libvoiceQoS Overview

PlayStation®Vita Programmer Tool Runtime Library Release 3.50

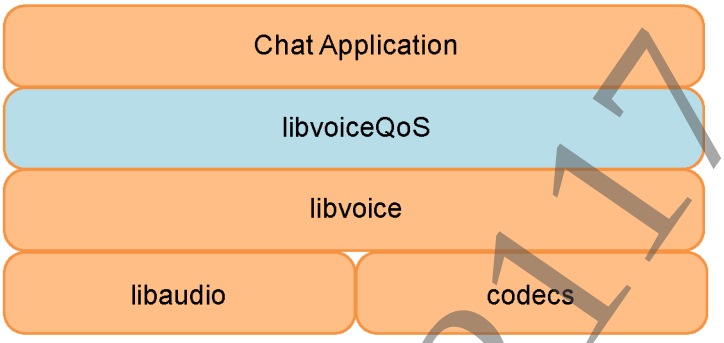
Document serial number: 000004892117

Table of Contents

1 Library Overview

The libvoiceQoS library provides a C function API for implementing voice chat with Quality of Service (QoS) functionality to ensure the priority and integrity of voice data. libvoiceQoS handles voice quality on behalf of the libvoice library, which is a voice data-handling library without QoS control. With libvoiceQoS, the voice quality is automatically managed at best effort. In addition, libvoiceQoS offers simpler APIs to use for VOIP development as compared to libvoice.

Figure 1 libvoiceQoS Architecture



Characteristics

libvoiceQoS builds upon the voice library. It creates and manages the voice chat topology; it provides quality-of-service adjustment for VOIP dynamically; it provides real-time voice data statistics.

Given a simple graph of local and remote endpoints and the connections between them, the library manages:

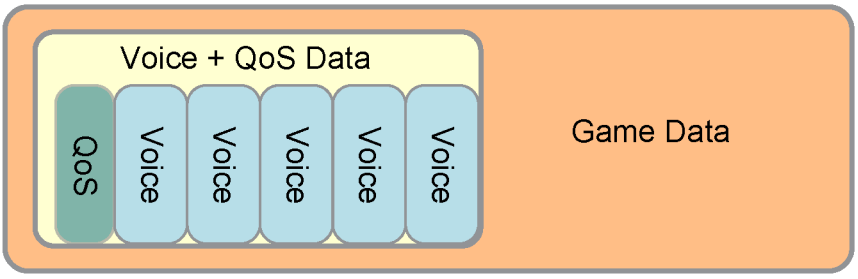
- Reading output and writing input
- Encoding and decoding
- Packetizing of voice data
- Adjustments of bandwidth and latency
- Handling of out-of-order and lost packets

The client of the library is responsible for handling all network connection setup and for transmitting and receiving all voice packets.

VoiceQoS Data Stream Layout

Generally, voice data is managed by the application the same way it handles normal game data. The game reserves space for voice-related data inside game data packets, which are then sent to the other players across the network. libvoiceQoS adds its own QoS data packet on top of the voice data from the libvoice (Figure 2). The game includes the VoiceQoS data with the normal game data packet and sends them out. The game controls the how often VoiceQoS data is put into the game data and sent out.

Figure 2 The QoS and Voice Data Packets with Game Data



Voice Connections

In a typical real-time voice chat application, all chat players setup their own network connections and communicate each other by sending and receiving voice data to each other. With libvoiceQoS, the topology setup for the chat party is greatly simplified: libvoiceQoS creates a matching audio and codec topology internally for the local and remote chat parties or "endpoints". The game is only responsible for creating the real network connections and passing the network data stream for the corresponding connections.

Figure 3 shows how microphone, speaker, and codecs are managed internally by libvoiceQoS.

