceNgsSystemSetFlags(
        SceNgsHSynSystem hSystemHandle,
        const SceUInt32 uSystemFlags
);
```

Calling Conditions

Can be called from an interrupt handler.

Multithread safe.

Arguments

hSystemHandle	System handle, returned from sceNgsSystemInit().
uSystemFlags	Mode for module parameter checking using bitwise system flags SCE_NGS_SYSTEM_FLAG_*.

Return Values

See SCE_NGS_OK and other NGS return codes for more information.

Description

Sets the mode for module parameter checking using the bitwise system flags.
Each sceNgsVoiceUnlockParams() and sceNgsVoiceSetParamsBlock() call will validate the parameter ranges according to the module parameter checking mode.

The system flags, currently supported, are:

- SCE_NGS_SYSTEM_FLAG_NO_MODULE_PARAM_CHECKING,
- SCE_NGS_SYSTEM_FLAG_MODULE_PARAM_CHECKING or
- SCE_NGS_SYSTEM_FLAG_DEFAULT.

See Also

sceNgsSystemInit

Module Command Functions

sceNgsModuleGetNumPresets

Returns the number of presets for the Module.

Definition

```
#include <ngs/ngs_top.h>
SceInt32 sceNgsModuleGetNumPresets(
        SceNgsHSynSystem hSystemHandle,
        const SceNgsModuleID uModuleID,
        SceUInt32 *puNumPresets
);
```

Calling Conditions

Can be called from an interrupt handler.

Multithread safe.

Arguments

hSystemHandle	System handle, returned from sceNgsSystemInit().
uModuleID	The ID of the Module.
puNumPresets	Pointer to the return value for the number of Module presets.

Return Values

See SCE_NGS_OK and other NGS return codes for more information.

Description

Returns the number of presets for the Module.

See Also

sceNgsSystemInit

Document serial number: 000004892117

sceNgsModuleGetPreset

Returns the indexed preset into the supplied buffer info (pointer and size).

Definition

```
#include <ngs/ngs_top.h>
SceInt32 sceNgsModuleGetPreset(
        SceNgsHSynSystem hSystemHandle,
        const SceNgsModuleID uModuleID,
        const SceUInt32 uPresetIndex,
        SceNgsBufferInfo *pParamsBuffer
);
```

Calling Conditions

Can be called from an interrupt handler.

Multithread safe.

Arguments

hSystemHandle	System handle, returned from sceNgsSystemInit().
uModuleID	The ID of the Module.
uPresetIndex	Index of the preset value for this Module.
pParamsBuffer	Pointer to output buffer where the preset data will be written.

Return Values

See SCE_NGS_OK and other NGS return codes for more information.

Description

Returns the indexed preset into the supplied buffer info (pointer and size). Note, the function will return an error if the supplied buffer has insufficient size.

See Also

sceNgsSystemInit

Rack Command Functions

sceNgsRackGetRequiredMemorySize

Query function for the size in bytes required to initialize a rack.

Definition

```
#include <ngs/ngs_top.h>
SceInt32 sceNgsRackGetRequiredMemorySize(
        SceNgsHSynSystem hSystemHandle,
        const SceNgsRackDescription *pRackDesc,
        SceUInt32 *pnSize
);
```

Calling Conditions

Can be called from an interrupt handler.

Multithread safe.

Arguments

hSystemHandle	System handle, returned from sceNgsSystemInit().
pRackDesc	Pointer to an SceNgsRackDescription structure.
pnSize	Pointer to a SceUInt32 member, where the required memory buffer size (in bytes) is written.

Return Values

See SCE_NGS_OK and other NGS return codes for more information.

Description

Calculates the memory size (in bytes) that the user needs to allocate and pass into sceNgsRackInit() for the initialization of a Rack.

See Also

sceNgsRackInit, sceNgsSystemInit

Document serial number: 000004892117

sceNgsRackInit

Initializes a Rack.

Definition

```
#include <ngs/ngs_top.h>
SceInt32 sceNgsRackInit(
        SceNgsHSynSystem hSystemHandle,
        SceNgsBufferInfo *pRackBuffer,
        const SceNgsRackDescription *pRackDesc,
        SceNgsHRack *pRackHandle
);
```

Calling Conditions

Can be called from an interrupt handler.

Multithread safe.

Arguments

hSystemHandle	System handle, returned from sceNgsSystemInit().
pRackBuffer	Pointer to buffer information, specifying the allocated memory area and the size of that area.
pRackDesc	Pointer to an SceNgsRackDescription structure.
pRackHandle	Pointer to returned rack handle.

Return Values

See SCE_NGS_OK and other NGS return codes for more information.

Description

Initializes a Rack.

See Also

sceNgsRackGetRequiredMemorySize, sceNgsSystemInit, SceNgsRackDescription

sceNgsRackGetVoiceHandle

Returns a Voice handle from a Rack.

Definition

```
#include <ngs/ngs_top.h>
SceInt32 sceNgsRackGetVoiceHandle(
        SceNgsHRack hRackHandle,
        const SceUInt32 uIndex,
        SceNgsHVoice *pVoiceHandle
);
```

Calling Conditions

Can be called from an interrupt handler.

Multithread safe.

Arguments

hRackHandle	Rack handle, returned from sceNgsRackInit().
uIndex	Index of Voice within the Rack, 0 to (number of Voices in Rack-1).
pVoiceHandle	Pointer to a SceNgsHVoice handle where the handle will be returned.

Return Values

See SCE_NGS_OK and other NGS return codes for more information.

Description

Returns a Voice handle from a Rack. The Voice handle can then be used to play audio data, or perform other Voice commands.

See Also

sceNgsRackInit, sceNgsVoiceInit and other SceNgsVoice* commands.

sceNgsRackRelease

Function to release a Rack from the NGS System.

Definition

