

PortSIP VoIP SDK Manual for Android

Version 11.3
Tue Nov 15 2016

Table of Contents

Welcome to the PortSIP VoIP SDK	4
Getting Started	4
Contents	4
SDK User Manual	4
Website	4
Support	4
Installation Prerequisites	4
Frequently Asked Questions	4
1. Where can I download the PortSIP VoIP SDK for trial?	4
2. How can I compile the sample project?	5
3. How can I create a new project with PortSIP VoIP SDK?	5
4. How can I test the P2P call (without SIP server)?	5
5. Is the SDK thread safe?	5
Module Index	7
Hierarchical Index	8
Class Index	9
Module Documentation	10
SDK Callback events	10
Register events	10
Call events	11
Refer events	14
Signaling events	15
MWI events	16
DTMF events	17
INFO/OPTIONS message events	18
Presence events	18
Play audio and video file finished events	21
RTP callback events	22
SDK functions	24
Initialize and register functions	24
Audio and video codecs functions	27
Additional settings functions	30
Access SIP message header functions	35
Audio and video functions	37
Call functions	40
Refer functions	44
Send audio and video stream functions	45
RTP packets, Audio stream and video stream callback	47
Record functions	48
Play audio and video file to remote functions	49
Conference functions	51
RTP and RTCP QOS functions	52
RTP statistics functions	54
Audio effect functions	56
Send OPTIONS/INFO/MESSAGE functions	57
Device Management functions	60
Class Documentation	64
com.portsip.AndroidGLES20	64
com.portsip.OnPortSIPEvent	65
com.portsip.PortSipEnumDefine	67
com.portsip.PortSipErrorcode	70
com.portsip.PortSipSdk	73

com.portsip.Renderer	77
com.portsip.SurfaceRenderer	78
com.portsip.VideoCaptureAndroid.....	79
com.portsip.VideoCaptureDeviceInfoAndroid.....	80
Index.....	81

Welcome to the PortSIP VoIP SDK

Create your SIP-based application for multiple platforms (iOS, Android, Windows, Mac OS/Linux) with our SDK.

The rewarding PortSIP VoIP SDK is a powerful and versatile set of tools that dramatically accelerate SIP application development. It includes a suite of stacks, SDKs, and some Sample projects, with each of them enables developers to combine all the necessary components to create an ideal development environment for every application's specific needs.

The PortSIP VoIP SDK complies with IETF and 3GPP standards, and is IMS-compliant (3GPP/3GPP2, TISPAN and PacketCable 2.0). These high performance SDKs provide unified API layers for full user control and flexibility.

Getting Started

You can download PortSIP VoIP SDK Sample projects at our [Website](#). Samples include demos for VC++, C#, VB.NET, Delphi XE, XCode (for iOS and Mac OS), Eclipse (Java for Android) with the sample project source code provided (with SDK source code exclusive). The sample projects demonstrate how to create a powerful SIP application with our SDK easily and quickly.

Contents

The sample package for downloading contains almost all of materials for PortSIP SDK: documentation, Dynamic/Static libraries, sources, headers, datasheet, and everything else a SDK user might need!

SDK User Manual

To be started with, it is recommended to read the documentation of PortSIP VoIP SDK, [SDK User Manual page](#), which gives a brief description of each API functions.

Website

Some general interest or often changing PortSIP SDK information will be posted on the [PortSIP website](#) in real time. The release contains links to the site, so while browsing you may see occasional broken links if you aren't connected to the Internet. To be sure everything needed for using the PortSIP VoIP SDK has been contained within the release.

Support

Please send email to our [Support team](#) if you need any help.

Installation Prerequisites

To use PortSIP VoIP/IMS SDK for Android for development, SDK version with later than API-9 is required.

Frequently Asked Questions

1. Where can I download the PortSIP VoIP SDK for trial?

All sample projects of the PortSIP VoIP SDK can be found and downloaded at:
<http://www.portsip.com/downloads>

2. How can I compile the sample project?

1. Download the sample project from PortSIP website.
2. Extract the .zip file.
3. Open the project by your Eclipse or Android studio:
4. Compile the sample project directly. The trial version SDK allows you a 2-3 minutes conversation.

3. How can I create a new project with PortSIP VoIP SDK?

1. Download the sample project and evaluation SDK and extract it to a specified directory
 2. Run Eclipse and create a new Android Application Project
 3. Copy all files form libs directory under extracted directory to the libs directoy of your new application.
 4. Import the dependent class form the SDK. For example:

```
import com.portsip.OnPortSIPEvent;  
import com.portsip.PortSipSdk;
```
 5. Inherit the interface OnPortSIPEvent to process the callback events.
 6. Initialize SDK. For example:

```
mPortSIPSDK = new PortSipSdk();  
mPortSIPSDK.setOnPortSIPEvent(instanceofOnPortSIPEvent);  
mPortSIPSDK.CreateCallManager(context);  
mPortSIPSDK.initialize(...);
```
- For more details please refer to the Sample project source code.

4. How can I test the P2P call (without SIP server)?

1. Download and extract the SDK sample project .zip file into local. Compile and run the "P2PSample" project.
2. Run the P2Psample on two devices. For example, run it on device A and device B, and IP address for A is 192.168.1.10, IP address for B is 192.168.1.11.
3. Enter a user name and password on A. For example, enter user name 111, and password aaa (you can enter anything for the password as the SDK will ignore it). Enter a user name and password on B. For example, enter user name 222, and password aaa.
4. Click the "Initialize" button on A and B. If the default port 5060 is already in use, the P2PSample will prompt "Initialize failure". In case of this, please click the "Uninitialize" button and change the local port, and click the "Initialize" button again.
5. The log box will appear "Initialized" if the SDK is successfully initialized.
6. To make call from A to B, enter "sip:222@192.168.1.11" and click "Dial" button; while to make call from B to A, enter "sip:111@192.168.1.10".

Note: If the local sip port is changed to other port, for example, A is using local port 5080, and B is using local port 6021, to make call from A to B, please enter "sip:222@192.168.1.11:6021" and dial; while to make call from B to A, enter "sip:111@192.168.1.10:5080".

5. Is the SDK thread safe?

Yes, the SDK is thread safe. You can call any of the API functions without the need to consider the multiple threads. Note: the SDK allows to call API functions in callback events directly - except for the "onAudioRawCallback", "onVideoRawCallback", "onReceivedRtpPacket", "onSendingRtpPacket" callbacks.

Module Index

Modules

Here is a list of all modules:

SDK Callback events	10
Register events	10
Call events	11
Refer events	14
Signaling events	15
MWI events	16
DTMF events	17
INFO/OPTIONS message events	18
Presence events	18
Play audio and video file finished events	21
RTP callback events	22
SDK functions	24
Initialize and register functions	24
Audio and video codecs functions	27
Additional settings functions	30
Access SIP message header functions	35
Audio and video functions	37
Call functions	40
Refer functions	44
Send audio and video stream functions	45
RTP packets, Audio stream and video stream callback	47
Record functions	48
Play audio and video file to remote functions	49
Conference functions	51
RTP and RTCP QOS functions	52
RTP statistics functions	54
Audio effect functions	56
Send OPTIONS/INFO/MESSAGE functions	57
Device Management functions	60

Hierarchical Index

Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

com.portsip.OnPortSIPEvent	65
com.portsip.PortSipEnumDefine	67
com.portsip.PortSipErrorcode	70
com.portsip.PortSipSdk	73
com.portsip.Renderer	77
Renderer	
com.portsip.AndroidGLES20	64
com.portsip.VideoCaptureDeviceInfoAndroid	80
Callback	
com.portsip.SurfaceRenderer	78
com.portsip.VideoCaptureAndroid	79
GLSurfaceView	
com.portsip.AndroidGLES20	64
PreviewCallback	
com.portsip.VideoCaptureAndroid	79

Class Index

Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

<u>com.portsip.AndroidGLES20</u>	64
<u>com.portsip.OnPortSIPEvent</u>	65
<u>com.portsip.PortSipEnumDefine</u>	67
<u>com.portsip.PortSipErrorcode</u>	70
<u>com.portsip.PortSipSdk</u>	73
<u>com.portsip.Renderer</u>	77
<u>com.portsip.SurfaceRenderer</u>	78
<u>com.portsip.VideoCaptureAndroid</u>	79
<u>com.portsip.VideoCaptureDeviceInfoAndroid</u>	80

Module Documentation

SDK Callback events

Modules

- [Register events](#)
 - [Call events](#)
 - [Refer events](#)
 - [Signaling events](#)
 - [MWI events](#)
 - [DTMF events](#)
 - [INFO/OPTIONS message events](#)
 - [Presence events](#)
 - [Play audio and video file finished events](#)
 - [RTP callback events](#)
-

Detailed Description

SDK Callback events

Register events

Functions

- void [com.portsip.OnPortSIPEvent.onRegisterSuccess](#) (String reason, int code)
 - void [com.portsip.OnPortSIPEvent.onRegisterFailure](#) (String reason, int code)
-

Detailed Description

Register events

Function Documentation

void com.portsip.OnPortSIPEvent.onRegisterSuccess (String *reason*, int *code*)

When successfully registered to server, this event will be triggered.

Parameters:

<i>reason</i>	The status text.
<i>code</i>	The status code.

void com.portsip.OnPortSIPEvent.onRegisterFailure (String *reason*, int *code*)

If failed to register to SIP server, this event will be triggered.

Parameters:

<i>reason</i>	The status text.
<i>code</i>	The status code.

Call events

Functions

- void [com.portsip.OnPortSIPEvent.onInviteIncoming](#) (long sessionId, String callerDisplayName, String caller, String calleeDisplayName, String callee, String audioCodecs, String videoCodecs, boolean existsAudio, boolean existsVideo)
- void [com.portsip.OnPortSIPEvent.onInviteTrying](#) (long sessionId)
- void [com.portsip.OnPortSIPEvent.onInviteSessionProgress](#) (long sessionId, String audioCodecs, String videoCodecs, boolean existsEarlyMedia, boolean existsAudio, boolean existsVideo)
- void [com.portsip.OnPortSIPEvent.onInviteRinging](#) (long sessionId, String statusText, int statusCode)
- void [com.portsip.OnPortSIPEvent.onInviteAnswered](#) (long sessionId, String callerDisplayName, String caller, String calleeDisplayName, String callee, String audioCodecs, String videoCodecs, boolean existsAudio, boolean existsVideo)
- void [com.portsip.OnPortSIPEvent.onInviteFailure](#) (long sessionId, String reason, int code)
- void [com.portsip.OnPortSIPEvent.onInviteUpdated](#) (long sessionId, String audioCodecs, String videoCodecs, boolean existsAudio, boolean existsVideo)
- void [com.portsip.OnPortSIPEvent.onInviteConnected](#) (long sessionId)
- void [com.portsip.OnPortSIPEvent.onInviteBeginingForward](#) (String forwardTo)
- void [com.portsip.OnPortSIPEvent.onInviteClosed](#) (long sessionId)
- void [com.portsip.OnPortSIPEvent.onRemoteHold](#) (long sessionId)
- void [com.portsip.OnPortSIPEvent.onRemoteUnHold](#) (long sessionId, String audioCodecs, String videoCodecs, boolean existsAudio, boolean existsVideo)

Detailed Description

Function Documentation

void com.portsip.OnPortSIPEvent.onInviteIncoming (long sessionId, String callerDisplayName, String caller, String calleeDisplayName, String callee, String audioCodecs, String videoCodecs, boolean existsAudio, boolean existsVideo)

When a call is coming, this event will be triggered.

Parameters:

<i>sessionId</i>	The session ID of the call.
<i>callerDisplayName</i>	The display name of caller
<i>caller</i>	The caller.
<i>calleeDisplayName</i>	The display name of callee.
<i>callee</i>	The callee.
<i>audioCodecs</i>	The matched audio codecs. It's separated by "#" if there are more than one codecs.
<i>videoCodecs</i>	The matched video codecs. It's separated by "#" if there are more than one codecs.
<i>existsAudio</i>	By setting to true, it means that this call include the audio.
<i>existsVideo</i>	By setting to true, it means that this call include the video.

void com.portsip.OnPortSIPEvent.onInviteTrying (long *sessionId*)

If the outgoing call is being processed, this event will be triggered.

Parameters:

<i>sessionId</i>	The session ID of the call.
------------------	-----------------------------

void com.portsip.OnPortSIPEvent.onInviteSessionProgress (long *sessionId*, String *audioCodecs*, String *videoCodecs*, boolean *existsEarlyMedia*, boolean *existsAudio*, boolean *existsVideo*)

Once the caller received the "183 session progress" message, this event would be triggered.

Parameters:

<i>sessionId</i>	The session ID of the call.
<i>audioCodecs</i>	The matched audio codecs. It's separated by "#" if there are more than one codecs.
<i>videoCodecs</i>	The matched video codecs. It's separated by "#" if there are more than one codecs.
<i>existsEarlyMedia</i>	By setting to true it means the call has early media.
<i>existsAudio</i>	By setting to true it means this call include the audio.
<i>existsVideo</i>	By setting to true it means this call include the video.

void com.portsip.OnPortSIPEvent.onInviteRinging (long *sessionId*, String *statusText*, int *statusCode*)

If the outgoing call is ringing, this event will be triggered.