Figure 3 Microphone, Speaker, and Codecs Managed Internally by libvoiceQoS

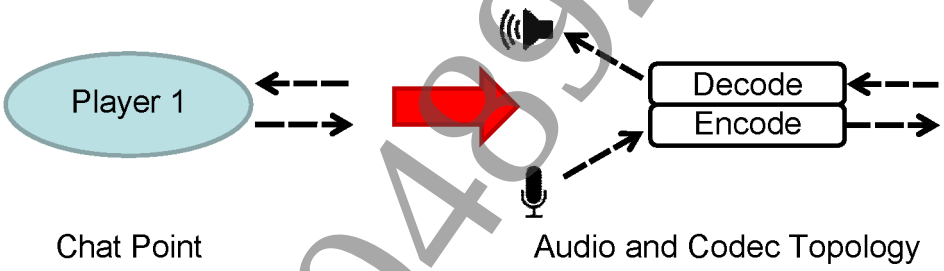


Figure 4 illustrates a local VoiceQoS endpoint. There is only one local endpoint on any system. Figure 5 shows multiple remote VoiceQoS endpoints.

Figure 4 Local VoiceQoS Endpoint

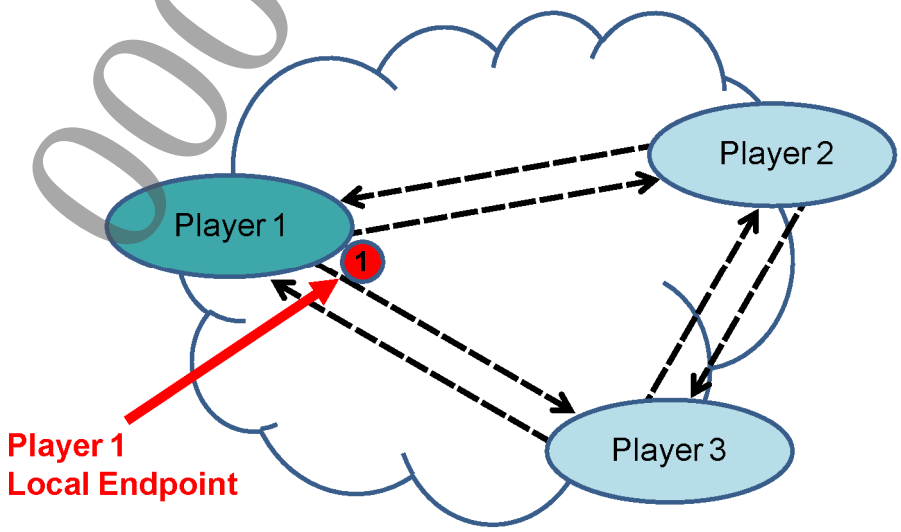


Figure 5 Remote VoiceQoS Endpoints

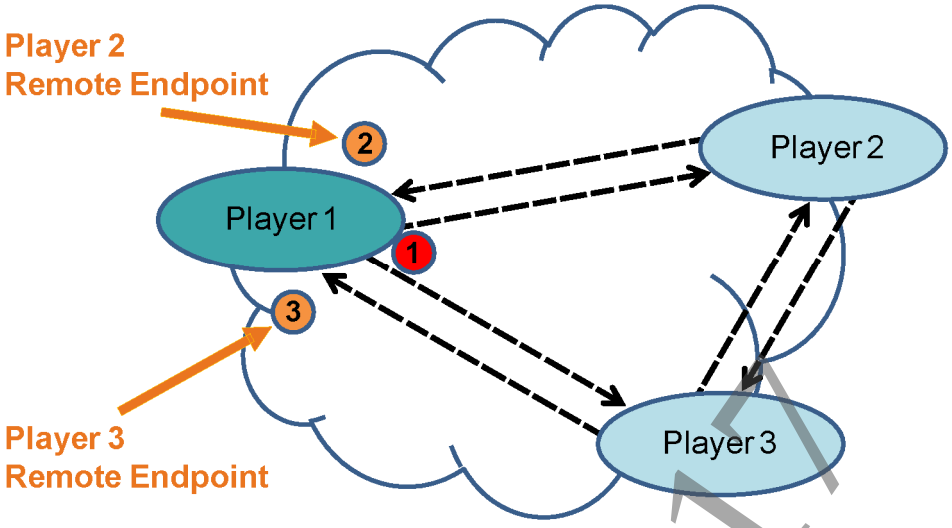
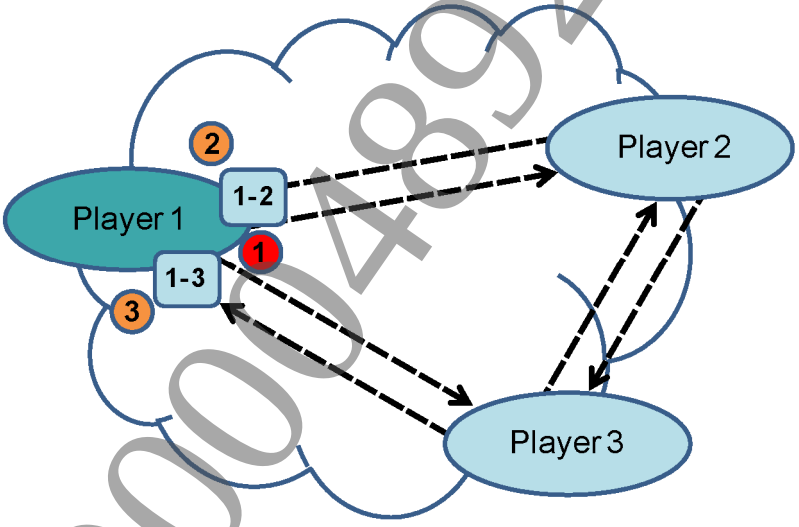