```
#include <ngs/ngs_top.h>
SceInt32 sceNgsRackRelease(
        SceNgsHRack hRackHandle,
        const SceNgsRackReleaseCallbackFunc callbackFuncPtr
);
```

Calling Conditions

Can be called from an interrupt handler.

Multithread safe.

Arguments

hRackHandle	Rack handle, returned from sceNgsRackInit().
callbackFuncPtr	Callback function called by sceNgsSystemUpdate().

Return Values

See SCE_NGS_OK and other NGS return codes for more information.

Description

Safely releases a Rack from the NGS System. If the user wishes to asynchronously unload, then a callback should be supplied; otherwise, if *FuncPtr* is NULL, the call to sceNgsRackRelease will be blocked until the Rack has been removed from the system and a system update (if active) has finished. Only when the Rack has safely been released is it OK for the user to free any allocated memory.

In the case of an asynchronous release, the user will be required to call sceNgsSystemUpdate() in order to generate the callback.

Note that this function stops all playing voices and removes all patches to and from the rack.

See Also

sceNgsRackInit, sceNgsSystemUpdate

sceNgsRackSetParamErrorCallback

Sets up an error callback for the Modules within the Rack.

Definition

```
#include <ngs/ngs_top.h>
SceInt32 sceNgsRackSetParamErrorCallback(
        SceNgsHRack hRackHandle,
        const SceNgsParamsErrorCallbackFunc callbackFuncPtr
);
```

Calling Conditions

Can be called from an interrupt handler.

Multithread safe.

Arguments

hRackHandle	Rack handle, returned from sceNgsRackInit().
callbackFuncPtr	Pointer to callback function, or pass SCE_NGS_NO_CALLBACK if no callback is required.

Return Values

See SCE_NGS_OK and other NGS return codes for more information.

Description

Sets up an error callback for the Modules within the Rack. If an error is returned during the asynchronous processing of a Module's parameter handling, then a callback is registered. The callbacks are batched and called at the end of sceNgsSystemUpdate(). If no callback is registered for the rack, the parameter error will pass to the system level param error callback (if registered), see sceNgsSystemSetParamErrorCallback().

See Also

sceNgsSystemSetParamErrorCallback, SceNgsCallbackInfo, sceNgsRackInit, sceNgsSystemUpdate

Voice Command Functions

sceNgsVoiceInit

Initializes a Voice.

Definition

```
#include <ngs/ngs_top.h>
SceInt32 sceNgsVoiceInit(
        SceNgsHVoice hVoiceHandle,
        const SceNgsVoicePreset *pPreset,
        const SceUInt32 uInitFlags
);
```

Calling Conditions

Can be called from an interrupt handler.

Multithread safe.

Arguments

hVoiceHandle	Voice Handle (see sceNgsRackGetVoiceHandle()).
pPreset	The preset to initialize the Voice with or NULL.
uInitFlags	See SCE_NGS_VOICE_INIT_BASE and other SCE_NGS_VOICE_INIT* values.

Return Values

See SCE_NGS_OK and other NGS return codes for more information.

Description

If *pPreset* is NULL the voice will attempt to use the default voice preset (0), if this is not present then each module will initialize with the default module preset (0).

Note, a voice must not be in the active mode when attempting to initialize.

See Also

sceNgsRackGetVoiceHandle, SceNgsVoicePreset, "Voice State Transitions" in the *NGS Overview*.

Document serial number: 000004892117

sceNgsVoicePlay

Starts the playback of a Voice.

Definition

```
#include <ngs/ngs_top.h>
SceInt32 sceNgsVoicePlay(
        SceNgsHVoice hVoiceHandle
);
```

Calling Conditions

Can be called from an interrupt handler.

Multithread safe.

Arguments

hVoiceHandle	Voice Handle (see sceNgsRackGetVoiceHandle()).

Return Values

See SCE_NGS_OK and other NGS return codes for more information.

Description

Starts playing a Voice. The Voice must first be initialized before an initial call to `sceNgsVoicePlay`. Refer to the *Voice State Transitions* section in the *NGS Overview* document for a diagram of voice state changes.

See Also

sceNgsRackGetVoiceHandle, "Voice State Transitions" in the *NGS Overview*.

sceNgsVoiceKeyOff

Changes the status of a Voice to a Key OFF state.

Definition

```
#include <ngs/ngs_top.h>
SceInt32 sceNgsVoiceKeyOff(
        SceNgsHVoice hVoiceHandle
);
```

Calling Conditions

Can be called from an interrupt handler.

Multithread safe.

Arguments

hVoiceHandle	Voice Handle (see sceNgsRackGetVoiceHandle()).

Return Values

See SCE_NGS_OK and other NGS return codes for more information.

Description

Changes the status of a Voice to a Key OFF state. Various Modules perform actions within a Key OFF state. For the Amplitude Envelope Module this will trigger a release phase (see Envelope Module information for more detail). If no Modules within a Voice recognize the Key OFF state it will kill the Voice immediately.

If your desired operation is to stop a Voice, use sceNgsVoiceKill().

See Also

sceNgsRackGetVoiceHandle, sceNgsVoiceKill, "Voice State Transitions" in the *NGS Overview*.

sceNgsVoiceKill

Stops a Voice playing.

Definition

```
#include <ngs/ngs_top.h>
SceInt32 sceNgsVoiceKill(
        SceNgsHVoice hVoiceHandle
);
```

Calling Conditions

Can be called from an interrupt handler.

Multithread safe.

Arguments

hVoiceHandle	Voice Handle (see sceNgsRackGetVoiceHandle()).

Return Values

See SCE_NGS_OK and other NGS return codes for more information.

Description

Stops a Voice playing.

See Also

sceNgsRackGetVoiceHandle, sceNgsVoicePlay, "Voice State Transitions" in the *NGS Overview*.

sceNgsVoicePause

Pauses a playing Voice.

Definition