Parameters:

<i>sessionId</i>	The session ID of the call.
<i>statusText</i>	The status text.
<i>statusCode</i>	The status code.

void com.portsip.OnPortSIPEvent.onInviteAnswered (long *sessionId*, String *callerDisplayName*, String *caller*, String *calleeDisplayName*, String *callee*, String *audioCodecs*, String *videoCodecs*, boolean *existsAudio*, boolean *existsVideo*)

If the remote party answered the call, this event would be triggered.

Parameters:

<i>sessionId</i>	The session ID of the call.
<i>callerDisplayName</i>	The display name of caller
<i>caller</i>	The caller.
<i>calleeDisplayName</i>	The display name of callee.
<i>callee</i>	The callee.
<i>audioCodecs</i>	The matched audio codecs. It's separated by "#" if there are more than one codecs.
<i>videoCodecs</i>	The matched video codecs. It's separated by "#" if there are more than one codecs.
<i>existsAudio</i>	By setting to true, this call includes the audio.
<i>existsVideo</i>	By setting to true, this call includes the video.

void com.portsip.OnPortSIPEvent.onInviteFailure (long *sessionId*, String *reason*, int *code*)

If the outgoing call is failed, this event will be triggered.

Parameters:

<i>sessionId</i>	The session ID of the call.
<i>reason</i>	The failure reason.
<i>code</i>	The failure code.

void com.portsip.OnPortSIPEvent.onInviteUpdated (long *sessionId*, String *audioCodecs*, String *videoCodecs*, boolean *existsAudio*, boolean *existsVideo*)

This event will be triggered when remote party updates the call.

Parameters:

<i>sessionId</i>	The session ID of the call.
<i>audioCodecs</i>	The matched audio codecs. It's separated by "#" if there are more than one codecs.
<i>videoCodecs</i>	The matched video codecs. It's separated by "#" if there are more than one codecs.
<i>existsAudio</i>	By setting to true, this call includes the audio.
<i>existsVideo</i>	By setting to true, this call includes the video.

void com.portsip.OnPortSIPEvent.onInviteConnected (long *sessionId*)

This event will be triggered when UAC sent/UAS received ACK (the call is connected). Some functions (hold, updateCall etc...) can be called only after the call connected, otherwise the functions will return error.

Parameters:

<i>sessionId</i>	The session ID of the call.
------------------	-----------------------------

void com.portsip.OnPortSIPEvent.onInviteBeginingForward (String *forwardTo*)

If the enableCallForward method is called and a call is incoming, the call will be forwarded automatically and this event will be triggered.

Parameters:

<i>forwardTo</i>	The target SIP URI of the call forwarding.
------------------	--

void com.portsip.OnPortSIPEvent.onInviteClosed (long *sessionId*)

This event is triggered once remote side ends the call.

Parameters:

<i>sessionId</i>	The session ID of the call.
------------------	-----------------------------

void com.portsip.OnPortSIPEvent.onRemoteHold (long *sessionId*)

If the remote side places the call on hold, this event will be triggered.

Parameters:

<i>sessionId</i>	The session ID of the call.
------------------	-----------------------------

void com.portsip.OnPortSIPEvent.onRemoteUnHold (long *sessionId*, String *audioCodecs*, String *videoCodecs*, boolean *existsAudio*, boolean *existsVideo*)

If the remote side un-holds the call, this event will be triggered

Parameters:

<i>sessionId</i>	The session ID of the call.
<i>audioCodecs</i>	The matched audio codecs. It's separated by "#" if there are more than one codec.

<i>videoCodecs</i>	The matched video codecs. It's separated by "#" if there are more than one codec.
<i>existsAudio</i>	By setting to true, this call includes the audio.
<i>existsVideo</i>	By setting to true, this call includes the video.

Refer events

Functions

- void [com.portsip.OnPortSIPEvent.onReceivedRefer](#) (long sessionId, long referId, String to, String from, String referSipMessage)
- void [com.portsip.OnPortSIPEvent.onReferAccepted](#) (long sessionId)
- void [com.portsip.OnPortSIPEvent.onReferRejected](#) (long sessionId, String reason, int code)
- void [com.portsip.OnPortSIPEvent.onTransferTrying](#) (long sessionId)
- void [com.portsip.OnPortSIPEvent.onTransferRinging](#) (long sessionId)
- void [com.portsip.OnPortSIPEvent.onACTVTransferSuccess](#) (long sessionId)
- void [com.portsip.OnPortSIPEvent.onACTVTransferFailure](#) (long sessionId, String reason, int code)

Detailed Description

Function Documentation

void com.portsip.OnPortSIPEvent.onReceivedRefer (long sessionId, long referId, String to, String from, String referSipMessage)

This event will be triggered once received a REFER message.

Parameters:

<i>sessionId</i>	The session ID of the call.
<i>referId</i>	The ID of the REFER message. Pass it to acceptRefer or rejectRefer
<i>to</i>	The refer target.
<i>from</i>	The sender of REFER message.
<i>referSipMessage</i>	The SIP message of "REFER". Pass it to "acceptRefer" function.

void com.portsip.OnPortSIPEvent.onReferAccepted (long sessionId)

This callback will be triggered once remote side calls "acceptRefer" to accept the REFER

Parameters:

<i>sessionId</i>	The session ID of the call.
------------------	-----------------------------

void com.portsip.OnPortSIPEvent.onReferRejected (long sessionId, String reason, int code)

This callback will be triggered once remote side calls "rejectRefer" to reject the REFER

Parameters:

<i>sessionId</i>	The session ID of the call.
<i>reason</i>	Reject reason.
<i>code</i>	Reject code.

void com.portsip.OnPortSIPEvent.onTransferTrying (long *sessionId*)

When the refer call is being processed, this event will be triggered.

Parameters:

<i>sessionId</i>	The session ID of the call.
------------------	-----------------------------

void com.portsip.OnPortSIPEvent.onTransferRinging (long *sessionId*)

When the refer call is ringing, this event will be triggered.

Parameters:

<i>sessionId</i>	The session ID of the call.
------------------	-----------------------------

void com.portsip.OnPortSIPEvent.onACTVTransferSuccess (long *sessionId*)

When the refer call succeeds, this event will be triggered. The ACTV means Active. For example: A establishes the call with B, A transfers B to C, C accepts the refer call, and A will receive this event.

Parameters:

<i>sessionId</i>	The session ID of the call.
------------------	-----------------------------

void com.portsip.OnPortSIPEvent.onACTVTransferFailure (long *sessionId*, String *reason*, int *code*)

When the refer call fails, this event will be triggered. The ACTV means Active. For example: A establish the call with B, A transfers B to C, C rejects this refer call, and A will receive this event.

Parameters:

<i>sessionId</i>	The session ID of the call.
<i>reason</i>	The error reason.
<i>code</i>	The error code.

Signaling events

Functions

- void [com.portsip.OnPortSIPEvent.onReceivedSignaling](#) (long *sessionId*, String *message*)
- void [com.portsip.OnPortSIPEvent.onSendingSignaling](#) (long *sessionId*, String *message*)

Detailed Description

Function Documentation

void com.portsip.OnPortSIPEvent.onReceivedSignaling (long *sessionId*, String *message*)

This event will be triggered when receiving a SIP message.

Parameters:

<i>sessionId</i>	The session ID of the call.
<i>message</i>	The received SIP message.

void com.portsip.OnPortSIPEvent.onSendingSignaling (long *sessionId*, String *message*)

This event will be triggered when sent a SIP message.

Parameters:

<i>sessionId</i>	The session ID of the call.
<i>message</i>	The sent SIP message.

MWI events

Functions

- void [com.portsip.OnPortSIPEvent.onWaitingVoiceMessage](#) (String *messageAccount*, int *urgentNewMessageCount*, int *urgentOldMessageCount*, int *newMessageCount*, int *oldMessageCount*)
- void [com.portsip.OnPortSIPEvent.onWaitingFaxMessage](#) (String *messageAccount*, int *urgentNewMessageCount*, int *urgentOldMessageCount*, int *newMessageCount*, int *oldMessageCount*)

Detailed Description

Function Documentation

void com.portsip.OnPortSIPEvent.onWaitingVoiceMessage (String *messageAccount*, int *urgentNewMessageCount*, int *urgentOldMessageCount*, int *newMessageCount*, int *oldMessageCount*)

If there is the waiting voice message(MWI), this event will be triggered.

Parameters:

<i>messageAccount</i>	Voice message account
<i>urgentNewMessageCount</i>	Count of new urgent messages.
<i>urgentOldMessageCount</i>	Count of history urgent message.
<i>newMessageCount</i>	Count of new messages.
<i>oldMessageCount</i>	Count of history messages.

void com.portsip.OnPortSIPEvent.onWaitingFaxMessage (String *messageAccount*, int *urgentNewMessageCount*, int *urgentOldMessageCount*, int *newMessageCount*, int *oldMessageCount*)

If there is waiting fax message (MWI), this event will be triggered.

Parameters:

<i>messageAccount</i>	Fax message account
<i>urgentNewMessageCount</i>	Count of new urgent messages.

<i>eCount</i>	
<i>urgentOldMessageCount</i>	Count of history urgent messages.
<i>newMessageCount</i>	Count of new messages.
<i>oldMessageCount</i>	Count of old messages.

DTMF events

Functions

- void [com.portsip.OnPortSIPEvent.onRecvDtmfTone](#) (long sessionId, int tone)

Detailed Description

Function Documentation

void com.portsip.OnPortSIPEvent.onRecvDtmfTone (long *sessionId*, int *tone*)

This event will be triggered when receiving a DTMF tone from remote side.

Parameters:

<i>sessionId</i>	Session ID of the call.
<i>tone</i>	

code	Description
0	The DTMF tone 0.
1	The DTMF tone 1.
2	The DTMF tone 2.
3	The DTMF tone 3.
4	The DTMF tone 4.
5	The DTMF tone 5.
6	The DTMF tone 6.
7	The DTMF tone 7.
8	The DTMF tone 8.
9	The DTMF tone 9.
10	The DTMF tone *.
11	The DTMF tone #.
12	The DTMF tone A.
13	The DTMF tone B.
14	The DTMF tone C.
15	The DTMF tone D.
16	The DTMF tone FLASH.

INFO/OPTIONS message events

Functions

- void [com.portsip.OnPortSIPEvent.onRecvOptions](#) (String optionsMessage)
 - void [com.portsip.OnPortSIPEvent.onRecvInfo](#) (String infoMessage)
-

Detailed Description

Function Documentation

void com.portsip.OnPortSIPEvent.onRecvOptions (String *optionsMessage*)

This event will be triggered when receiving the OPTIONS message.

Parameters:

<i>optionsMessage</i>	The received whole OPTIONS message in text format.
-----------------------	--

void com.portsip.OnPortSIPEvent.onRecvInfo (String *infoMessage*)

This event will be triggered when receiving the INFO message.

Parameters:

<i>infoMessage</i>	The whole INFO message received in text format.
--------------------	---

Presence events

Functions

- void [com.portsip.OnPortSIPEvent.onPresenceRecvSubscribe](#) (long subscribeId, String fromDisplayName, String from, String subject)
 - void [com.portsip.OnPortSIPEvent.onPresenceOnline](#) (String fromDisplayName, String from, String stateText)
 - void [com.portsip.OnPortSIPEvent.onPresenceOffline](#) (String fromDisplayName, String from)
 - void [com.portsip.OnPortSIPEvent.onRecvMessage](#) (long sessionId, String mimeType, String subMimeType, byte[] messageData, int messageDataLength)
 - void [com.portsip.OnPortSIPEvent.onRecvOutOfDialogMessage](#) (String fromDisplayName, String from, String toDisplayName, String to, String mimeType, String subMimeType, byte[] messageData, int messageDataLength)
 - void [com.portsip.OnPortSIPEvent.onSendMessageSuccess](#) (long sessionId, long messageId)
 - void [com.portsip.OnPortSIPEvent.onSendMessageFailure](#) (long sessionId, long messageId, String reason, int code)
 - void [com.portsip.OnPortSIPEvent.onSendOutOfDialogMessageSuccess](#) (long messageId, String fromDisplayName, String from, String toDisplayName, String to)
 - void [com.portsip.OnPortSIPEvent.onSendOutOfDialogMessageFailure](#) (long messageId, String fromDisplayName, String from, String toDisplayName, String to, String reason, int code)
-

Detailed Description

Function Documentation

void com.portsip.OnPortSIPEvent.onPresenceRecvSubscribe (long *subscribeId*, String *fromDisplayName*, String *from*, String *subject*)

This event will be triggered when receiving the SUBSCRIBE request from a contact.

Parameters:

<i>subscribeId</i>	The ID of SUBSCRIBE request.
<i>fromDisplayName</i>	The display name of contact.
<i>from</i>	The contact who sends the SUBSCRIBE request.
<i>subject</i>	The subject of the SUBSCRIBE request.

void com.portsip.OnPortSIPEvent.onPresenceOnline (String *fromDisplayName*, String *from*, String *stateText*)

When the contact is online or changes presence status, this event will be triggered.

Parameters:

<i>fromDisplayName</i>	The display name of contact.
<i>from</i>	The contact who sends the SUBSCRIBE request.
<i>stateText</i>	The presence status text.

void com.portsip.OnPortSIPEvent.onPresenceOffline (String *fromDisplayName*, String *from*)

When the contact is offline, this event will be triggered.

Parameters:

<i>fromDisplayName</i>	The display name of contact.
<i>from</i>	The contact who sends the SUBSCRIBE request

void com.portsip.OnPortSIPEvent.onRecvMessage (long *sessionId*, String *mimeType*, String *subMimeType*, byte [] *messageData*, int *messageDataLength*)

This event will be triggered when receiving a MESSAGE message in dialog.