Figure 6 shows the creation of a VoiceQoS connection between a local endpoint and a remote endpoint. Player 1 has two connections: 1-2, 1-3.

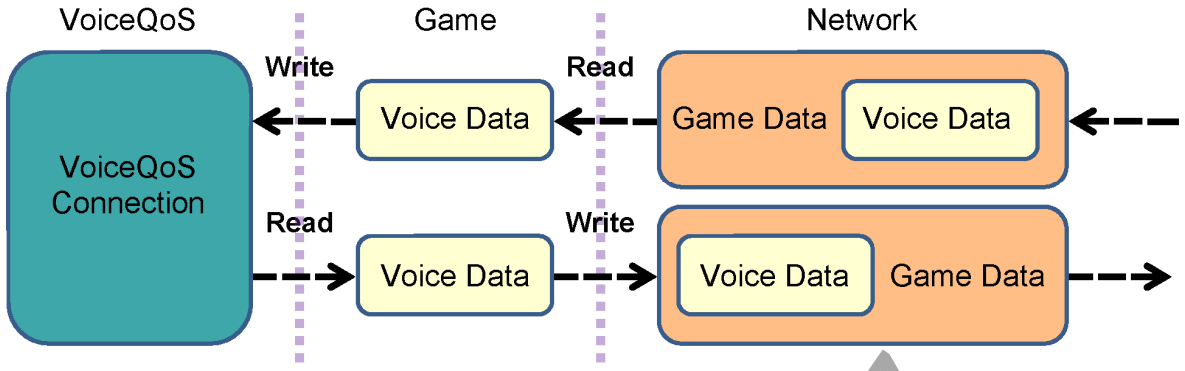
Figure 6 VoiceQoS Connection



VoiceQoS Read/Write

After VoiceQoS connections are establised among all chat parties, the application reads VoiceQoS data from the incoming network data stream and writes to the corresponding VoiceQoS connection. The application reads VoiceSoQ streams from the VoiceQoS connection and writes them into the corresponding outgoing network data stream. Figure 7 shows this data stream flow.

Figure 7 Data Stream Flow



In summary, the application manages the VoiceQoS data flow by calling libvoiceQoS read and write functions at its own pace after a VoiceQoS connection is established. The application also queries VoiceQoS status or set VoiceQoS attributes. The application should query the heartbeat status for the connection occasionally to know if the voice connection is still alive.

QoS Protocol

libvoiceQoS uses a lighweight protocol to do the QoS adjustment. The QoS data iteself is capped very small to reduce the bandwidth overhead in the network. To ensure voice quality of service, the internal QoS protocol dynamically reorders voice packets, adjusts voice codec bit rates, and resets the jitter buffer based on the current network conditions.