```
#include <ngs/ngs_top.h>
SceInt32 sceNgsVoicePause(
        SceNgsHVoice hVoiceHandle
);
```

Calling Conditions

Can be called from an interrupt handler.

Multithread safe.

Arguments

hVoiceHandle	Voice Handle (see sceNgsRackGetVoiceHandle()).

Return Values

See SCE_NGS_OK and other NGS return codes for more information.

Description

Pauses a playing Voice. Pausing a Voice causes all processing within the Voice to stop until the sceNgsVoiceResume() function is called.

Note that multiple Voices can be synchronized when paused if using the sceNgsSystemLock() and sceNgsSystemUnlock() commands.

See Also

sceNgsRackGetVoiceHandle, sceNgsVoicePlay, sceNgsVoiceResume, "Voice State Transitions" in the *NGS Overview*.

sceNgsVoiceResume

Resumes a paused a Voice.

Definition

```
#include <ngs/ngs_top.h>
SceInt32 sceNgsVoiceResume(
        SceNgsHVoice hVoiceHandle
);
```

Calling Conditions

Can be called from an interrupt handler.

Multithread safe.

Arguments

hVoiceHandle	Voice Handle (see sceNgsRackGetVoiceHandle()).

Return Values

See SCE_NGS_OK and other NGS return codes for more information.

Description

Resumes a paused a Voice. Resuming a Voice causes all processing within the Voice to resume from the point it was previously paused.

Note that multiple Voices can be synchronised when paused if using the sceNgsSystemLock() and sceNgsSystemUnlock() commands.

See Also

sceNgsRackGetVoiceHandle, sceNgsVoicePause, sceNgsVoicePlay, "Voice State Transitions" in the *NGS Overview*.

sceNgsVoiceSetPreset

Sets a Voice with preset values.

Definition

```
#include <ngs/ngs_top.h>
SceInt32 sceNgsVoiceSetPreset(
        SceNgsHVoice hVoiceHandle,
        const SceNgsVoicePreset *pVoicePreset
);
```

Calling Conditions

Can be called from an interrupt handler.

Multithread safe.

Arguments

hVoiceHandle	Voice Handle (see sceNgsRackGetVoiceHandle()).
pVoicePreset	The preset values for each Module within a Voice to be initialized with.

Return Values

See SCE_NGS_OK and other NGS return codes for more information.

Description

Sets a Voice with preset values. This can be called while a Voice is playing.

See Also

sceNgsRackGetVoiceHandle, SceNgsVoicePreset

sceNgsVoiceLockParams

Returns a structure containing a module's parameters so that they can be modified by the user.

Definition

```
#include <ngs/ngs_top.h>
SceInt32 sceNgsVoiceLockParams(
        SceNgsHVoice hVoiceHandle,
        const SceUInt32 uModule,
        const SceNgsParamsID uParamsInterfaceId,
        SceNgsBufferInfo *pParamsBuffer
);
```

Calling Conditions

Can be called from an interrupt handler.

Multithread safe.

Arguments

hVoiceHandle	Handle of a Voice.
uModule	Index to required Voice's Module.
uParamsInterfaceId	Parameter Structure ID, see the relevant Module header file.
pParamsBuffer	Pointer to buffer structure where the address and size of the parameter block will be returned.

Return Values

See SCE_NGS_OK and other NGS return codes for more information.

Description

This function returns a structure containing the size and address of the module parameters for the specified voice. The function also locks those parameters therefore allowing only one thread at a time to access and edit the parameters. The NGS System will continue to use the previous set of parameters until such time as sceNgsVoiceUnlockParams() is called for the given parameter block.

See Also

sceNgsVoiceUnlockParams

sceNgsVoiceUnlockParams

Unlocks a Voice's Module from being previously locked.

Definition

```
#include <ngs/ngs_top.h>
SceInt32 sceNgsVoiceUnlockParams(
        SceNgsHVoice hVoiceHandle,
        const SceUInt32 uModule
);
```

Calling Conditions

Can be called from an interrupt handler.

Multithread safe.

Arguments

hVoiceHandle	Handle of a Voice.
uModule	Index to required Voice's Module.

Return Values

See SCE_NGS_OK and other NGS return codes for more information.

Description

Unlocks a Voice's Module from being previously locked. By unlocking a Voice, any modifications to the Module data will then take effect during the next NGS System update. This ensures that an update does not take place mid-way through the user modifying Module parameters.

See Also

sceNgsVoiceLockParams, sceNgsSystemUpdate

sceNgsVoiceSetParamsBlock

Allows the user to set the Voice parameters for one or more Modules.

Definition

```
#include <ngs/ngs_top.h>
SceInt32 sceNgsVoiceSetParamsBlock(
        SceNgsHVoice hVoiceHandle,
        const SceNgsModuleParamHeader *pParamData,
        const SceUInt32 uSize,
        SceInt32 *pnErrorCount
);
```

Calling Conditions

Can be called from an interrupt handler.

Multithread safe.

Arguments

hVoiceHandle	Handle of a Voice.
pParamData	Address of parameter data block.
uSize	Size of parameter data block.
pnErrorCount	Pointer to error counter, returns number of Module setup errors.

Return Values

See SCE_NGS_OK and other NGS return codes for more information.

Description

This function provides an alternative method to parameter setting to `sceNgsVoiceLock()` and `sceNgsVoiceUnlock()`. Allows the user to set the Voice parameters for one or more Modules. Each parameter block should be preceded by an SceNgsModuleParamHeader. To setup multiple Modules pass to the function a continuous block of memory containing the concatenated block parameter interfaces.

Note that the parameter structures must not be locked whilst attempting to set them up; otherwise, an error code will be generated.

See Also

sceNgsRackGetVoiceHandle, sceNgsVoiceLockParams, sceNgsVoiceUnlockParams

sceNgsVoiceBypassModule

Causes a Voice's Module to be processed or bypassed.

Definition