Parameters:

<i>sessionId</i>	The session ID of the call.
<i>mimeType</i>	The message mime type.
<i>subMimeType</i>	The message sub mime type.
<i>messageData</i>	The received message body. It can be text or binary data. Use the mimeType and subMimeType to differentiate them. For example, if the mimeType is "text" and subMimeType is "plain", "messageData" is text message body. If the mimeType is "application" and subMimeType is "vnd.3gpp.sms", "messageData" is binary message body.
<i>messageDataLength</i>	The length of "messageData".

void com.portsip.OnPortSIPEvent.onRecvOutOfDialogMessage (String *fromDisplayName*, String *from*, String *toDisplayName*, String *to*, String *mimeType*, String *subMimeType*, byte [] *messageData*, int *messageDataLength*)

This event will be triggered when receiving a MESSAGE message out of dialog. For example pager message.

Parameters:

<i>fromDisplayName</i>	The display name of sender.
<i>from</i>	The message sender.
<i>toDisplayName</i>	The display name of receiver.
<i>to</i>	The receiver.
<i>mimeType</i>	The message mime type.
<i>subMimeType</i>	The message sub mime type.
<i>messageData</i>	The received message body. It can be text or binary data. Use the mimeType and subMimeType to differentiate them. For example, if the mimeType is "text" and subMimeType is "plain", "messageData" is text message body. If the mimeType is "application" and subMimeType is "vnd.3gpp.sms", "messageData" is binary message body.
<i>messageDataLength</i>	The length of "messageData".

void com.portsip.OnPortSIPEvent.onSendMessageSuccess (long *sessionId*, long *messageId*)

If the message is sent successfully in dialog, this event will be triggered.

Parameters:

<i>sessionId</i>	The session ID of the call.
<i>messageId</i>	The message ID, it's equals the return value of sendMessage function.

void com.portsip.OnPortSIPEvent.onSendMessageFailure (long *sessionId*, long *messageId*, String *reason*, int *code*)

If the message is failed to be sent out of dialog, this event will be triggered.

Parameters:

<i>sessionId</i>	The session ID of the call.
<i>messageId</i>	The message ID. It's equal to the return value of sendMessage function.
<i>reason</i>	The failure reason.
<i>code</i>	Failure code.

void com.portsip.OnPortSIPEvent.onSendOutOfDialogMessageSuccess (long *messageId*, String *fromDisplayName*, String *from*, String *toDisplayName*, String *to*)

If the message is sent successfully out of dialog, this event will be triggered.

Parameters:

<i>messageId</i>	The message ID. It's equal to the return value of SendOutOfDialogMessage function.
<i>fromDisplayName</i>	The display name of message sender.
<i>from</i>	The message sender.
<i>toDisplayName</i>	The display name of message receiver.
<i>to</i>	The message receiver.

void com.portsip.OnPortSIPEvent.onSendOutOfDialogMessageFailure (long *messageId*, String *fromDisplayName*, String *from*, String *toDisplayName*, String *to*, String *reason*, int *code*)

If the message failed to be sent out of dialog, this event would be triggered.

Parameters:

<i>messageId</i>	The message ID. It's equal to the return value of SendOutOfDialogMessage function.
<i>fromDisplayName</i>	The display name of message sender
<i>from</i>	The message sender.
<i>toDisplayName</i>	The display name of message receiver.
<i>to</i>	The message receiver.
<i>reason</i>	The failure reason.
<i>code</i>	The failure code.

Play audio and video file finished events

Functions

- void [com.portsip.OnPortSIPEvent.onPlayAudioFileFinished](#) (long sessionId, String fileName)
- void [com.portsip.OnPortSIPEvent.onPlayVideoFileFinished](#) (long sessionId)

Detailed Description

Function Documentation

void com.portsip.OnPortSIPEvent.onPlayAudioFileFinished (long *sessionId*, String *fileName*)

If called playAudioFileToRemote function with no loop mode, this event will be triggered once the file play finished.

Parameters:

<i>sessionId</i>	The session ID of the call.
<i>fileName</i>	The play file name.

void com.portsip.OnPortSIPEvent.onPlayVideoFileFinished (long *sessionId*)

If called playVideoFileToRemote function with no loop mode, this event will be triggered once the file play finished.

Parameters:

<i>sessionId</i>	The session ID of the call.
------------------	-----------------------------

RTP callback events

Functions

- void [com.portsip.OnPortSIPEvent.onReceivedRTPPacket](#) (long sessionId, boolean isAudio, byte[] RTPPacket, int packetSize)
- void [com.portsip.OnPortSIPEvent.onSendingRTPPacket](#) (long sessionId, boolean isAudio, byte[] RTPPacket, int packetSize)
- void [com.portsip.OnPortSIPEvent.onAudioRawCallback](#) (long sessionId, int enum_audioCallbackMode, byte[] data, int dataLength, int samplingFreqHz)
- void [com.portsip.OnPortSIPEvent.onVideoRawCallback](#) (long sessionId, int enum_videoCallbackMode, int width, int height, byte[] data, int dataLength)
- void [com.portsip.OnPortSIPEvent.onVideoDecodedInfoCallback](#) (long sessionId, int width, int height, int framerate, int bitrate)

Detailed Description

Function Documentation

void com.portsip.OnPortSIPEvent.onReceivedRTPPacket (long sessionId, boolean isAudio, byte [] RTPPacket, int packetSize)

If setRTPCallback function is called to enabled the RTP callback, this event will be triggered once receiving a RTP packet.

Parameters:

<i>sessionId</i>	The session ID of the call.
<i>isAudio</i>	If the received RTP packet is of audio, this parameter returns true; otherwise false.
<i>RTPPacket</i>	The memory of whole RTP packet.
<i>packetSize</i>	The size of received RTP Packet. Remarks

Don't call any SDK API functions in this event directly. If you want to call the API functions or other code which is time-consuming, you should post a message to another thread and execute SDK API functions or other code in another thread.

void com.portsip.OnPortSIPEvent.onSendingRTPPacket (long sessionId, boolean isAudio, byte [] RTPPacket, int packetSize)

If setRTPCallback function is called to enable the RTP callback, this event will be triggered once sending a RTP packet.

Parameters:

<i>sessionId</i>	The session ID of the call.
<i>isAudio</i>	If the received RTP packet is of audio, this parameter returns true; otherwise false.
<i>RTPPacket</i>	The memory of whole RTP packet.
<i>packetSize</i>	The size of received RTP Packet. Remarks

Don't call any SDK API functions in this event directly. If you want to call the API functions or other code which is time-consuming, you should post a message to another thread and execute SDK API functions or other code in another thread.

void com.portsip.OnPortSIPEvent.onAudioRawCallback (long *sessionId*, int *enum_audioCallbackMode*, byte [] *data*, int *dataLength*, int *samplingFreqHz*)

This event will be triggered once receiving the audio packets if called [enableAudioStreamCallback](#) function.

Parameters:

<i>sessionId</i>	The session ID of the call.
<i>enum_audioCallbackMode</i>	The type which is passed in enableAudioStreamCallback function. Below types allowed: ENUM_AUDIOSTREAM_NONE , ENUM_AUDIOSTREAM_LOCAL_MIX , ENUM_AUDIOSTREAM_LOCAL_PER_CHANNEL , ENUM_AUDIOSTREAM_REMOTE_MIX , ENUM_AUDIOSTREAM_REMOTE_PER_CHANNEL .
<i>data</i>	The memory of audio stream. It's in PCM format.
<i>dataLength</i>	The data size.
<i>samplingFreqHz</i>	The audio stream sample in HZ. For example, it's 8000 or 16000. Remarks

Don't call any SDK API functions in this event directly. If you want to call the API functions or other code which is time-consuming, you should post a message to another thread and execute SDK API functions or other code in another thread.

See also:

[PortSipSdk::enableAudioStreamCallback](#)

void com.portsip.OnPortSIPEvent.onVideoRawCallback (long *sessionId*, int *enum_videoCallbackMode*, int *width*, int *height*, byte [] *data*, int *dataLength*)

This event will be triggered once receiving the video packets if [enableVideoStreamCallback](#) function is called.

Parameters:

<i>sessionId</i>	The session ID of the call.
<i>enum_videoCallbackMode</i>	The type which is passed in enableVideoStreamCallback function. Below types allowed: ENUM_VIDEOSTREAM_NONE , ENUM_VIDEOSTREAM_LOCAL , ENUM_VIDEOSTREAM_REMOTE , ENUM_VIDEOSTREAM_BOTH .
<i>width</i>	The width of video image.
<i>height</i>	The height of video image.
<i>data</i>	The memory of video stream. It's in YUV420 format, YV12.
<i>dataLength</i>	The data size.

See also:

[PortSipSdk::enableVideoStreamCallback](#)

void com.portsip.OnPortSIPEvent.onVideoDecodedInfoCallback (long *sessionId*, int *width*, int *height*, int *framerate*, int *bitrate*)

This event will be triggered once receiving the changed video size, given that [enableVideoDecoderCallback](#) function is called.

Parameters:

<i>sessionId</i>	The session ID of the call.
<i>width</i>	The width of received video image.

<i>height</i>	The height of received video image.
<i>framerate</i>	The frame rate value of received video.
<i>bitrate</i>	The bitrate value of received video.

See also:

[PortSipSdk::enableVideoDecoderCallback](#)

SDK functions

Modules

- [Initialize and register functions](#)
- [Audio and video codecs functions](#)
- [Additional settings functions](#)
- [Access SIP message header functions](#)
- [Audio and video functions](#)
- [Call functions](#)
- [Refer functions](#)
- [Send audio and video stream functions](#)
- [RTP packets, Audio stream and video stream callback](#)
- [Record functions](#)
- [Play audio and video file to remote functions](#)
- [Conference functions](#)
- [RTP and RTCP QOS functions](#)
- [RTP statistics functions](#)
- [Audio effect functions](#)
- [Send OPTIONS/INFO/MESSAGE functions](#)
- [Device Management functions.](#)

Detailed Description

Initialize and register functions

Functions

- void [com.portsip.PortSipSdk.CreateCallManager](#) (Context context)
- void [com.portsip.PortSipSdk.DeleteCallManager](#) ()
- int [com.portsip.PortSipSdk.initialize](#) (int enum_transport, int enum_LogLevel, String LogPath, int maxLines, String agent, int audioDeviceLayer, int videoDeviceLayer)
- int [com.portsip.PortSipSdk.setInstanceId](#) (String instanceId)
- int [com.portsip.PortSipSdk.setUser](#) (String userName, String displayName, String authName, String password, String localIP, int localSIPPort, String userDomain, String SIPServer, int SIPServerPort, String STUNServer, int STUNServerPort, String outboundServer, int outboundServerPort)
- int [com.portsip.PortSipSdk.registerServer](#) (int expires, int retryTimes)
- int [com.portsip.PortSipSdk.unregisterServer](#) ()
- int [com.portsip.PortSipSdk.setLicenseKey](#) (String key)

Detailed Description

Function Documentation

void com.portsip.PortSipSdk.CreateCallManager (Context *context*)

Create the callback handlers.

Parameters:

<i>context</i>	The context of application.
----------------	-----------------------------

void com.portsip.PortSipSdk.DeleteCallManager ()

Release the callback Handlers.

int com.portsip.PortSipSdk.initialize (int *enum_transport*, int *enum_LogLevel*, String *LogPath*, int *maxLines*, String *agent*, int *audioDeviceLayer*, int *videoDeviceLayer*)

Initialize the SDK.

Parameters:

<i>enum_transport</i>	Transport for SIP signaling, which can be set as: ENUM_TRANSPORT_UDP , ENUM_TRANSPORT_TCP , ENUM_TRANSPORT_TLS , ENUM_TRANSPORT_PERS . The ENUM_TRANSPORT_PERS is the private PortSIP transport for anti-SIP blocking, which must work with the PERS.
<i>enum_LogLevel</i>	Set the application log level. The SDK will generate "PortSIP_Log_datatime.log" file if the log is enabled. ENUM_LOG_LEVEL_NONE ENUM_LOG_LEVEL_DEBUG ENUM_LOG_LEVEL_ERROR ENUM_LOG_LEVEL_WARNING ENUM_LOG_LEVEL_INFO ENUM_LOG_LEVEL_DEBUG
<i>LogPath</i>	The path for storing log file. The path (folder) specified MUST be existent.
<i>maxLines</i>	Theoretically, unlimited count of lines are supported depending on the device capability. For SIP client, it is recommended to limit it as ranging 1 - 100.
<i>agent</i>	The User-Agent header to be inserted in to SIP messages.
<i>audioDeviceLayer</i>	Specifies the audio device layer that should be using: 0 = Use the OS defaulted device. 1 = Virtual device, usually use this for the device that has no sound device installed. 2 = AndroidOpenSLES, use the OpenSL ES for audio device. This is valid for Android only. If the voice received is bad with this option, please try AndroidAudioTrackJni. 3 = AndroidAudioTrackJni, use Audio Track JNI for audio device. This is valid for Android only. If the voice received is bad with this option, please try AndroidOpenSLES.
<i>videoDeviceLayer</i>	Specifies the video device layer that should be using: 0 = Use the OS defaulted device. 1 = Use Virtual device, usually use this for the device that has no camera installed.

Returns:

If the function succeeds, it returns value 0. If the function fails, it will return a specific error code

int com.portsip.PortSipSdk.setInstanceId (String *instanceId*)

Set the instance Id, the outbound instanceId((RFC5626)) used in contact headers.