The application can query status attributes such as the current voice bitrate, the accumulated voice data receiving ratio, and the remote endpoint heartbeat status on any VoiceQoS connection through `sceVoiceQoSGetStatus()`.

Table 1 QoS Status

Status	Description
`SCE_VOICE_QOS_IN_BITRATE`	Current VoiceQoS input codec rate.
`SCE_VOICE_QOS_OUT_BITRATE`	Current VoiceQoS output codec rate.
`SCE_VOICE_QOS_OUT_READ_BITRATE`	Current VoiceQoS reading rate issued by the application.
`SCE_VOICE_QOS_IN_FRAME_RECEIVED_RATIO`	Ratio of frames received to total frames per connection, as a percentage.
`SCE_VOICE_QOS_HEARTBEAT_FLAG`	Remote heartbeat on status flag.

Currently, libvoiceQoS gives access to local endpoint and connection attributes through the following calls:

- `sceVoiceQoSSetLocalEndpointAttribute()`
- `sceVoiceQoSGetLocalEndpointAttribute()`
- `sceVoiceQoSSetConnectionAttribute()`
- `sceVoiceQoSGetConnectionAttribute()`

Table 2 Local Endpoint and Connection Attributes

Attribute	Description
`SCE_VOICE_QOS_ATTR_MIC_VOLUME`	Set the microphone volume of the local endpoint.
`SCE_VOICE_QOS_ATTR_MIC_MUTE`	Mute the microphone of the local endpoint.
`SCE_VOICE_QOS_ATTR_SPEAKER_VOLUME`	Set the speaker volume of the local endpoint.
`SCE_VOICE_QOS_ATTR_SPEAKER_MUTE`	Mute the speaker of the local endpoint.
`SCE_VOICE_QOS_ATTR_DESIRED_OUT_BIT_RATE`	Set the desired out bit-rate for the connection.

Attribute	Description
SCE_VOICE_QOS_ATTR_MIC_ACTIVE	Get the microphone state, which is either active or inactive, of the local endpoint.

Sample Programs

Sample programs for libvoiceQoS are located in the samples directory as follows:

```
samples/sample_code/audio_video/api_libvoiceqos/loop
```

The `voiceqos_loop` sample creates a simple VoiceQoS connection from the local endpoint to a remote endpoint. It shows how to get VoiceQoS data from the read API call `sceVoiceQoSReadPacket()` and how to write VoiceQoS data to the write API call `sceVoiceQoSWritePacket()`. It also queries the heartbeat status of the connection.

2 Using the Library

Basic Procedure

(1) Load the PRX Module

libvoiceQoS and libvoice are provided as two PRX modules. First, use

```
sceSysmoduleLoadModule(SCE_SYSMODULE_VOICE)
```

to load the libvoice module. Then, use

```
sceSysmoduleLoadModule(SCE_SYSMODULE_VOICEQOS)
```

to load the libvoiceQoS module.

(2) Initialize the Library

Call `sceVoiceQoSInit()` to initialize libvoiceQoS.

(3) Set Up a Connection

Call `sceVoiceQoSCreateLocalEndpoint()` to create the local endpoint and get an ID for it.

```
sceVoiceQoSCreateLocalEndpoint(&localId)
```

Call `sceVoiceQoSCreateRemoteEndpoint()` to create a remote endpoint and get an ID for it.

```
sceVoiceQoSCreateRemoteEndpoint(&remoteId)
```

Call `sceVoiceQoSConnect()` to setup a connection between the local endpoint and remote endpoint and to get an ID for the connection.

```
sceVoiceQoSConnect(&connectionId, localId, remoteId)
```

(4) Read/Write VoiceQoS Data and Check Status

Call `sceVoiceQoSReadPacket()` to read the VoiceQoS data from the corresponding connection specified by the connection ID.

```
sceVoiceQoSReadPacket(connectionId, dataBuf, sizeof(dataBuf))
```

The application then sends the VoiceQoS data to the remote player through the network.