```
#include <ngs/ngs_top.h>
SceInt32 sceNgsVoiceBypassModule(
        SceNgsHVoice hVoiceHandle,
        const SceUInt32 uModule,
        const SceUInt32 uBypassFlag
);
```

Calling Conditions

Can be called from an interrupt handler.

Multithread safe.

Arguments

hVoiceHandle	Handle of a Voice.
uModule	Index to required Voice's Module.
uBypassFlag	Flag: SCE_NGS_MODULE_FLAG_BYPASSED or SCE_NGS_MODULE_FLAG_NOT_BYPASSED.

Return Values

See SCE_NGS_OK and other NGS return codes for more information.

Description

Causes a Voice's Module to be processed or bypassed. If bypassed, each Module may process this case differently.

For example:

a) Many Modules would copy the input audio signal from a Module to its output buffer (bypassing the processing in between);

b) Some Modules may do (a), but also still process the data as if it were going to be output (this allows for Modules such as Filters or Reverbs to act correctly when they are no longer bypassed).

sceNgsVoiceSetModuleCallback

Sets up a callback for a specific Module within a given Voice.

Definition

```
#include <ngs/ngs_top.h>
SceInt32 sceNgsVoiceSetModuleCallback(
        SceNgsHVoice hVoiceHandle,
        const SceUInt32 uModule,
        const SceNgsModuleCallbackFunc callbackFuncPtr,
        void *pUserData
);
```

Calling Conditions

Can be called from an interrupt handler.

Multithread safe.

Arguments

hVoiceHandle	Handle of a Voice.
uModule	Index to required Voice's Module.
callbackFuncPtr	Function pointer to callback or SCE_NGS_NO_CALLBACK to disable callback on this Module.
pUserData	Pointer returned in *pUserData* field of SceNgsCallbackInfo structure supplied to callback.

Return Values

See SCE_NGS_OK and other NGS return codes for more information.

Description

Sets up a callback for a specific Module within a given Voice.

The Module's callback behavior is defined by the Module type.

See Also

SceNgsCallbackInfo

sceNgsVoiceSetFinishedCallback

Sets up a callback for a specific Voice.

Definition

```
#include <ngs/ngs_top.h>
SceInt32 sceNgsVoiceSetFinishedCallback(
        SceNgsHVoice hVoiceHandle,
        const SceNgsCallbackFunc callbackFuncPtr,
        void *pUserData
);
```

Calling Conditions

Can be called from an interrupt handler.

Multithread safe.

Arguments

hVoiceHandle	Handle of a Voice.
callbackFuncPtr	Function pointer to callback or SCE_NGS_NO_CALLBACK to disable callback on this Module.
pUserData	Pointer returned in *pUserData* field of SceNgsCallbackInfo structure supplied to callback.

Return Values

See SCE_NGS_OK and other NGS return codes for more information.

Description

Sets up a callback which is triggered once the Voice transitions to stopped from a finalizing (note-off) condition via sceNgsVoiceKeyOff(), the callback is not generated if the voice is immediately stopped via sceNgsVoiceKill().

The callback will also be generated if a one-shot sound effect finishes playback.

See Also

SceNgsCallbackInfo, sceNgsVoiceKeyOff, sceNgsVoiceKill, “Voice State Transitions” in the *NGS Overview*.

sceNgsVoiceGetStateData

Retrieves the user state data for a Voice's Module.

Definition

```
#include <ngs/ngs_top.h>
SceInt32 sceNgsVoiceGetStateData(
        SceNgsHVoice hVoiceHandle,
        const SceUInt32 uModule,
        void *pMem,
        const SceUInt32 uMemSize
);
```

Calling Conditions

Can be called from an interrupt handler.

Multithread safe.

Arguments

hVoiceHandle	Handle of a Voice.
uModule	Index to required Voice's Module.
pMem	Address into which the returned state data will be written.
uMemSize	Size of the return data buffer specified by *pMem*.

Return Values

See SCE_NGS_OK and other NGS return codes for more information.

Description

Retrieves the user state data for a Voice's Module. This should not be confused with the Voice Param Data. A Voice's Module may have the ability to output information that the user would need to access. For example, for the PCM player Module, it is possible to retrieve information such as the current playback position (see SceNgsPlayerStates in the *NGS Modules Reference* for more information). The sceNgsVoiceGetStateData() function can then be used to retrieve such information.

See Also

SceNgsPlayerStates

sceNgsVoiceGetInfo

Retrieves the current status and other information about a Voice.

Definition

```
#include <ngs/ngs_top.h>
SceInt32 sceNgsVoiceGetInfo(
        SceNgsHVoice hVoiceHandle,
        SceNgsVoiceInfo *pInfo
);
```

Calling Conditions

Can be called from an interrupt handler.

Multithread safe.

Arguments

hVoiceHandle	Handle of a Voice.
pInfo	See SceNgsVoiceInfo structure.

Return Values

See SCE_NGS_OK and other NGS return codes for more information.

Description

Retrieves the current status and other information about a Voice.

See Also

SceNgsVoiceInfo, "Voice State Transitions" in the *NGS Overview*.

sceNgsVoiceGetModuleType

Returns the type of the Voice Module.

Definition

```
#include <ngs/ngs_top.h>
SceInt32 sceNgsVoiceGetModuleType(
        SceNgsHVoice hVoiceHandle,
        const SceUInt32 uModule,
        SceNgsModuleID *pModuleType
);
```

Calling Conditions

Can be called from an interrupt handler.

Multithread safe.

Arguments

hVoiceHandle	Handle of a Voice.
uModule	Index of the Voice Module.
pModuleType	Pointer to a Module type structure.

Return Values

See SCE_NGS_OK and other NGS return codes for more information.

Description

Pointer to a SceNgsModuleID value returned by the function. (See the module ID defines list in the *NGS Modules Reference* for more information.)

sceNgsVoiceGetModuleBypass

Returns the bypass state of the given Voice Module.

Definition

