libsulpha Overview

PlayStation®Vita Programmer Tool Runtime Library Release 3.50

Document serial number: 000004892117

Table of Contents

1 Library Overview

Characteristics

libsulpha is a library for visualizing, debugging and analyzing audio information. Such information can be captured from other SDK audio libraries and then analyzed via the Windows Sulpha Tool. The Sulpha system allows the user to trace problems or artifacts, by giving them the ability to analyze both the captured audio library function calls (along with their parameters), as well as by hearing (and seeing) the audio data that was also playing at that time. This allows for a fast and precise method of debugging audio problems.

Sulpha captures can also be sent to SCE support so that the support departments can easily see what the problem is, and how it occurs. This helps reduce the overall support time for audio related issues.

The Sulpha API and library functionality is included in the SDK (`libSceSulpha_stub.a` and `sulpha.h`).

In addition to this, Sulpha provides a mechanism to register plug-in agents which can perform system-specific captures. The agent currently available is Sulpha NGS (`sulpha_ngs.h`).

Sulpha requires a user-allocated memory buffer for storage and transmission of debug and PCM data. The size of the buffer depends on the configuration settings supplied to the Sulpha initialization function.

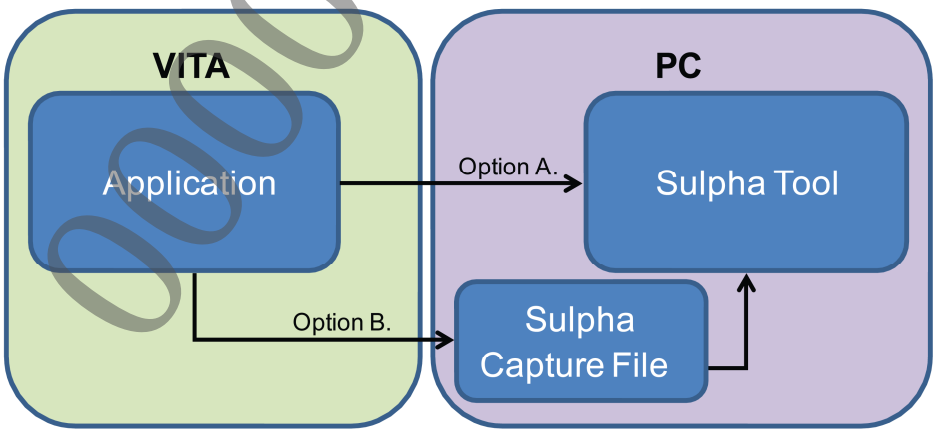
Sulpha Components

Sulpha consists of an API and a Windows based tool, the *Sulpha Tool*.

The API is placed within the VITA system, and is used to start and stop the capture of audio data. Data can be captured in a number of ways: Either saved as a file to disk (which can then be imported in the Sulpha Tool at a later date), or can be directly streamed to the Sulpha Tool.

The Sulpha Tool allows for the captured data to be viewed in many ways, as well as giving the user the ability to *play* capture files (showing all of the parameters which have been updated in real-time, as well as outputting the audio data via the PC speakers).

Figure 1



Option A: Direct capture of NGS audio activity to PC via Sulpha Tool

Option B: Save capture to disk. Capture can then be loaded into the Sulpha Tool.

Files

The files required to use the libsulpha library are as follows.

Table 1 Required Files

File Name	Description
`sulpha.h`	Header file
`libSceSulpha_stub.a`	Library file

Getting Started

The following lists the minimum requirements to start using Sulpha in your application:

(1) Include the Sulpha header file (`sulpha.h`) and link to the Sulpha core library (`libSceSulpha.a`).

(2) Add code to initiate, update and shutdown the Sulpha instance.

(3) Add code to initiate and register your Sulpha agents.

(4) Build and run the application.

(5) Start the Sulpha Tool and connect it to the development tool running your application (or open the captured file if using the file writing output mechanism).

Sample Programs

Sample programs using libsulpha are as follows.

samples\sample_code\audio_video\api_libngs

The samples within this folder can all output Sulpha data to file. This data can then be loaded into the Sulpha Tool for previewing.