Parameters:

<i>instanceId</i>	The SIP instance ID. If this function is not called, the SDK will generate an instance ID automatically. The instance ID MUST be unique on the same device (device ID or IMEI ID is recommended). Recommend to call this function to set the ID on Android devices.
-------------------	---

Returns:

If the function succeeds, it will return value 0. If the function fails, it will return a specific error code.

int com.portsip.PortSipSdk.setUser (String *userName*, String *displayName*, String *authName*, String *password*, String *localIP*, int *localSIPPort*, String *userDomain*, String *SIPServer*, int *SIPServerPort*, String *STUNServer*, int *STUNServerPort*, String *outboundServer*, int *outboundServerPort*)

Set user account info.

Parameters:

<i>userName</i>	Account (username) of the SIP, usually provided by an IP-Telephony service provider.
<i>displayName</i>	The name displayed. You can set it as your like, such as "James Kend". It's optional.
<i>authName</i>	Authorization user name (usually equal to the username).
<i>password</i>	User's password. It's optional.
<i>localIP</i>	The local computer IP address (for example: 192.168.1.108), it will be used for sending and receiving SIP messages and RTP packets. If the local IP is provided with IPv6 format, the SDK will be using IPv6. If you want the SDK to choose correct network interface (IP) automatically, please pass "0.0.0.0" for IPv4, or "::" for IPv6.
<i>localSIPPort</i>	The listening port for SIP message transmission, for example 5060.
<i>userDomain</i>	User domain; this parameter is optional, which allows to transfer an empty string if you are not using the domain.
<i>SIPServer</i>	SIP proxy server IP or domain, for example xx.xxx.xx.x or sip.xxx.com.
<i>SIPServerPort</i>	Port of the SIP proxy server, for example 5060.
<i>STUNServer</i>	Stun server for NAT traversal. It's optional and can be used to transfer empty string to disable STUN.
<i>STUNServerPort</i>	STUN server port, it will be ignored if the outboundServer is empty.
<i>outboundServer</i>	Outbound proxy server, for example sip.domain.com. It's optional and allows to transfer an empty string if not using the outbound server.
<i>outboundServerPort</i>	Outbound proxy server port, it will be ignored if the outboundServer is empty.

Returns:

If this function succeeds, it will return value 0. If it fails, it will return a specific error code.

int com.portsip.PortSipSdk.registerServer (int *expires*, int *retryTimes*)

Register to SIP proxy server (login to server)

Parameters:

<i>expires</i>	Time interval for registration refreshment, in seconds. The maximum of supported value is 3600. It will be inserted into SIP REGISTER message headers.
<i>retryTimes</i>	The maximum of retry attempts if failed to refresh the registration. By setting

	to ≤ 0 , the attempt will be disabled and onRegisterFailure callback will be triggered when facing retry failure.
--	--

Returns:

If this function succeeds, it will return value 0. If fails, it will return a specific error code. If the registration to server succeeds, onRegisterSuccess will be triggered; otherwise onRegisterFailure will be triggered.

int com.portsip.PortSipSdk.unregisterServer ()

Un-register from the SIP proxy server.

Returns:

If this function succeeds, it will return value 0. If the function fails, it will return a specific error code.

int com.portsip.PortSipSdk.setLicenseKey (String key)

Set the license key. It must be called before setUser function.

Parameters:

<i>key</i>	The SDK license key. Please purchase from PortSIP.
------------	--

Returns:

If the function succeeds, it will return value 0. If the function fails, it will return a specific error code.

Audio and video codecs functions

Functions

- int [com.portsip.PortSipSdk.addAudioCodec](#) (int enum_audiocodec)
- int [com.portsip.PortSipSdk.addVideoCodec](#) (int enum_videocodec)
- boolean [com.portsip.PortSipSdk.isAudioCodecEmpty](#) ()
- boolean [com.portsip.PortSipSdk.isVideoCodecEmpty](#) ()
- int [com.portsip.PortSipSdk.setAudioCodecPayloadType](#) (int enum_audiocodec, int payloadType)
- int [com.portsip.PortSipSdk.setVideoCodecPayloadType](#) (int enum_videocodec, int payloadType)
- void [com.portsip.PortSipSdk.clearAudioCodec](#) ()
- void [com.portsip.PortSipSdk.clearVideoCodec](#) ()
- int [com.portsip.PortSipSdk.setAudioCodecParameter](#) (int enum_audiocodec, String sdpParameter)
- int [com.portsip.PortSipSdk.setVideoCodecParameter](#) (int enum_videocodec, String sdpParameter)

Detailed Description

Function Documentation

int com.portsip.PortSipSdk.addAudioCodec (int enum_audiocodec)

Enable an audio codec, and it will be shown in SDP.

Parameters:

<i>enum_audiocodec</i>	Audio codec type.allow :
ENUM_AUDIOCODEC_G729 ,	ENUM_AUDIOCODEC_PCMA ,
ENUM_AUDIOCODEC_PCMU ,	ENUM_AUDIOCODEC_GSM ,
ENUM_AUDIOCODEC_G722 ,	ENUM_AUDIOCODEC_ILBC ,
ENUM_AUDIOCODEC_AMR ,	ENUM_AUDIOCODEC_AMRWB ,
ENUM_AUDIOCODEC_SPEEX ,	ENUM_AUDIOCODEC_SPEEXWB ,
ENUM_AUDIOCODEC_ISACWB ,	ENUM_AUDIOCODEC_ISACSWB ,
ENUM_AUDIOCODEC_OPUS , ENUM_AUDIOCODEC_DTMF	

Returns:

If the function succeeds, it will return value 0. If the function fails, it will return a specific error code.

int com.portsip.PortSipSdk.addVideoCodec (int *enum_videocodec*)

Enable a video codec, and it will be shown in SDP.

Parameters:

<i>enum_videocodec</i>	Video codec type. Supported types include: ENUM_VIDEOCODEC_H263 , ENUM_VIDEOCODEC_H263_1998 , ENUM_VIDEOCODEC_H264 , ENUM_VIDEOCODEC_VP8 .
------------------------	--

Returns:

If the function succeeds, it will return value 0. If the function fails, it will return a specific error code.

boolean com.portsip.PortSipSdk.isAudioCodecEmpty ()

Detect if the audio codecs are enabled.

Returns:

If no audio codec enabled, it will return value true; otherwise it returns false.

boolean com.portsip.PortSipSdk.isVideoCodecEmpty ()

Detect if the video codecs are enabled.

Returns:

If no video codec enabled, it will return value true; otherwise it returns false.

int com.portsip.PortSipSdk.setAudioCodecPayloadType (int *enum_audiocodec*, int *payloadType*)

Set the RTP payload type for dynamic audio codec.

Parameters:

<i>enum_audiocodec</i>	Audio codec types. Supported types include: ENUM_AUDIOCODEC_G729 , ENUM_AUDIOCODEC_PCMA , ENUM_AUDIOCODEC_PCMU , ENUM_AUDIOCODEC_GSM , ENUM_AUDIOCODEC_G722 , ENUM_AUDIOCODEC_ILBC , ENUM_AUDIOCODEC_AMR , ENUM_AUDIOCODEC_AMRWB , ENUM_AUDIOCODEC_SPEEX , ENUM_AUDIOCODEC_SPEEXWB , ENUM_AUDIOCODEC_ISACWB , ENUM_AUDIOCODEC_ISACSWB , ENUM_AUDIOCODEC_OPUS , ENUM_AUDIOCODEC_DTMF
<i>payloadType</i>	The new RTP payload type that you want to set.

Returns:

If the function succeeds, it will return value 0. If the function fails, it will return a specific error code.

int com.portsip.PortSipSdk.setVideoCodecPayloadType (int *enum_videocodec*, int *payloadType*)

Set the RTP payload type for dynamic video codec.

Parameters:

<i>enum_videocodec</i>	Video codec type. Supported types include: ENUM_VIDEOCODEC_H263 , ENUM_VIDEOCODEC_H263_1998 , ENUM_VIDEOCODEC_H264 , ENUM_VIDEOCODEC_VP8 .
<i>payloadType</i>	The new RTP payload type that you want to set.

Returns:

If the function succeeds, it will return value 0. If the function fails, it will return a specific error code.

void com.portsip.PortSipSdk.clearAudioCodec ()

Remove all the enabled audio codecs.

void com.portsip.PortSipSdk.clearVideoCodec ()

Remove all the enabled video codecs.

int com.portsip.PortSipSdk.setAudioCodecParameter (int *enum_audiocodec*, String *sdpParameter*)

Set the codec parameter for audio codec.

Parameters:

<i>enum_audiocodec</i>	Audio codec type. Supported types include: ENUM_AUDIOCODEC_G729 , ENUM_AUDIOCODEC_PCMA , ENUM_AUDIOCODEC_PCMU , ENUM_AUDIOCODEC_GSM , ENUM_AUDIOCODEC_G722 , ENUM_AUDIOCODEC_ILBC , ENUM_AUDIOCODEC_AMR , ENUM_AUDIOCODEC_AMRWB , ENUM_AUDIOCODEC_SPEEX , ENUM_AUDIOCODEC_SPEEXWB , ENUM_AUDIOCODEC_ISACWB , ENUM_AUDIOCODEC_ISACSWB , ENUM_AUDIOCODEC_OPUS , ENUM_AUDIOCODEC_DTMF .
<i>sdpParameter</i>	The parameter is in string format.

Returns:

If the function succeeds, it will return value 0. If the function fails, it will return a specific error code.

See also:

[PortSipEnumDefine](#)

Remarks:

Example:

```
setAudioCodecParameter(AUDIOCODEC_AMR, "mode-set=0; octet-align=1; robust-sorting=0")
```

int com.portsip.PortSipSdk.setVideoCodecParameter (int *enum_videocodec*, String *sdpParameter*)

Set the codec parameter for video codec.

Parameters:

<i>enum_videocodec</i>	Video codec types. Supported types include: ENUM_VIDEOCODEC_H263 , ENUM_VIDEOCODEC_H263_1998 , ENUM_VIDEOCODEC_H264 , ENUM_VIDEOCODEC_VP8 .
<i>sdpParameter</i>	The parameter is in string format.

Returns:

If the function succeeds, it will return value 0. If the function fails, it will return a specific error code.

Remarks:

Example:

```
setVideoCodecParameter (PortSipEnumDefine.ENUM_VIDEOCODEC_H264,  
"profile-level-id=420033; packetization-mode=0");
```

Additional settings functions

Functions

- int [com.portsip.PortSipSdk.setDisplayName](#) (String displayName)
- String [com.portsip.PortSipSdk.getVersion](#) ()
- int [com.portsip.PortSipSdk.enableReliableProvisional](#) (boolean enable)
- int [com.portsip.PortSipSdk.enable3GppTags](#) (boolean enable)
- void [com.portsip.PortSipSdk.enableCallbackSendingSignaling](#) (boolean enable)
- void [com.portsip.PortSipSdk.setSrtpPolicy](#) (int enum_srtpolicy)
- int [com.portsip.PortSipSdk.setRtpPortRange](#) (int minimumRtpAudioPort, int maximumRtpAudioPort, int minimumRtpVideoPort, int maximumRtpVideoPort)
- int [com.portsip.PortSipSdk.setRtcpPortRange](#) (int minimumRtcpAudioPort, int maximumRtcpAudioPort, int minimumRtcpVideoPort, int maximumRtcpVideoPort)
- int [com.portsip.PortSipSdk.enableCallForward](#) (boolean forBusyOnly, String forwardTo)
- int [com.portsip.PortSipSdk.disableCallForward](#) ()
- int [com.portsip.PortSipSdk.enableSessionTimer](#) (int timerSeconds)
- void [com.portsip.PortSipSdk.disableSessionTimer](#) ()
- void [com.portsip.PortSipSdk.setDoNotDisturb](#) (boolean forBusyOnly)
- int [com.portsip.PortSipSdk.detectMwi](#) ()
- int [com.portsip.PortSipSdk.enableCheckMwi](#) (boolean state)
- int [com.portsip.PortSipSdk.setRtpKeepAlive](#) (boolean state, int keepAlivePayloadType, int deltaTransmitTimeMS)
- int [com.portsip.PortSipSdk.setKeepAliveTime](#) (int keepAliveTime)
- int [com.portsip.PortSipSdk.setAudioSamples](#) (int ptime, int maxptime)
- int [com.portsip.PortSipSdk.addSupportedMimeType](#) (String methodName, String mimeType, String subMimeType)

Detailed Description

Function Documentation

int com.portsip.PortSipSdk.setDisplayName (String *displayName*)

Set presentation name for user.

Parameters:

<i>displayName</i>	The presentation name.
--------------------	------------------------

Returns:

If the function succeeds, it will return value 0. If the function fails, it will return a specific error code.

String com.portsip.PortSipSdk.getVersion ()

Get the version number of the current SDK.

Returns:

String with version description

int com.portsip.PortSipSdk.enableReliableProvisional (boolean *enable*)

Enable/Disable PRACK.

Parameters:

<i>enable</i>	Set to true to enable the SDK support PRACK. In default the PRACK is disabled.
---------------	--

Returns:

If the function succeeds, it will return value 0. If the function fails, it will return a specific error code.

int com.portsip.PortSipSdk.enable3GppTags (boolean *enable*)

Enable/disable the 3Gpp tags, including "ims.icsi.mmtel" and "g.3gpp.smsip".