```
#include <ngs/ngs_top.h>
SceInt32 sceNgsVoiceGetModuleBypass(
        SceNgsHVoice hVoiceHandle,
        const SceUInt32 uModule,
        SceUInt32 *puBypassFlag
);
```

Calling Conditions

Can be called from an interrupt handler.

Multithread safe.

Arguments

hVoiceHandle	Handle of a Voice.
uModule	Index of the Voice Module.
puBypassFlag	Pointer to the returned bypass state.

Return Values

See SCE_NGS_OK and other NGS return codes for more information.

Description

Returns the bypass state of the given Voice Module.

sceNgsVoiceGetParamsOutOfRange

Returns a text string giving debug information.

Definition

```
#include <ngs/ngs_top.h>
SceInt32 sceNgsVoiceGetParamsOutOfRange(
        SceNgsHVoice hVoiceHandle,
        const SceUInt32 uModule,
        char *pszMessageBuffer
);
```

Calling Conditions

Can be called from an interrupt handler.

Multithread safe.

Arguments

hVoiceHandle	Handle of a Voice.
uModule	Index of the Voice Module.
pszMessageBuffer	Pointer to the address of a minimum 128 char buffer to return debug information.

Return Values

See SCE_NGS_OK and other NGS return codes for more information.

Description

Returns a text string giving further debug information after the user has received a SCE_NGS_PARAM_OUT_OF_RANGE return value, detailing which parameter was at fault.

See Also

sceNgsVoiceUnlockParams, sceNgsVoiceSetParamsBlock

sceNgsVoiceGetOutputPatch

Returns a patch Handle from a Voice.

Definition

```
#include <ngs/ngs_top.h>
SceInt32 sceNgsVoiceGetOutputPatch(
        SceNgsHVoice hVoiceHandle,
        const SceInt32 nOutputIndex,
        const SceInt32 nSubIndex,
        SceNgsHPatch *pPatchHandle
);
```

Calling Conditions

Can be called from an interrupt handler.

Multithread safe.

Arguments

hVoiceHandle	Handle of a Voice.
nOutputIndex	Index of the output you wish to specify.
nSubIndex	Sub-index of the output you wish to specify.
pPatchHandle	Address of `SceNgsHPatch` value returned by the function.

Return Values

See `SCE_NGS_OK` and other NGS return codes for more information.

Description

Returns a patch Handle from a Voice. A Patch is a connection between a Voice and another Voice in another Rack. This Handle can then be used to remove the connection or edit the volumes.

See Also

`sceNgsVoicePatchSetVolume`, `sceNgsVoicePatchSetVolumes`, `sceNgsVoicePatchSetVolumesMatrix`

sceNgsVoicePatchSetVolume

Modifies the volumes within an existing patch.

Definition

```
#include <ngs/ngs_top.h>
SceInt32 sceNgsVoicePatchSetVolume(
        SceNgsHPatch hPatchHandle,
        const SceInt32 nOutputChannel,
        const SceInt32 nInputChannel,
        const SceFloat32 fVol
);
```

Calling Conditions

Can be called from an interrupt handler.

Multithread safe.

Arguments

hPatchHandle	Handle to be obtained from system.
nOutputChannel	Source Voice audio output channel number.
nInputChannel	Destination Voice audio input channel number.
fVol	Volume scalar value.

Return Values

See SCE_NGS_OK and other NGS return codes for more information.

Description

Sets the patch volume of a specified patch between two Voices. The volume is a scalar value (1.0 = destination Voice receives 100% of source signal, 0.5f = destination Voice receives 50% of source signal, 0.0f = silence).

Note that there are no maximum or minimum values for volume. Using a negative value will invert the phase of the PCM signal.

See Also

sceNgsVoiceGetOutputPatch, sceNgsPatchCreateRouting

sceNgsVoicePatchSetVolumes

Modifies the volumes within an existing patch.

Definition

```
#include <ngs/ngs_top.h>
SceInt32 sceNgsVoicePatchSetVolumes(
        SceNgsHPatch hPatchHandle,
        const SceInt32 nOutputChannel,
        const SceFloat32 *pVolumes,
        const SceInt32 nVols
);
```

Calling Conditions

Can be called from an interrupt handler.

Multithread safe.

Arguments

hPatchHandle	Handle to be obtained from system.
nOutputChannel	Source Voice audio output channel number.
pVolumes	Pointer to a list of volume scalar values.
nVols	Number of Destination Voice input channels set (0-n).

Return Values

See SCE_NGS_OK and other NGS return codes for more information.

Description

Sets the patch volume of a specified patch between two Voices. The volume is a scalar value (1.0 = destination voice receives 100% of source signal, 0.5f = destination voice receives 50% of source signal, 0.0f = silence).

Note that there are no maximum or minimum values for volume. Using a negative value will invert the phase of the PCM signal.

See Also

sceNgsVoiceGetOutputPatch, sceNgsPatchCreateRouting

sceNgsVoicePatchSetVolumesMatrix

Modifies the volumes within an existing patch.

Definition

```
#include <ngs/ngs_top.h>
SceInt32 sceNgsVoicePatchSetVolumesMatrix(
        SceNgsHPatch hPatchHandle,
        const SceNgsVolumeMatrix *pMatrix
);
```

Calling Conditions

Can be called from an interrupt handler.

Multithread safe.

Arguments

hPatchHandle	Handle to be obtained from system.
pMatrix	Pointer to SceNgsVolumeMatrix structure containing volume data. The volume data is arranged as [s][d]=f. Where: s = source channel (output from voice / input to patch). d = destination channel (input to next voice / output from patch). f = linear volume scalar value (float).

Return Values

See SCE_NGS_OK and other NGS return codes for more information.

Description

Sets the patch volume of a specified patch between two Voices using an SceNgsVolumeMatrix structure. This allows n*n volumes to be set at the same time.

The volume is a scalar value (1.0 = destination voice receives 100% of source signal, 0.5f = destination voice receives 50% of source signal, 0.0f = silence).

Note that there are no maximum or minimum values for volume. Using a negative value will invert the phase of the PCM signal.

See Also

sceNgsVoiceGetOutputPatch, sceNgsPatchCreateRouting

Voice Definition Functions

sceNgsVoiceDefGetCompressorBuss

Returns a pointer to the compressor buss Voice Definition.

Definition