Call `sceVoiceQoSWritePacket()` to write the incoming VoiceQoS data from the remote player to the corresponding connection specified by the connection ID.

```
sceVoiceQoSWritePacket(connectionId, dataBuf, &writeSize);
```

Call `sceVoiceQoSGetStatus()` to get the heartbeat status of the connection.

Continuously read and write the VoiceQoS data until the player quits the chat.

(5) Terminate a Connection

Call `sceVoiceQoSDisconnect()` to tear down a connection.

```
sceVoiceQoSDisconnect(connectionId)
```

Call `sceVoiceQoSDeleteLocalEndpoint()` to delete the local endpoint.

```
sceVoiceQoSDeleteLocalEndpoint(localId)
```

Call `sceVoiceQoSDeleteRemoteEndpoint()` to delete the remote endpoint.

```
sceVoiceQoSDeleteRemoteEndpoint(remoteId)
```

(6) End the Library

Call `sceVoiceQoSEnd()` to end libvoiceQoS.

(7) Unload the Module

First, use `sceSysmoduleUnloadModule` (`SCE_SYSMODULE_VOICEQOS`) to unload libvoiceQoS.

Then, use `sceSysmoduleUnloadModule` (`SCE_SYSMODULE_VOICE`) to unload libvoice.

List of Functions

Functions of libvoiceQoS can be separated into groups: those that perform creation and attribute management of endpoints and connections, those that read and write data across the connection, and those that initialize and terminate the library.

Functions Performing Initialization and Termination

Table 3 Initialization and Termination Functions (not multithread safe)

Function	Description
`sceVoiceQoSInit`	Initializes the VoiceQoS library.
`sceVoiceQoSEnd`	Terminates the VoiceQoS library.

Functions Performing Creation and Management of Endpoints and Connections

Table 4 Creation and Management of Endpoints and Connections Functions (multithread safe)

Function	Description
`sceVoiceQoSCreateLocalEndpoint`	Creates a local endpoint.
`sceVoiceQoSDeleteLocalEndpoint`	Deletes a local endpoint.
`sceVoiceQoSCreateRemoteEndpoint`	Deletes registration of the queue for normal audio event obtainment.
`sceVoiceQoSDeleteRemoteEndpoint`	Deletes registration of the queue for before-mix audio event obtainment.
`sceVoiceQoSConnect`	Create a connection between a local and remote endpoint.
`sceVoiceQoSDisconnect`	Deletes a connection between a local and remote endpoint.
`sceVoiceQoSGetLocalEndpoint`	Gets the local endpoint of the connection.
`sceVoiceQoSGetRemoteEndpoint`	Gets the remote endpoint of the connection.
`sceVoiceQoSSetLocalEndpointAttribute`	Sets local microphone and speaker attributes.
`sceVoiceQoSGetLocalEndpointAttribute`	Gets local microphone and speaker attributes.
`sceVoiceQoSSetConnectionAttribute`	Sets connection attributes.
`sceVoiceQoSGetConnectionAttribute`	Gets connection attributes.
`sceVoiceQoSGetStatus`	Gets a status value for the connection.

Functions Performing Data I/O

Table 5 Data I/O Functions (multithread safe)

Function	Description
`sceVoiceQoSWritePacket`	Writes a data packet received from a remote endpoint.
`sceVoiceQoSReadPacket`	Reads a data packet to send to a remote endpoint.

3 Notes

Resource Limitations

Currently a total of seven remote endpoints can be created using libvoiceQoS. This implies the maximum number of players in the chat is eight.

Single Voice-Out Bit-Rate Support

libvoiceQoS has only a single encoder on the system due to the limitation of libvoice. Updates to the voice-out bit-rate on one connection automatically triggers voice-out bit-rate updates on the rest of the connections on the system. For example, player A, B, and C are in a chat. If player A changes the voice-out bit-rate on the connection to player B, libvoiceQoS resets the voice-out bit-rate to the latest bit-rate on the connection from player A to player C, too.