Parameters:

<i>enable</i>	Set to true to enable 3Gpp tags for SDK.
---------------	--

Returns:

If the function succeeds, it will return value 0. If the function fails, it will return a specific error code.

void com.portsip.PortSipSdk.enableCallbackSendingSignaling (boolean *enable*)

Enable/disable the callback of the sent SIP messages.

Parameters:

<i>enable</i>	Set as true to enable the callback of the sent SIP messages, or false to disable it. Once enabled, the "onSendingSignaling" event will be triggered once the SDK sent a SIP message.
---------------	--

void com.portsip.PortSipSdk.setSrtpPolicy (int *enum_srtpolicy*)

Set the SRTP policy.

Parameters:

<i>enum_srtpolicy</i>	The SRTP policy.allow: ENUM_SRTTPOLICY_NONE , ENUM_SRTTPOLICY_FORCE , ENUM_SRTTPOLICY_PREFER .
-----------------------	--

int com.portsip.PortSipSdk.setRtpPortRange (int *minimumRtpAudioPort*, int *maximumRtpAudioPort*, int *minimumRtpVideoPort*, int *maximumRtpVideoPort*)

This function allows to set the RTP port range for audio and video streaming.

Parameters:

<i>minimumRtpAudioPort</i>	The minimum RTP port for audio stream.
<i>maximumRtpAudioPort</i>	The maximum RTP port for audio stream.
<i>minimumRtpVideoPort</i>	The minimum RTP port for video stream.
<i>maximumRtpVideoPort</i>	The maximum RTP port for video stream.

Returns:

If the function succeeds, it will return value 0. If the function fails, it will return a specific error code.

Remarks:

The port range ((max - min) % maxCallLines) should be greater than 4.

int com.portsip.PortSipSdk.setRtcpPortRange (int *minimumRtcpAudioPort*, int *maximumRtcpAudioPort*, int *minimumRtcpVideoPort*, int *maximumRtcpVideoPort*)

This function allows to set the RTCP port range for audio and video streaming.

Parameters:

<i>minimumRtcpAudioPort</i>	The minimum RTCP port for audio stream.
<i>maximumRtcpAudioPort</i>	The maximum RTCP port for audio stream.
<i>minimumRtcpVideoPort</i>	The minimum RTCP port for video stream.
<i>maximumRtcpVideoPort</i>	The maximum RTCP port for video stream.

Returns:

If the function succeeds, it will return value 0. If the function fails, it will return a specific error code.

Remarks:

The port range ((max - min) % maxCallLines) should be greater than 4.

int com.portsip.PortSipSdk.enableCallForward (boolean *forBusyOnly*, String *forwardTo*)

Enable call forwarding.

Parameters:

<i>forBusyOnly</i>	If this parameter is set to true, the SDK will forward incoming calls when the user is currently busy. If set it to false, SDK will forward all incoming calls.
<i>forwardTo</i>	The target to which the call will be forwarded. It must be in the format of sip: xxxx@sip.portsip.com .

Returns:

If the function succeeds, it will return value 0. If the function fails, it will return a specific error code.

int com.portsip.PortSipSdk.disableCallForward ()

Disable the call forwarding. The SDK will not forward any incoming call when this function is called.

Returns:

If the function succeeds, it will not return value 0. If the function fails, it will return a specific error code.

int com.portsip.PortSipSdk.enableSessionTimer (int *timerSeconds*)

This function allows to periodically refresh Session Initiation Protocol (SIP) sessions by sending repeated INVITE requests.

Parameters:

<i>timerSeconds</i>	The value of the refresh interval in seconds. A minimum of 90 seconds required.
---------------------	---

Returns:

If the function succeeds, it will return value 0. If the function fails, it will return a specific error code.

Remarks:

The repeated INVITE requests, or re-INVITES, are sent during an active call log to allow user agents (UA) or proxies to determine the status of a SIP session. Without this keep-alive mechanism, proxies that remember incoming and outgoing requests (stateful proxies) may continue to retain call state in vain. If a UA fails to send a BYE message at the end of a session, or if the BYE message is lost due to network problems, a stateful proxy will not know that the session has ended. The re-INVITES ensure that active sessions stay active and completed sessions are terminated.

void com.portsip.PortSipSdk.disableSessionTimer ()

Disable the session timer.

void com.portsip.PortSipSdk.setDoNotDisturb (boolean *forBusyOnly*)

Enable/disable the "Do not disturb" status.

Parameters:

<i>forBusyOnly</i>	If it is set to true, the SDK will reject all incoming calls.
--------------------	---

int com.portsip.PortSipSdk.detectMwi ()

Used for obtaining MWI status.

Returns:

If the function succeeds, it will return value 0. If the function fails, it will return a specific error code.

int com.portsip.PortSipSdk.enableCheckMwi (boolean *state*)

Enable/disable to access MWI (Message Waiting Indication).

Parameters:

<i>state</i>	If it is set to true, MWI will be checked automatically once successfully registered to a SIP proxy server;
--------------	---

Returns:

If the function succeeds, it will return value 0. If the function fails, it will return a specific error code.

int com.portsip.PortSipSdk.setRtpKeepAlive (boolean *state*, int *keepAlivePayloadType*, int *deltaTransmitTimeMS*)

Enable or disable to send RTP keep-alive packet when the call is ongoing.

Parameters:

<i>state</i>	When it's set to true, it's allowed to send the keep-alive packet during the conversation;
<i>keepAlivePayloadType</i>	The payload type of the keep-alive RTP packet. It's usually set to 126.
<i>deltaTransmitTimeMS</i>	The interval for sending keep-alive RTP packet, in millisecond. Recommended value ranges 15000 - 300000.

Returns:

If the function succeeds, it will return value 0. If the function fails, it will return a specific error code.

int com.portsip.PortSipSdk.setKeepAliveTime (int *keepAliveTime*)

Enable or disable to send SIP keep-alive packet.

Parameters:

<i>keepAliveTime</i>	This is the time interval for SIP keep-alive, in seconds. When it is set to 0, the SIP keep-alive will be disabled. Recommended value is 30 or 50.
----------------------	--

Returns:

If the function succeeds, it will return value 0. If the function fails, it will return a specific error code.

int com.portsip.PortSipSdk.setAudioSamples (int *ptime*, int *maxptime*)

Set the audio capture sample which will be present in the SDP of INVITE and 200 OK message as "ptime" and "maxptime" attribute.

Parameters:

<i>ptime</i>	It's should be a multiple of 10 between 10 - 60 (included 10 and 60).
<i>maxptime</i>	The "maxptime" attribute should be a multiple of 10 between 10 - 60 (included 10 and 60). It can't be less than "ptime".

Returns:

If the function succeeds, it will return value 0. If the function fails, it will return a specific error code.

int com.portsip.PortSipSdk.addSupportedMimeType (String *methodName*, String *mimeType*, String *subMimeType*)

Set the SDK to receive SIP messages that include special mime type.

Parameters:

<i>methodName</i>	Method name of the SIP message, such as INVITE, OPTION, INFO, MESSAGE, UPDATE, ACK etc. For more details please refer to RFC3261.
<i>mimeType</i>	The mime type of SIP message.
<i>subMimeType</i>	The sub mime type of SIP message.

Returns:

If the function succeeds, it will return value 0. If the function fails, it will return a specific error code.

Remarks:

In default, PortSIP VoIP SDK supports media types (mime types) included in the below incoming SIP messages:

```
"message/sipfrag" in NOTIFY message.  
"application/simple-message-summary" in NOTIFY message.  
"text/plain" in MESSAGE message. "application/dtmf-relay" in INFO  
message. <br> "application/media_control+xml" in INFO message.
```

The SDK allows to receive SIP messages that include above mime types. Now if remote side send an INFO SIP message with its "Content-Type" header value "text/plain", SDK will reject this INFO message, because "text/plain" of INFO message is not included in the default type list. How should we enable the SDK to receive SIP INFO messages that include "text/plain" mime type? The answer is `addSupportedMimyType`:

```
addSupportedMimeType("INFO", "text", "plain");
```

If the user wishes to receive the NOTIFY message with "application/media_control+xml", it should be set as below:

```
addSupportedMimeType("NOTIFY", "application", "media_control+xml");
```

For more details about the mime type, please visit:
<http://www.iana.org/assignments/media-types/>

Access SIP message header functions

Functions

- String [com.portsip.PortSipSdk.getExtensionHeaderValue](#) (String sipMessage, String headerName)
- int [com.portsip.PortSipSdk.addExtensionHeader](#) (String headerName, String headerValue)
- int [com.portsip.PortSipSdk.clearAddExtensionHeaders](#) ()
- int [com.portsip.PortSipSdk.modifyHeaderValue](#) (String headerName, String headerValue)
- int [com.portsip.PortSipSdk.clearModifyHeaders](#) ()

Detailed Description

Function Documentation

String [com.portsip.PortSipSdk.getExtensionHeaderValue](#) (String *sipMessage*, String *headerName*)

Access the SIP header of SIP message.

Parameters:

<i>sipMessage</i>	The SIP message.
<i>headerName</i>	The header of which user wishes to access the SIP message.

Returns:

String. The SIP header of SIP message.

int com.portsip.PortSipSdk.addExtensionHeader (String *headerName*, String *headerValue*)

Add the extension header (custom header) into every outgoing SIP message.

Parameters:

<i>headerName</i>	The custom header name which will be present in every outgoing SIP message.
<i>headerValue</i>	The custom header value.

Returns:

If the function succeeds, it will return value 0. If the function fails, it will return a specific error code.

int com.portsip.PortSipSdk.clearAddExtensionHeaders ()

Clear the added extension headers (custom headers)

Returns:

If the function succeeds, it will return value 0. If the function fails, it will return a specific error code.

Remarks:

For example, we have added two custom headers into every outgoing SIP message and want to have them removed.

```
addExtensionHeader("Billing", "usd100.00");  
addExtensionHeader("ServiceId", "8873456");  
clearAddextensionHeaders();
```

If this function is called, the added extension headers will no longer appear in outgoing SIP message.

int com.portsip.PortSipSdk.modifyHeaderValue (String *headerName*, String *headerValue*)

Modify the special SIP header value for every outgoing SIP message.

Parameters:

<i>headerName</i>	The SIP header name for which the value will be modified.
<i>headerValue</i>	The header value that needs to be modified.

Returns:

If the function succeeds, it will return value 0. If the function fails, it will return a specific error code.

Remarks:

Example: modify "Expires" header and "User-Agent" header value for every outgoing SIP message:

```
modifyHeaderValue("Expires", "1000");  
modifyHeaderValue("User-Agent", "MyTest Softphone 1.0");
```

int com.portsip.PortSipSdk.clearModifyHeaders ()

Clear the modify headers value. Once cleared, it will no longer modify every outgoing SIP message header values.

Returns:

If the function succeeds, it will return value 0. If the function fails, it will return a specific error code.

Remarks:

Example: modify two headers value for every outgoing SIP message and then clear it:

```
modifyHeaderValue("Expires", "1000");
modifyHeaderValue("User-Agent", "MyTest Softphone 1.0");
clearModifyHeaders();
```

Audio and video functions

Functions

- int [com.portsip.PortSipSdk.setVideoDeviceId](#) (int deviceId)
- int [com.portsip.PortSipSdk.setVideoResolution](#) (int width, int height)
- int [com.portsip.PortSipSdk.setAudioBitrate](#) (long sessionId, int enum_codec, int bitrateKbps)
- int [com.portsip.PortSipSdk.setVideoBitrate](#) (int bitrateKbps)
- int [com.portsip.PortSipSdk.setVideoFrameRate](#) (int frameRate)
- int [com.portsip.PortSipSdk.sendVideo](#) (long sessionId, boolean send)
- int [com.portsip.PortSipSdk.setVideoOrientation](#) (int enum_rotation)
- void [com.portsip.PortSipSdk.setLocalVideoWindow](#) (Object localVideoView)
- int [com.portsip.PortSipSdk.setRemoteVideoWindow](#) (long sessionId, Object remoteVideoView)
- void [com.portsip.PortSipSdk.displayLocalVideo](#) (boolean state)
- int [com.portsip.PortSipSdk.setVideoNackStatus](#) (boolean state)
- void [com.portsip.PortSipSdk.muteMicrophone](#) (boolean mute)
- void [com.portsip.PortSipSdk.muteSpeaker](#) (boolean mute)
- int [com.portsip.PortSipSdk.getDynamicSpeakerVolumeLevel](#) ()
- int [com.portsip.PortSipSdk.getDynamicMicrophoneVolumeLevel](#) ()
- int [com.portsip.PortSipSdk.setChannelOutputVolumeScaling](#) (long sessionId, int scaling)
- int [com.portsip.PortSipSdk.setLoudspeakerStatus](#) (boolean useSpeaker)

Detailed Description

Function Documentation

int com.portsip.PortSipSdk.setVideoDeviceId (int *deviceId*)

Set the video device that will be used for video call.

Parameters:

<i>deviceId</i>	Device ID (index) for video device (camera).
-----------------	--

Returns:

If the function succeeds, it will return value 0. If the function fails, it will return a specific error code.

int com.portsip.PortSipSdk.setVideoResolution (int *width*, int *height*)

Set the video capturing resolution.

Parameters:

<i>width</i>	Video resolution, width
<i>height</i>	Video resolution, height

Returns:

If the function succeeds, it will return value 0. If the function fails, it will return a specific error code.

int com.portsip.PortSipSdk.setAudioBitrate (long *sessionId*, int *enum_audiocodec*, int *bitrateKbps*)

Set the audio bitrate.