```
#include <ngs/templates/compressor_buss_voice.h>
struct SceNgsVoiceDefinition *sceNgsVoiceDefGetCompressorBuss(void);
```

Calling Conditions

Can be called from an interrupt handler.

Multithread safe.

Arguments

None

Return Values

A Voice Definition structure, `SceNgsVoiceDefinition`.

Description

Returns a pointer to the compressor buss Voice Definition. This Voice applies a compressor effect to Voices patched into it.

See Also

`SceNgsRackDescription`

sceNgsVoiceDefGetCompressorSideChainBuss

Returns a pointer to a side chain compressor buss Voice Definition.

Definition

```
#include <ngs/templates/compressor_side_chain_buss_voice.h>
struct SceNgsVoiceDefinition *sceNgsVoiceDefGetCompressorSideChainBuss(void);
```

Calling Conditions

Can be called from an interrupt handler.

Multithread safe.

Arguments

None

Return Values

A Voice Definition structure, `SceNgsVoiceDefinition`.

Description

Returns a pointer to a side chain compressor buss Voice Definition. This Voice type applies a compressor effect to the Voices patched into input 0. The amount of compression is controlled by a second signal which should be patched into input1.

See Also

`SceNgsRackDescription`

sceNgsVoiceDefGetDelayBuss

Returns a pointer to the delay buss Voice Definition.

Definition

```
#include <ngs/templates/delay_buss_voice.h>
struct SceNgsVoiceDefinition *sceNgsVoiceDefGetDelayBuss(void);
```

Calling Conditions

Can be called from an interrupt handler.

Multithread safe.

Arguments

None

Return Values

A Voice Definition structure, `SceNgsVoiceDefinition`.

Description

Returns a pointer to the delay buss Voice Definition. This Voice type applies a delay effect to Voices patched into it.

See Also

`SceNgsRackDescription`

sceNgsVoiceDefGetDistortionBuss

Returns a pointer to the distortion buss Voice Definition.

Definition

```
#include <ngs/templates/distortion_buss_voice.h>
struct SceNgsVoiceDefinition *sceNgsVoiceDefGetDistortionBuss(void);
```

Calling Conditions

Can be called from an interrupt handler.

Multithread safe.

Arguments

None

Return Values

A Voice Definition structure, `SceNgsVoiceDefinition`.

Description

Returns a pointer to the distortion buss Voice Definition. This Voice type applies a distortion effect to Voices patched into it.

See Also

`SceNgsRackDescription`

sceNgsVoiceDefGetEnvelopeBuss

Returns a pointer to the envelope buss Voice Definition.

Definition

```
#include <ngs/templates/env_buss_voice.h>
struct SceNgsVoiceDefinition *sceNgsVoiceDefGetEnvelopeBuss(void);
```

Calling Conditions

Can be called from an interrupt handler.

Multithread safe.

Arguments

None

Return Values

A Voice Definition structure, `SceNgsVoiceDefinition`.

Description

Returns a pointer to the envelope buss Voice Definition. This Voice type applies an envelope effect to Voices patched into it.

See Also

`SceNgsRackDescription`

sceNgsVoiceDefGetEqBuss

Returns a pointer to the equalization buss Voice Definition.

Definition

```
#include <ngs/templates/eq_buss_voice.h>
struct SceNgsVoiceDefinition *sceNgsVoiceDefGetEqBuss(void);
```

Calling Conditions

Can be called from an interrupt handler.

Multithread safe.

Arguments

None

Return Values

A Voice Definition structure, `SceNgsVoiceDefinition`.

Description

Returns a pointer to the equalization buss Voice Definition. This Voice type applies an equalization effect to Voices patched into it via a bank of 4 filters connected in serial.

See Also

`SceNgsRackDescription`

sceNgsVoiceDefGetMasterBuss

Returns a pointer to the master buss Voice Definition.

Definition

```
#include <ngs/templates/master_buss_voice.h>
struct SceNgsVoiceDefinition *sceNgsVoiceDefGetMasterBuss(void);
```

Calling Conditions

Can be called from an interrupt handler.

Multithread safe.

Arguments

None

Return Values

A Voice Definition structure, `SceNgsVoiceDefinition`.

Description

Returns a pointer to the master buss Voice Definition. This Voice type allows the user to get the output PCM audio data from the system by outputting the Voices patched into it. However, the audio can't be output from the Module to other Modules.

See Also

SceNgsRackDescription

sceNgsVoiceDefGetMixerBuss

Returns a pointer to the mixer buss Voice Definition.

Definition

```
#include <ngs/templates/mixer_buss_voice.h>
&struct SceNgsVoiceDefinition *sceNgsVoiceDefGetMixerBuss(void);
```

Calling Conditions

Can be called from an interrupt handler.

Multithread safe.

Arguments

None

Return Values

A Voice Definition structure, `SceNgsVoiceDefinition`.

Description

Returns a pointer to the mixer buss Voice Definition. This Voice type allows users to mix the outputs of various Voices by means of Voices patched into the mixer; no further DSP effects are present in this Voice type.

See Also

`SceNgsRackDescription`

sceNgsVoiceDefGetPauserBuss

Returns a pointer to the pauser buss Voice Definition.

Definition

```
#include <ngs/templates/pauser_buss_voice.h>
struct SceNgsVoiceDefinition *sceNgsVoiceDefGetPauserBuss(void);
```

Calling Conditions

Can be called from an interrupt handler.

Multithread safe.

Arguments

None

Return Values

A Voice Definition structure, `SceNgsVoiceDefinition`.

Description

Returns a pointer to the pauser buss Voice Definition. This Voice type allows the user to seamlessly pause and resume the audio patched into the Voice.

See Also

`SceNgsRackDescription`

sceNgsVoiceDefGetPitchShiftBuss

Returns a pointer to the Pitch Shift buss Voice Definition.

Definition