Reference Materials

For details regarding the Sulpha API, see the following document:

- *libsulpha Reference*

For details regarding the Sulpha Tool, see the following document:

- *Sulpha Tool User's Guide*

For an overview of libNGS, see the following documents:

- *NGS Overview*
- *NGS Reference*
- *NGS Modules Overview*
- *NGS Modules Reference*

2 Using the Library

Basic Procedure

(1) Initialization

To use libsulpha, first initialize it as follows:

(1) Get the default configuration for Sulpha:

```
SceSulphaConfig config;
sceSulphaGetDefaultConfig(&config);
```

(2) Query the size needed for the Sulpha data buffers:

```
uint32_t size;
sceSulphaGetNeededMemory(&config, &size);
```

(3) Allocate memory and initiate Sulpha:

```
m_pSulphaMem = malloc(size);
sceSulphaInit(&config, m_pSulphaMem, size)
```

Where:

config is the configuration structure obtained in step (1)

size is the memory requirement obtained in step (2)

At this point the Sulpha instance is in a capture ready state.

(2) Registering Sulpha Agents

The registration of Sulpha agents is similar to the initialization of the Sulpha core:

(1) Get the default configuration for the agent:

```
SceSulphaNgsConfig ngsConfig;
sceSulphaNgsGetDefaultConfig(&ngsConfig);
```

(2) Query the size needed for the agent data buffers:

```
uint32_t agentSizeInBytes;
sceSulphaNgsGetNeededMemory(&ngsConfig, &agentSizeInBytes);
```

(3) Allocate memory and initiate the agent

```
m_pAgentsMem = malloc(agentSizeInBytes);
sceSulphaNgsInit(&ngsConfig, m_pAgentsMem, agentSizeInBytes);
```

(3) Naming Synths, Racks, Voices and PCM Data Buffers

"Synths", "Racks" and "Voices" are all terms used within the *libNGS* Synth library. For further information regarding the meaning of these terms, please see the libNGS documentation.

Naming objects can be useful when reviewing captured data in the Sulpha Tool. If an object is named then that name appears in the Sulpha Tool automatically when data for that object displays. This aids the developer's understanding when associated with Synths, Racks and Voices.

Please refer to the *libsulpha Reference* for a detailed look at the API.

(1) To name a static buffer (buffers used to supply an input to a voice) use the following function:

```
sceSulphaNgsSetSampleName(pBufferAddr, nBufferBytes, "GunFire.wav");
```

(2) To name a Synths, Racks or Voices use the following:

```
sceSulphaNgsSetSynthName(hSynth, "MySynth");
sceSulphaNgsSetRackName(hRack, "SFX Rack");
sceSulphaNgsSetVoiceName(hVoice, "Weapon 0");
```

(3) To remove or clear a name, supply a null value for the objects name:

```
sceSulphaNgsSetSampleName(pBufferAddr, nBufferBytes, NULL);
```

(4) Adding Trace Messages

The user can use custom text messages to indicate application specific information. This can be useful for a variety of purposes. For example, if a game is paused and a GUI is displayed then in-game sounds are muted and a series of *menu opening* sounds are played. A message like the following can be useful to explain what is happening:

```
sceSulphaNgsTrace("In-game GUI displayed");
```

By adding a comment like this, it makes it easier to identify what is occurring in the game system when reviewing captured data.

(5) Termination

To shut down Sulpha:

(1) Disconnect the file capture (if applicable):

```
sceSulphaFileDisconnect();
```

(2) Unregister the Sulpha NGS agent (if applicable):

```
sceSulphaNgsShutdown();
```

(3) Free the agent memory

```
if (m_pAgentMem != NULL){
    free(m_pAgentMem);
    m_pAgentMem = NULL;
}
```

(4) Call the Sulpha shutdown function

```
sceSulphaShutdown();
```

(5) Free the memory allocated to Sulpha

```
if (m_pSulphaMem != NULL){
    free(m_pSulphaMem);
    m_pSulphaMem = NULL;
```