Parameters:

<i>sessionId</i>	The session ID of the call.
<i>enum_audiocodec</i>	Audio codec type allowed: ENUM_AUDIOCODEC_OPUS
<i>bitrateKbps</i>	The Audio bitrate in KBPS.

Returns:

If the function succeeds, it will return value 0. If the function fails, it will return a specific error code.

int com.portsip.PortSipSdk.setVideoBitrate (int *bitrateKbps*)

Set the video bitrate.

Parameters:

<i>bitrateKbps</i>	The video bitrate in KBPS.
--------------------	----------------------------

Returns:

If the function succeeds, it will return value 0. If the function fails, it will return a specific error code.

int com.portsip.PortSipSdk.setVideoFrameRate (int *frameRate*)

Set the video frame rate. Usually you do not need to call this function to set the frame rate since the SDK uses default frame rate.

Parameters:

<i>frameRate</i>	The frame rate value, with its minimum of 5, and maximum value of 30. The greater the value is, the better video quality enabled and more bandwidth required;
------------------	---

Returns:

If the function succeeds, it will return value 0. If the function fails, it will return a specific error code.

int com.portsip.PortSipSdk.sendVideo (long *sessionId*, boolean *send*)

Send the video to remote side.

Parameters:

<i>sessionId</i>	The session ID of the call.
<i>send</i>	Set to true to send the video, or false to stop sending.

Returns:

If the function succeeds, it will return value 0. If the function fails, it will return a specific error code.

int com.portsip.PortSipSdk.setVideoOrientation (int *enum_rotation*)

Change the orientation of the video.

Parameters:

<i>enum_rotation</i>	The video rotation that you want to set. Supported rotation include: ENUM_ROTATE_CAPTURE_FRAME_0 , ENUM_ROTATE_CAPTURE_FRAME_90 , ENUM_ROTATE_CAPTURE_FRAME_180 , ENUM_ROTATE_CAPTURE_FRAME_270 .
----------------------	---

Returns:

If the function succeeds, it will return value 0. If the function fails, it will return a specific error code.

void com.portsip.PortSipSdk.setLocalVideoWindow (Object *localVideoView*)

Set the window that is used for displaying the local video image.

Parameters:

<i>localVideoView</i>	SurfaceView a SurfaceView for displaying local video image from camera.
-----------------------	---

int com.portsip.PortSipSdk.setRemoteVideoWindow (long *sessionId*, Object *remoteVideoView*)

Set the window for a session that is used for displaying the received remote video image.

Parameters:

<i>sessionId</i>	The session ID of the call.
<i>remoteVideoView</i>	SurfaceView a SurfaceView for displaying the received remote video image.

Returns:

If the function succeeds, it will return value 0. If the function fails, it will return a specific error code.

void com.portsip.PortSipSdk.displayLocalVideo (boolean *state*)

Start/stop displaying the local video image.

Parameters:

<i>state</i>	Set to true to display local video iamge.
--------------	---

int com.portsip.PortSipSdk.setVideoNackStatus (boolean *state*)

Enable/disable the NACK feature (rfc6642) which helps to improve the video quality.

Parameters:

<i>state</i>	Set to true to enable.
--------------	------------------------

Returns:

If the function succeeds, it will return value 0. If the function fails, it will return a specific error code.

void com.portsip.PortSipSdk.muteMicrophone (boolean *mute*)

Mute the device microphone.

Parameters:

<i>mute</i>	If the value is set to true, the microphone is muted; user could also set it to false to un-mute it.
-------------	--

void com.portsip.PortSipSdk.muteSpeaker (boolean *mute*)

Mute the device speaker. This feature is unavailable for Android and iOS.

Parameters:

<i>mute</i>	If the value is set to true, the speaker is muted; user could also set it to false to un-mute it.
-------------	---

int com.portsip.PortSipSdk.getDynamicSpeakerVolumeLevel ()

Obtain the dynamic microphone volume level from current call. We usually set a timer to call this function to refresh the volume level indicator.

Returns:

The dynamic speaker volume by this parameter with the range 0 - 9.

int com.portsip.PortSipSdk.getDynamicMicrophoneVolumeLevel ()

Obtain the dynamic microphone volume level from current call. Usually set a timer to call this function to refresh the volume level indicator.

Returns:

The dynamic microphone volume by this parameter with the range 0 - 9.

int com.portsip.PortSipSdk.setChannelOutputVolumeScaling (long *sessionId*, int *scaling*)

Set a volume |scaling| to be applied to the outgoing signal of a specific audio channel.

Parameters:

<i>sessionId</i>	The session ID of the call.
<i>scaling</i>	Valid scale ranges [0, 1000]. Default is 100.

Returns:

If the function succeeds, it will return value 0. If the function fails, it will return a specific error code.

int com.portsip.PortSipSdk.setLoudspeakerStatus (boolean *useSpeaker*)

Set the audio device that will be used for audio call. For Android and iOS, switch between earphone and Loudspeaker allowed.

Parameters:

<i>useSpeaker</i>	Set to true the SDK use loudspeaker for audio call, this is just available for mobile platform only.
-------------------	--

Returns:

If the function succeeds, it will return value 0. If the function fails, it will return a specific error code.

Call functions

Functions

- long [com.portsip.PortSipSdk.call](#) (String callee, boolean sendSdp, boolean videoCall)
- int [com.portsip.PortSipSdk.rejectCall](#) (long sessionId, int code)
- int [com.portsip.PortSipSdk.hangUp](#) (long sessionId)
- int [com.portsip.PortSipSdk.answerCall](#) (long sessionId, boolean videoCall)
- int [com.portsip.PortSipSdk.updateCall](#) (long sessionId, boolean enableAudio, boolean enableVideo)
- int [com.portsip.PortSipSdk.hold](#) (long sessionId)

- int [com.portsip.PortSipSdk.unHold](#) (long sessionId)
- int [com.portsip.PortSipSdk.muteSession](#) (long sessionId, boolean muteIncomingAudio, boolean muteOutgoingAudio, boolean muteIncomingVideo, boolean muteOutgoingVideo)
- int [com.portsip.PortSipSdk.forwardCall](#) (long sessionId, String forwardTo)
- int [com.portsip.PortSipSdk.sendDtmf](#) (long sessionId, int enum_dtmfMethod, int code, int dtmfDuration, boolean playDtmfTone)

Detailed Description

Function Documentation

long com.portsip.PortSipSdk.call (String *callee*, boolean *sendSdp*, boolean *videoCall*)

Make a call

Parameters:

<i>callee</i>	The callee. It can be a name only or full SIP URI, for example: user001 or sip: user001@sip.iptel.org or sip: user002@sip.yourdomain.com :5068
<i>sendSdp</i>	If it is set to false, the outgoing call will not include the SDP in INVITE message.
<i>videoCall</i>	If it is set to true and at least one video codec was added, the outgoing call will include the video codec into SDP. Otherwise no video codec will be added into outgoing SDP.

Returns:

If the function succeeds, it will return the session ID of the call, which is greater than 0. If the function fails, it will return a specific error code.

Note: the function success just means the outgoing call is processing, you need to detect the call final state in onInviteTrying, onInviteRinging, onInviteFailure callback events.

int com.portsip.PortSipSdk.rejectCall (long *sessionId*, int *code*)

rejectCall Reject the incoming call.

Parameters:

<i>sessionId</i>	The session ID of the call.
<i>code</i>	Reject code, for example, 486, 480 etc.

Returns:

If the function succeeds, it will return value 0. If the function fails, it will return a specific error code.

int com.portsip.PortSipSdk.hangUp (long *sessionId*)

hangUp Hang up the call.

Parameters:

<i>sessionId</i>	Session ID of the call.
------------------	-------------------------

Returns:

If the function succeeds, it will return value 0. If the function fails, it will return a specific error code.

int com.portsip.PortSipSdk.answerCall (long *sessionId*, boolean *videoCall*)

answerCall Answer the incoming call.

Parameters:

<i>sessionId</i>	The session ID of call.
<i>videoCall</i>	If the incoming call is a video call and the video codec is matched, set to true to answer the video call. If set to false, the answer call doesn't include video codec answer anyway.

Returns:

If the function succeeds, it will return value 0. If the function fails, it will return a specific error code.

int com.portsip.PortSipSdk.updateCall (long *sessionId*, boolean *enableAudio*, boolean *enableVideo*)

updateCall Use the re-INVITE to update the established call.

Parameters:

<i>sessionId</i>	The session ID of call.
<i>enableAudio</i>	Set to true to allow the audio in updated call, or false to disable audio in updated call.
<i>enableVideo</i>	Set to true to allow the video in update call, or false to disable video in updated call.

Returns:

If the function succeeds, it will return value 0. If the function fails, it will return specific error code.

Remarks:

Example usage:

Example 1: A called B with the audio only, and B answered A, there would be an audio conversation between A and B. Now A want to see B through video, A could use these functions to fulfill it.

```
clearVideoCodec();  
addVideoCodec(VIDEOCODEC H264);  
updateCall(sessionId, true, true);
```

Example 2: Remove video stream from the current conversation.

```
updateCall(sessionId, true, false);
```

int com.portsip.PortSipSdk.hold (long *sessionId*)

To place a call on hold.

Parameters:

<i>sessionId</i>	The session ID of call.
------------------	-------------------------

Returns:

If the function succeeds, it will return value 0. If the function fails, it will return a specific error code.

int com.portsip.PortSipSdk.unHold (long *sessionId*)

Take off hold.

Parameters:

<i>sessionId</i>	The session ID of call.
------------------	-------------------------

Returns:

If the function succeeds, it will return value 0. If the function fails, it will return a specific error code.

int com.portsip.PortSipSdk.muteSession (long *sessionId*, boolean *muteIncomingAudio*, boolean *muteOutgoingAudio*, boolean *muteIncomingVideo*, boolean *muteOutgoingVideo*)

Mute the specified audio or video session.

Parameters:

<i>sessionId</i>	The session ID of the call.
<i>muteIncomingAudio</i>	Set it to true to mute incoming audio stream. Once set, remote side audio cannot be heard.
<i>muteOutgoingAudio</i>	Set it to true to mute outgoing audio stream. Once set, the remote side cannot hear the audio.
<i>muteIncomingVideo</i>	Set it to true to mute incoming video stream. Once set, remote side video cannot be seen.
<i>muteOutgoingVideo</i>	Set it to true to mute outgoing video stream, the remote side cannot see the video.

Returns:

If the function succeeds, it will return value 0. If the function fails, it will return a specific error code.

int com.portsip.PortSipSdk.forwardCall (long *sessionId*, String *forwardTo*)

Forward call to another one when receiving the incoming call.

Parameters:

<i>sessionId</i>	The session ID of the call.
<i>forwardTo</i>	Target of the forward. It can be either "sip:number@sipserver.com" or "number".

Returns:

If the function succeeds, it will return value 0. If the function fails, it will return value a specific error code.

int com.portsip.PortSipSdk.sendDtmf (long *sessionId*, int *enum_dtmfMethod*, int *code*, int *dtmfDuration*, boolean *playDtmfTone*)

Send DTMF tone.

Parameters:

<i>sessionId</i>	The session ID of the call.
<i>enum_dtmfMethod</i>	DTMF tone could be sent via two methods: DTMF_RFC2833 or DTMF_INFO. The DTMF_RFC2833 is recommend.
<i>code</i>	The DTMF tone. Values include:

code	Description
0	The DTMF tone 0.
1	The DTMF tone 1.
2	The DTMF tone 2.
3	The DTMF tone 3.
4	The DTMF tone 4.
5	The DTMF tone 5.
6	The DTMF tone 6.
7	The DTMF tone 7.

8	The DTMF tone 8.
9	The DTMF tone 9.
10	The DTMF tone *.
11	The DTMF tone #.
12	The DTMF tone A.
13	The DTMF tone B.
14	The DTMF tone C.
15	The DTMF tone D.
16	The DTMF tone FLASH.

Parameters:

<i>dtmfDuration</i>	The DTMF tone samples. Recommended value 160.
<i>playDtmfTone</i>	Set to true the SDK play local DTMF tone sound during send DTMF.

Returns:

If the function succeeds, it will return value 0. If the function fails, it will return a specific error code.

Refer functions

Functions

- int [com.portsip.PortSipSdk.refer](#) (long sessionId, String referTo)
- int [com.portsip.PortSipSdk.attendedRefer](#) (long sessionId, long replaceSessionId, String referTo)
- long [com.portsip.PortSipSdk.acceptRefer](#) (long referId, String referSignaling)
- int [com.portsip.PortSipSdk.rejectRefer](#) (long referId)

Detailed Description

Function Documentation

int com.portsip.PortSipSdk.refer (long sessionId, String referTo)

Transfer the current call to another callee.

Parameters:

<i>sessionId</i>	The session ID of the call.
<i>referTo</i>	Target callee of the transfer. It can be either "sip:number@sipserver.com" or "number".

Returns:

If the function succeeds, it will return value 0. If the function fails, it will return a specific error code.

Remarks:

```
refer(sessionId, "sip:testuser12@sip.portsip.com");
```

You can refer to the video on Youtube at:

<https://www.youtube.com/watch?v=2w9EGgr3FY>, which will demonstrate how to complete the transfer.

int com.portsip.PortSipSdk.attendedRefer (long *sessionId*, long *replaceSessionId*, String *referTo*)

Make an attended refer.

Parameters:

<i>sessionId</i>	The session ID of the call.
<i>replaceSessionId</i>	Session ID of the replace call.
<i>referTo</i>	Target callee of the refer. It can be either "sip:number@sipserver.com" or "number".