```
#include <ngs/templates/pitchshift_buss_voice.h>
struct SceNgsVoiceDefinition *sceNgsVoiceDefGetPitchShiftBuss(void);
```

Calling Conditions

Can be called from an interrupt handler.

Multithread safe.

Arguments

None

Return Values

A Voice Definition structure, `SceNgsVoiceDefinition`.

Description

Returns a pointer to the Pitch Shift buss Voice Definition. This Voice type allows the user to modify the pitch of the Voices patched into it.

See Also

`SceNgsRackDescription`

sceNgsVoiceDefGetReverbBuss

Returns a pointer to the Reverb buss Voice Definition.

Definition

```
#include <ngs/templates/reverb_buss_voice.h>
struct SceNgsVoiceDefinition *sceNgsVoiceDefGetReverbBuss(void);
```

Calling Conditions

Can be called from an interrupt handler.

Multithread safe.

Arguments

None

Return Values

A Voice Definition structure, `SceNgsVoiceDefinition`.

Description

Returns a pointer to the Reverb buss Voice Definition. This Voice type allows the user to apply reverb to the audio patched into the Voice.

See Also

`SceNgsRackDescription`

sceNgsVoiceDefGetSasEmuVoice

Returns a pointer to a libSas emulation voice type.

Definition

```
#include <ngs/templates/sas_emu_voice.h>
struct SceNgsVoiceDefinition *sceNgsVoiceDefGetSasEmuVoice(void);
```

Calling Conditions

Can be called from an interrupt handler.

Multithread safe.

Arguments

None

Return Values

A Voice Definition structure, `SceNgsVoiceDefinition`.

Description

Returns a pointer to a libSas emulation voice type. The voice type is of a simple structure containing a PCM/VAG player module followed by an envelope module.

See Also

`SceNgsRackDescription`

sceNgsVoiceDefGetSimpleVoice

Returns a pointer to a simple voice definition.

Definition

```
#include <ngs/templates/simple_voice.h>
struct SceNgsVoiceDefinition *sceNgsVoiceDefGetSimpleVoice(void);
```

Calling Conditions

Can be called from an interrupt handler.

Multithread safe.

Arguments

None

Return Values

A Voice Definition structure, `SceNgsVoiceDefinition`.

Description

Returns a pointer to a simple voice definition suitable for 3D positional sources. The voice has a PCM/VAG player module as the source which is then routed through the following modules in series: Parametric EQ, Envelope and Pauser. The output is then split into two parallel paths to the outputs with an independent Filter module on each. The parallel pathing is designed such that obstruction and occlusion effects can be applied by modifying the filters to wet and dry routing independently.

See Also

`SceNgsRackDescription`

sceNgsVoiceDefGetSimpleAtrac9Voice

Returns a pointer to a simple ATRAC9™ voice definition.

Definition

```
#include <ngs/templates/simple_voice_at9.h>
struct SceNgsVoiceDefinition *sceNgsVoiceDefGetSimpleAtrac9Voice(void);
```

Calling Conditions

Can be called from an interrupt handler.

Multithread safe.

Arguments

None

Return Values

A Voice Definition structure, `SceNgsVoiceDefinition`.

Description

Returns a pointer to a simple ATRAC9™ Voice Definition. PCM/VAG Player Module in simple Voice Definition is replaced with ATRAC9™ Player Module, and this module configuration is used for this Voice type.

See Also

`SceNgsRackDescription`

sceNgsVoiceDefGetTemplate1

Returns a pointer to the T1 Voice Definition.

Definition

```
#include <ngs/templates/voice_template_1.h>
struct SceNgsVoiceDefinition *sceNgsVoiceDefGetTemplate1(void);
```

Calling Conditions

Can be called from an interrupt handler.

Multithread safe.

Arguments

None

Return Values

A Voice Definition structure, `SceNgsVoiceDefinition`.

Description

Returns a pointer to the T1 Voice Definition. This Voice type acts as a source and can playback both PCM and ADPCM data as well as having a signal generator. The Voice is also equipped with a parametric EQ, distortion and envelope in serial before splitting to four outputs. Each output has an independent parametric EQ attached.

See Also

`SceNgsRackDescription`

sceNgsVoiceDefGetAtrac9Voice

Returns a pointer to the ATRAC9™ Voice Definition.

Definition

```
#include <ngs/templates/voice_template_at9.h>
struct SceNgsVoiceDefinition *sceNgsVoiceDefGetAtrac9Voice(void);
```

Calling Conditions

Can be called from an interrupt handler.

Multithread safe.

Arguments

None

Return Values

A Voice Definition structure, `SceNgsVoiceDefinition`.

Description

Returns a pointer to the ATRAC9™ Voice Definition. PCM/VAG Player Module in T1 Voice Definition is replaced with ATRAC9™ Player Module, and this module configuration is used for this Voice type.

See Also

`SceNgsRackDescription`

Patch Command Functions

sceNgsPatchCreateRouting

Creates a routing patch between two Voices.

Definition

```
#include <ngs/ngs_top.h>
SceInt32 sceNgsPatchCreateRouting(
        const SceNgsPatchSetupInfo *pPatchInfo,
        SceNgsHPatch *pPatchHandle
);
```

Calling Conditions

Can be called from an interrupt handler.

Multithread safe.

Arguments

pPatchInfo	Information regarding the routing patch required to be created.
pPatchHandle	Patch handle, returned to user.

Return Values

See SCE_NGS_OK and other NGS return codes for more information.

Description

Creates a routing patch between two Voices. The Voices must be in separate Racks and the routing must not cause circular dependency.

sceNgsPatchGetInfo

Returns information regarding an existing routing patch between two Voices.

Definition