Returns:

If the function succeeds, it will return value 0. If the function fails, it will return a specific error code.

Remarks:

Please read the sample project source code to get more details, or you can refer to the video on YouTube at:

<https://www.youtube.com/watch?v=2w9EGgr3FY>

Note: Please use Windows Media Player to play the AVI file which demonstrates how to complete the transfer.

long com.portsip.PortSipSdk.acceptRefer (long *referId*, String *referSignaling*)

By accepting the REFER request, a new call will be made if this function is called. The function is usually called after onReceivedRefer callback event.

Parameters:

<i>referId</i>	The ID of REFER request that comes from onReceivedRefer callback event.
<i>referSignaling</i>	The SIP message of REFER request that comes from onReceivedRefer callback event.

Returns:

If the function succeeds, it will return a session ID greater than 0 to the new call for REFER; otherwise it will return a specific error code less than 0;

int com.portsip.PortSipSdk.rejectRefer (long *referId*)

Reject the REFER request.

Parameters:

<i>referId</i>	The ID of REFER request that comes from onReceivedRefer callback event.
----------------	---

Returns:

If the function succeeds, it will return value 0. If the function fails, it will return a specific error code.

Send audio and video stream functions

Functions

- int [com.portsip.PortSipSdk.enableSendPcmStreamToRemote](#) (long sessionId, boolean state, int streamSamplesPerSec)
- int [com.portsip.PortSipSdk.sendPcmStreamToRemote](#) (long sessionId, byte[] data, int dataLength)

- int [com.portsip.PortSipSdk.enableSendVideoStreamToRemote](#) (long sessionId, boolean state)
- int [com.portsip.PortSipSdk.sendVideoStreamToRemote](#) (long sessionId, byte[] data, int dataLength, int width, int height)

Detailed Description

Function Documentation

int com.portsip.PortSipSdk.enableSendPcmStreamToRemote (long sessionId, boolean state, int streamSamplesPerSec)

Enable the SDK send PCM stream data to remote side from another source instead of microphone. This function MUST be called first to send the PCM stream data to another side.

Parameters:

<i>sessionId</i>	The session ID of call.
<i>state</i>	Set to true to enable the send stream, or false to disable.
<i>streamSamplesPerSec</i>	The PCM stream data sample, in seconds. For example 8000 or 16000.

Returns:

If the function succeeds, it will return value 0. If the function fails, it will return a specific error code.

int com.portsip.PortSipSdk.sendPcmStreamToRemote (long sessionId, byte [] data, int dataLength)

Send the audio stream in PCM format from another source instead of audio device capturing (microphone).

Parameters:

<i>sessionId</i>	Session ID of the call conversation.
<i>data</i>	The PCM audio stream data. It must be 16bit, mono.
<i>dataLength</i>	The size of data.

Returns:

If the function succeeds, it will return value 0. If the function fails, it will return a specific error code.

Remarks:

Usually we should use it like below:

```
enableSendPcmStreamToRemote(sessionId, true, 16000);
sendPcmStreamToRemote(sessionId, data, dataSize);
```

int com.portsip.PortSipSdk.enableSendVideoStreamToRemote (long sessionId, boolean state)

Enable the SDK to send video stream data to remote side from another source instead of camera.

This function MUST be called first to send the video stream data to another side.

Parameters:

<i>sessionId</i>	The session ID of call.
<i>state</i>	Set to true to enable the send stream, or false to disable.

Returns:

If the function succeeds, it will return value 0. If the function fails, it will return a specific error code.

int com.portsip.PortSipSdk.sendVideoStreamToRemote (long sessionId, byte [] data, int dataLength, int width, int height)

Send the video stream in i420 from another source instead of video device capturing (camera).

Before calling this function, you MUST call the enableSendVideoStreamToRemote function.

Parameters:

<i>sessionId</i>	Session ID of the call conversation.
<i>data</i>	The video stream data. It must be in i420 format.
<i>dataLength</i>	The size of data.
<i>width</i>	The width of the video image.
<i>height</i>	The height of video image.

Returns:

If the function succeeds, it will return value is 0. If the function fails, it will return a specific error code.

RTP packets, Audio stream and video stream callback

Functions

- void [com.portsip.PortSipSdk.setRtpCallback](#) (boolean enable)
- void [com.portsip.PortSipSdk.enableAudioStreamCallback](#) (long sessionId, boolean enable, int enum_audioCallbackMode)
- void [com.portsip.PortSipSdk.enableVideoStreamCallback](#) (long sessionId, int enum_videoCallbackMode)
- void [com.portsip.PortSipSdk.enableVideoDecoderCallback](#) (boolean enable)

Detailed Description

functions

Function Documentation

void com.portsip.PortSipSdk.setRtpCallback (boolean enable)

Set the RTP callbacks to allow access to the sent and received RTP packets.

Parameters:

<i>enable</i>	Set to true to enable the RTP callback for receiving and sending RTP packets. The onSendingRtpPacket and onReceivedRtpPacket events will be triggered.
---------------	--

void com.portsip.PortSipSdk.enableAudioStreamCallback (long *sessionId*, boolean *enable*, int *enum_audioCallbackMode*)

Enable/disable the audio stream callback. The onAudioRawCallback event will be triggered if the callback is enabled.

Parameters:

<i>sessionId</i>	The session ID of call.
<i>enable</i>	Set to true to enable audio stream callback, or false to stop the callback.
<i>enum_audioCallbackMode</i>	The audio stream callback mode. Supported modes include: ENUM_AUDIOSTREAM_NONE , ENUM_AUDIOSTREAM_LOCAL_MIX , ENUM_AUDIOSTREAM_LOCAL_PER_CHANNEL , ENUM_AUDIOSTREAM_REMOTE_MIX , ENUM_AUDIOSTREAM_REMOTE_PER_CHANNEL .

void com.portsip.PortSipSdk.enableVideoStreamCallback (long *sessionId*, int *enum_videoCallbackMode*)

Enable/disable the video stream callback, the onVideoRawCallback event will be triggered if the callback is enabled.

Parameters:

<i>sessionId</i>	The session ID of call.
<i>enum_videoCallbackMode</i>	The video stream callback mode. Supported modes include: ENUM_VIDEOSTREAM_NONE , ENUM_VIDEOSTREAM_LOCAL , ENUM_VIDEOSTREAM_REMOTE , ENUM_VIDEOSTREAM_BOTH .

void com.portsip.PortSipSdk.enableVideoDecoderCallback (boolean *enable*)

Enable/disable the video Decoder Info callback. The onVideoDecodedInfoCallback event will be triggered if the callback is enabled.

Parameters:

<i>sessionId</i>	The session ID of call.
<i>enable</i>	Set to true to enable video Decoder Info callback, or false to stop the callback.

Record functions

Functions

- int [com.portsip.PortSipSdk.startRecord](#) (long sessionId, String recordFilePath, String recordFileName, boolean appendTimeStamp, int enum_audioFileFormat, int enum_audioRecordMode, int enum_videocodec, int enum_videoRecordMode)
- int [com.portsip.PortSipSdk.stopRecord](#) (long sessionId)

Detailed Description

Function Documentation

int com.portsip.PortSipSdk.startRecord (long *sessionId*, String *recordFilePath*, String *recordFileName*, boolean *appendTimeStamp*, int *enum_audioFileFormat*, int *enum_audioRecordMode*, int *enum_videocodec*, int *enum_videoRecordMode*)

Start recording the call.

Parameters:

<i>sessionId</i>	The session ID of call conversation.
<i>recordFilePath</i>	The file path to save record file. It must be existent.
<i>recordFileName</i>	The file name of record file. For example audiorecord.wav or videorecord.avi.
<i>appendTimeStamp</i>	Set to true to append the timestamp to the name of the recording file.
<i>enum_audioFileFormat</i>	The audio record file format, allow below values: FILEFORMAT_WAVE = 1, /// The record audio file is WAVE format. FILEFORMAT_AMR, /// The record audio file is in AMR format with all voice data compressed by AMR codec.
<i>enum_audioRecordMode</i>	The audio record mode, allow below values: RECORD_NONE = 0, /// Not Record. RECORD_RECV = 1, /// Only record the received data. RECORD_SEND, /// Only record send data. RECORD_BOTH /// The record audio file is WAVE format.
<i>enum_videocodec</i>	The codec which is used for compressing the video data to save into video record file.
<i>enum_videoRecordMode</i>	Allow to set video record mode. Support to record received and/or sent video.

Returns:

If the function succeeds, it will return value 0. If the function fails, it will return a specific error code.

int com.portsip.PortSipSdk.stopRecord (long *sessionId*)

Stop recording.

Parameters:

<i>sessionId</i>	The session ID of call conversation.
------------------	--------------------------------------

Returns:

If the function succeeds, it will return value 0. If the function fails, it will return a specific error code.

Play audio and video file to remote functions

Functions

- int [com.portsip.PortSipSdk.playVideoFileToRemote](#) (long *sessionId*, String *aviFile*, boolean *loop*, boolean *playAudio*)
- int [com.portsip.PortSipSdk.stopPlayVideoFileToRemote](#) (long *sessionId*)
- int [com.portsip.PortSipSdk.playAudioFileToRemote](#) (long *sessionId*, String *filename*, int *fileSamplesPerSec*, boolean *loop*)
- int [com.portsip.PortSipSdk.stopPlayAudioFileToRemote](#) (long *sessionId*)
- int [com.portsip.PortSipSdk.playAudioFileToRemoteAsBackground](#) (long *sessionId*, String *filename*, int *fileSamplesPerSec*)
- int [com.portsip.PortSipSdk.stopPlayAudioFileToRemoteAsBackground](#) (long *sessionId*)

Detailed Description

Function Documentation

int com.portsip.PortSipSdk.playVideoFileToRemote (long *sessionId*, String *aviFile*, boolean *loop*, boolean *playAudio*)

Play an AVI file to remote party.

Parameters:

<i>sessionId</i>	Session ID of the call.
<i>aviFile</i>	The full filepath, such as "/mnt/sdcard/test.avi".
<i>loop</i>	Set to false to stop playing video file when it is ended, or true to play it repeatedly.
<i>playAudio</i>	If set to true, audio and video will be played together; or false to play the video only.

Returns:

If the function succeeds, it will return value 0. If the function fails, it will return a specific error code.

int com.portsip.PortSipSdk.stopPlayVideoFileToRemote (long *sessionId*)

Stop play video file to remote side.

Parameters:

<i>sessionId</i>	Session ID of the call.
------------------	-------------------------

Returns:

If the function succeeds, it will return value 0. If the function fails, it will return a specific error code.

int com.portsip.PortSipSdk.playAudioFileToRemote (long *sessionId*, String *filename*, int *fileSamplesPerSec*, boolean *loop*)

Play a wave file to remote party.

Parameters:

<i>sessionId</i>	Session ID of the call.
<i>filename</i>	The full filepath, such as "/mnt/sdcard/test.wav".
<i>fileSamplesPerSec</i>	The wave file sample in seconds. It could be 8000, 16000 or 32000.
<i>loop</i>	Set to false to stop playing audio file when it is ended, or true to play it repeatedly.

Returns:

If the function succeeds, it will return value 0. If the function fails, it will return a specific error code.

int com.portsip.PortSipSdk.stopPlayAudioFileToRemote (long *sessionId*)

Stop playing wave file to remote side.

Parameters:

<i>sessionId</i>	Session ID of the call.
------------------	-------------------------

Returns:

If the function succeeds, it will return value 0. If the function fails, it will return a specific error code.

int com.portsip.PortSipSdk.playAudioFileToRemoteAsBackground (long *sessionId*, String *filename*, int *fileSamplesPerSec*)

Play a wave file to remote party as conversation background sound.

Parameters:

<i>sessionId</i>	Session ID of the call.
<i>filename</i>	The full filepath, such as "/mnt/sdcard/test.wav".
<i>fileSamplesPerSec</i>	The wave file sample, in seconds. It should be 8000, 16000 or 32000.

Returns:

If the function succeeds, it will return value 0. If the function fails, it will return a specific error code.

int com.portsip.PortSipSdk.stopPlayAudioFileToRemoteAsBackground (long *sessionId*)

Stop playing a wave file to remote party as background sound for the conversation.

Parameters:

<i>sessionId</i>	Session ID of the call.
------------------	-------------------------

Returns:

If the function succeeds, it will return value 0. If the function fails, it will return a specific error code.

Conference functions

Functions

- int [com.portsip.PortSipSdk.createConference](#) (Object *conferenceVideoWindow*, int *videoWidth*, int *videoHeight*, boolean *displayLocalVideoInConference*)
- void [com.portsip.PortSipSdk.destroyConference](#) ()
- int [com.portsip.PortSipSdk.setConferenceVideoWindow](#) (Object *conferenceVideoWindow*)
- int [com.portsip.PortSipSdk.joinToConference](#) (long *sessionId*)
- int [com.portsip.PortSipSdk.removeFromConference](#) (long *sessionId*)

Detailed Description

Function Documentation

int com.portsip.PortSipSdk.createConference (Object *conferenceVideoWindow*, int *videoWidth*, int *videoHeight*, boolean *displayLocalVideoInConference*)

Create a conference. It will be failed if the existing conference is not ended yet.