```
#include <ngs/ngs_top.h>
SceInt32 sceNgsPatchGetInfo(
        SceNgsHPatch hPatchHandle,
        SceNgsPatchRouteInfo *pRouteInfo,
        SceNgsPatchSetupInfo *pSetup
);
```

Calling Conditions

Can be called from an interrupt handler.

Multithread safe.

Arguments

hPatchHandle	Patch handle, returned to user.
pRouteInfo	Structure to contain information regarding the routing (number of input channels, number of output channels and volumes).
pSetup	Structure to contain information regarding the patch (source and destination Voices, etc.).

Return Values

See SCE_NGS_OK and other NGS return codes for more information.

Description

Returns information regarding an existing routing patch between two Voices. The Voices must be in separate Racks and the routing must not cause circular dependency.

sceNgsPatchRemoveRouting

Disconnects a patch between two Voices.

Definition

```
#include <ngs/ngs_top.h>
SceInt32 sceNgsPatchRemoveRouting(
        SceNgsHPatch hPatchHandle
);
```

Calling Conditions

Can be called from an interrupt handler.

Multithread safe.

Arguments

hPatchHandle	Handle to be removed.

Return Values

See SCE_NGS_OK and other NGS return codes for more information.

Description

Disconnects a patch between two Voices.

ATRAC9™ Helper Command Functions

sceNgsAT9GetSectionDetails

Aids arbitrary, sample accurate, seeking and/or looping with ATRAC9™ files.

Definition

```
#include <ngs/ngs_top.h>
SceInt32 sceNgsAT9GetSectionDetails(
        const SceInt32 nStartSampleOffset,
        const SceInt32 nNumSamples,
        const SceInt32 nConfigData,
        SceNgsAT9SkipBufferInfo *pAt9InfoBuffer
);
```

Calling Conditions

Can be called from an interrupt handler.

Multithread safe.

Arguments

nStartSampleOffset	Start sample offset from original (pre-encoded) data at which the segment should start. If taking loop information from the ATRAC9™ file header, this should be bitwise OR'ed with SCE_NGS_SAMPLE_OFFSET_FROM_AT9_HEADER.
nNumSamples	Number of samples requested in segment.
nConfigData	Config data value from loaded ATRAC9™ file header.
pAt9InfoBuffer	Pointer to SceNgsAT9SkipBufferInfo to fill with return values to pass into SceNgsAT9BufferParams for desired playback.

Return Values

See SCE_NGS_OK and other NGS return codes for more information.

Description

This function is designed to aid arbitrary, sample accurate, seeking and/or looping with ATRAC9™ files.

The user passes the configuration data (see ConfigData in the "fmt Chunk" section of the "ATRAC9™ File Format" document) and the desired start and size of the segment.

The function then calculates the required offsets within the data to pass into the NGS ATRAC9™ Player DSP buffer for the desired playback.

When retrieving a sample offset which is embedded in the header information for the ATRAC9™ file, you should logically OR SCE_NGS_SAMPLE_OFFSET_FROM_AT9_HEADER with the sample offset.

Callback Type Definitions

SceNgsCallbackFunc

Base callback function pointer type for all NGS callbacks.

Definition

```
#include <ngs/ngs_top.h>
typedef void (*SceNgsCallbackFunc)(
        const SceNgsCallbackInfo *pCallbackInfo
);
```

Arguments

pCallbackInfo	Pointer to an SceNgsRackReleaseCallbackInfo structure describing the callback information of the Rack.

Return Values

None

Description

Base callback function pointer type for all NGS callbacks. The callbacks generated by Modules will vary depending on the Module in question.

Rack release callbacks: When requesting the release of a Rack, it is necessary to wait for the system to be in a stable state before any allocated memory can be freed. By setting a callback function, the user will be notified via this function when the Rack has been released from the system

Module callbacks: Modules may have callbacks, which the user can setup, for example a PCM player may trigger callbacks when input buffers have been used, allowing the user to stream data into a double buffer.

Params Error callbacks: These callbacks are triggered if an error is reported in the modules internal processing. Parameter error callbacks can be associated at a system-wide and/or rack level. Note: In the case of param error callbacks only the last error will be reported.

See Also

sceNgsRackRelease, sceNgsVoiceSetModuleCallback, sceNgsSystemSetParamErrorCallback, sceNgsRackSetParamErrorCallback

SceNgsRackReleaseCallbackFunc

Callback function that is called after a Rack has been released from the system.

Definition

```
#include <ngs/ngs_top.h>
typedef SceNgsCallbackFunc SceNgsRackReleaseCallbackFunc;
```

Description

Callback function that is called after a Rack has been released from the system.

When requesting the release of a Rack it is necessary to wait for the system to be in a stable state before any allocated memory can be freed. By setting a callback function the user will be notified when the Rack has been released from the system.

See Also

sceNgsRackRelease

SceNgsModuleCallbackFunc

Function prototype for callback functions.

Definition

```
#include <ngs/ngs_top.h>
typedef SceNgsCallbackFunc SceNgsModuleCallbackFunc;
```

Description

Function prototype for callback functions.

Modules may have callbacks, which the user can setup. For example, a PCM player may trigger callbacks when input buffers have been used, allowing the user to stream data into a double buffer.

See Also

sceNgsVoiceSetModuleCallback

SceNgsParamsErrorCallbackFunc

Callback function that triggers if an error is reported in a Module's `ParamToStates()` functions.

Definition

```
#include <ngs/ngs_top.h>
typedef SceNgsCallbackFunc SceNgsParamsErrorCallbackFunc;
```

Description

Callback function that triggers if an error is reported in a Module's `ParamToStates()` functions. A typical example would be from an out of range parameter, resulting in the error return code from the internal function.

Parameter error callbacks can be associated at a system-wide and/or Rack level.

See Also

sceNgsSystemSetParamErrorCallback, sceNgsRackSetParamErrorCallback