Parameters:

<i>conferenceVideoWindow</i>	SurfaceView The window used for displaying the conference video.
<i>videoWidth</i>	Width of conference video resolution
<i>videoHeight</i>	Height of conference video resolution
<i>displayLocalVideoInConference</i>	Display local video during conference

Returns:

If the function succeeds, it will return value 0. If the function fails, it will return a specific error code.

void com.portsip.PortSipSdk.destroyConference ()

End the exist conference.

int com.portsip.PortSipSdk.setConferenceVideoWindow (Object conferenceVideoWindow)

Set the window for a conference that is used for displaying the received remote video image.

Parameters:

<i>conferenceVideoWindow</i>	SurfaceView The window which is used for displaying the conference video
------------------------------	--

Returns:

If the function succeeds, it will return value 0. If the function fails, it will return a specific error code.

int com.portsip.PortSipSdk.joinToConference (long sessionId)

Join a session into existing conference. If the call is in hold, it will be un-hold automatically.

Parameters:

<i>sessionId</i>	Session ID of the call.
------------------	-------------------------

Returns:

If the function succeeds, it will return value 0. If the function fails, it will return a specific error code.

int com.portsip.PortSipSdk.removeFromConference (long sessionId)

Remove a session from an existing conference.

Parameters:

<i>sessionId</i>	Session ID of the call.
------------------	-------------------------

Returns:

If the function succeeds, it will return value 0. If the function fails, it will return a specific error code.

RTP and RTCP QOS functions

Functions

- int [com.portsip.PortSipSdk.setAudioRtcpBandwidth](#) (long sessionId, int BitsRR, int BitsRS, int KBitsAS)

- int [com.portsip.PortSipSdk.setVideoRtcpBandwidth](#) (long sessionId, int BitsRR, int BitsRS, int KBitsAS)
- int [com.portsip.PortSipSdk.setAudioQos](#) (boolean enable, int DSCPValue, int priority)
- int [com.portsip.PortSipSdk.setVideoQos](#) (boolean enable, int DSCPValue)
- int [com.portsip.PortSipSdk.setVideoMTU](#) (int mtu)

Detailed Description

Function Documentation

int com.portsip.PortSipSdk.setAudioRtcpBandwidth (long sessionId, int BitsRR, int BitsRS, int KBitsAS)

Set the audio RTCP bandwidth parameters as RFC3556.

Parameters:

<i>sessionId</i>	Set the audio RTCP bandwidth parameters as RFC3556.
<i>BitsRR</i>	The bits for the RR parameter.
<i>BitsRS</i>	The bits for the RS parameter.
<i>KBitsAS</i>	The Kbits for the AS parameter.

Returns:

If the function succeeds, it will return value 0. If the function fails, it will return a specific error code.

int com.portsip.PortSipSdk.setVideoRtcpBandwidth (long sessionId, int BitsRR, int BitsRS, int KBitsAS)

Set the video RTCP bandwidth parameters as the RFC3556.

Parameters:

<i>sessionId</i>	The session ID of call conversation.
<i>BitsRR</i>	The bits for the RR parameter.
<i>BitsRS</i>	The bits for the RS parameter.
<i>KBitsAS</i>	The Kbits for the AS parameter.

Returns:

If the function succeeds, it will return value 0. If the function fails, it will return a specific error code.

int com.portsip.PortSipSdk.setAudioQos (boolean enable, int DSCPValue, int priority)

Set the DSCP (differentiated services code point) value of QoS (Quality of Service) for audio channel.

Parameters:

<i>enable</i>	Set to true to enable audio QoS.
<i>DSCPValue</i>	The six-bit DSCP value. Valid range is 0-63. As defined in RFC 2472, the DSCP value is the high-order 6 bits of the IP version 4 (IPv4) TOS field and the IP version 6 (IPv6) Traffic Class field.
<i>priority</i>	The 802.1p priority(PCP) field in a 802.1Q/VLAN tag. Values 0-7 indicates the priority, while value -1 leaves the priority setting unchanged.

Returns:

If the function succeeds, it will return value 0. If the function fails, it will return a specific error code.

int com.portsip.PortSipSdk.setVideoQos (boolean *enable*, int *DSCPValue*)

Set the DSCP(differentiated services code point) value of QoS(Quality of Service) for video channel.

Parameters:

<i>enable</i>	Set as true to enable QoS, or false to disable.
<i>DSCPValue</i>	The six-bit DSCP value. Valid range is 0-63. As defined in RFC 2472, the DSCP value is the high-order 6 bits of the IP version 4 (IPv4) TOS field and the IP version 6 (IPv6) Traffic Class field.

Returns:

If the function succeeds, it will return value 0. If the function fails, it will return a specific error code.

int com.portsip.PortSipSdk.setVideoMTU (int *mtu*)

Set the MTU size for video RTP packet.

Parameters:

<i>mtu</i>	Set MTU value. Allow values range 512 - 65507. Default is 14000.
------------	--

Returns:

If the function succeeds, the return value is 0. If the function fails, the return value is a specific error code.

RTP statistics functions

Functions

- int [com.portsip.PortSipSdk.getNetworkStatistics](#) (long sessionId, int[] statistics)
- int [com.portsip.PortSipSdk.getAudioRtpStatistics](#) (long sessionId, int[] statistics)
- int [com.portsip.PortSipSdk.getAudioRtcpStatistics](#) (long sessionId, int[] statistics)
- int [com.portsip.PortSipSdk.getVideoRtpStatistics](#) (long sessionId, int[] statistics)

Detailed Description

Function Documentation

int com.portsip.PortSipSdk.getNetworkStatistics (long *sessionId*, int [] *statistics*)

Get the "in-call" statistics. The statistics are reset after the query.

Parameters:

<i>sessionId</i>	The session ID of call conversation.
<i>statistics</i>	Return network statistic statistics[0] - Current jitter buffer size in ms.

	statistics[1] - Preferred buffer size in ms. statistics[2] - Loss rate (network + late) in percent. statistics[3] - Late loss rate in percent. statistics[4] - Fraction (of original stream) of synthesized speech inserted through expansion. statistics[5] - Fraction of synthesized speech inserted through pre-emptive expansion. statistics[6] - fraction of data removed through acceleration through acceleration.
--	--

Returns:

If the function succeeds, it will return value 0. If the function fails, it will return a specific error code.

int com.portsip.PortSipSdk.getAudioRtpStatistics (long *sessionId*, int [] *statistics*)

Obtain the RTP statistics of audio channel.

Parameters:

<i>sessionId</i>	The session ID of call conversation.
<i>statistics</i>	Return audio RTP statistic statistics[0] - Short-time average jitter (in milliseconds). statistics[1] - Maximum short-time jitter (in milliseconds). statistics[2] - The number of discarded packets on a channel during the call.

Returns:

If the function succeeds, it will return value 0. If the function fails, it will return a specific error code.

int com.portsip.PortSipSdk.getAudioRtcpStatistics (long *sessionId*, int [] *statistics*)

Obtain the RTCP statistics of audio channel.

Parameters:

<i>sessionId</i>	The session ID of call conversation.
<i>statistics</i>	Return audio RTCP statistics statistics[0] - The number of sent bytes. statistics[1] - The number of sent packets. statistics[2] - The number of received bytes. statistics[3] - The number of received packets. statistics[4] - Fraction of sent loss in percentage. statistics[5] - The number of sent cumulative lost packet. statistics[6] - Fraction of received loss in percent. statistics[7] - The number of received cumulative lost packets.

Returns:

If the function succeeds, it will return value is 0. If the function fails, it will return a specific error code.

int com.portsip.PortSipSdk.getVideoRtpStatistics (long *sessionId*, int [] *statistics*)

Obtain the RTCP statistics of audio channel.

Parameters:

<i>sessionId</i>	The session ID of call conversation.
<i>statistics</i>	Return Video RTCP statistic statistics[0] - The number of sent bytes. statistics[1] - The number of sent packets. statistics[2] - The number of received bytes.

	statistics[3] - The number of received packets.
--	---

Returns:

If the function succeeds, it will return value 0. If the function fails, it will return a specific error code.

Audio effect functions

Functions

- void [com.portsip.PortSipSdk.enableVAD](#) (boolean state)
- void [com.portsip.PortSipSdk.enableAEC](#) (int enum_aecMode)
- void [com.portsip.PortSipSdk.enableCNG](#) (boolean state)
- void [com.portsip.PortSipSdk.enableAGC](#) (int enum_agcMode)
- void [com.portsip.PortSipSdk.enableANS](#) (int enum_nsMode)

Detailed Description

Function Documentation

void com.portsip.PortSipSdk.enableVAD (boolean state)

Enable/disable Voice Activity Detection(VAD).

Parameters:

<i>state</i>	Set to true to enable VAD, or false to disable.
--------------	---

void com.portsip.PortSipSdk.enableAEC (int enum_aecMode)

Enable/disable AEC (Acoustic Echo Cancellation).

Parameters:

<i>enum_aecMode</i>	
---------------------	--

Mode	Description
EC_NONE	0 - Disable AEC.
EC_DEFAULT	1 - Platform with default AEC.
EC_CONFERENCE	2 - Desktop platform (windows,MAC), default for Conferencing (aggressive AEC).
EC_AEC	3 - Desktop platform (windows,MAC), Acoustic Echo Cancellation (default desktop Platform).
EC_AECM_1	4 - Mobile platform (iOS,Android) with use of most earpieces.
EC_AECM_2	5 - Mobile platform(iOS,Android) with use of loud earpiece or quiet speakerphone.
EC_AECM_3	6 - Mobile platform(iOS,Android) most speakerphone use (Mobile Platform default).
EC_AECM_4	7 - Mobile platform (iOS,Android) with loud

	speakerphone.
--	---------------

void com.portsip.PortSipSdk.enableCNG (boolean *state*)

Enable/disable Comfort Noise Generator(CNG).

Parameters:

<i>state</i>	Set to true to enable CNG, or false to disable.
--------------	---

void com.portsip.PortSipSdk.enableAGC (int *enum_agcMode*)

Enable/disable Automatic Gain Control(AGC).

Parameters:

<i>enum_agcMode</i>	
---------------------	--

Mode	Description
AGC_NONE	0 - Disable AGC.
AGC_DEFAULT	1 - Platform by default.
AGC_ADAPTIVE_ANALOG	2 - Desktop platform (windows,MAC), adaptive mode for use when analog volume control exists.
AGC_ADAPTIVE_DIGITAL	3 - Scaling takes place in the digital domain (e.g. for conference servers and embedded devices).
AGC_FIXED_DIGITAL	4 - Can be used on embedded devices where the capture signal level is predictable.

void com.portsip.PortSipSdk.enableANS (int *enum_nsMode*)

Enable/disable Audio Noise Suppression(ANS).

Parameters:

<i>enum_nsMode</i>	
--------------------	--

Mode	Description
NS_NONE	0 - Disable NS.
NS_DEFAULT	1 - Platform in default.
NS_Conference	2 - Conferencing in default.
NS_LOW_SUPPRESSION	3 - Lowest suppression.
NS_MODERATE_SUPPRESSION	4 - Moderate suppression.
NS_HIGH_SUPPRESSION	5 - High suppression.
NS_VERY_HIGH_SUPPRESSION	6 - Highest suppression.

Send OPTIONS/INFO/MESSAGE functions

Functions

- int [com.portsip.PortSipSdk.sendOptions](#) (String to, String sdp)
- int [com.portsip.PortSipSdk.sendInfo](#) (long sessionId, String mimeType, String subMimeType, String infoContents)

- long [com.portsip.PortSipSdk.sendMessage](#) (long sessionId, String mimeType, String subMimeType, byte[] message, int messageLength)
- long [com.portsip.PortSipSdk.sendOutOfDialogMessage](#) (String to, String mimeType, String subMimeType, byte[] message, int messageLength)
- long [com.portsip.PortSipSdk.presenceSubscribeContact](#) (String contact, String subject)
- int [com.portsip.PortSipSdk.presenceRejectSubscribe](#) (long subscribeId)
- int [com.portsip.PortSipSdk.presenceAcceptSubscribe](#) (long subscribeId)
- int [com.portsip.PortSipSdk.presenceOnline](#) (long subscribeId, String statusText)
- int [com.portsip.PortSipSdk.presenceOffline](#) (long subscribeId)

Detailed Description

Function Documentation

int com.portsip.PortSipSdk.sendOptions (String to, String sdp)

Send OPTIONS message.

Parameters:

<i>to</i>	The recipient of OPTIONS message.
<i>sdp</i>	The SDP of OPTIONS message. It's optional if user does not want to send the SDP with OPTIONS message.

Returns:

If the function succeeds, it will return value 0. If the function fails, it will return value a specific error code.

int com.portsip.PortSipSdk.sendInfo (long sessionId, String mimeType, String subMimeType, String infoContents)

Send a INFO message to remote side in dialog.

Parameters:

<i>sessionId</i>	The session ID of call.
<i>mimeType</i>	The mime type of INFO message.
<i>subMimeType</i>	The sub mime type of INFO message.
<i>infoContents</i>	The contents that will be sent with INFO message.

Returns:

If the function succeeds, it will return value 0. If the function fails, it will return a specific error code.

long com.portsip.PortSipSdk.sendMessage (long sessionId, String mimeType, String subMimeType, byte [] message, int messageLength)

Send a MESSAGE message to remote side in dialog.

Parameters:

<i>sessionId</i>	The session ID of call.
<i>mimeType</i>	The mime type of MESSAGE message.
<i>subMimeType</i>	The sub mime type of MESSAGE message.
<i>message</i>	The contents that will be sent with MESSAGE message. Binary data allowed.
<i>messageLength</i>	The message size.

Returns:

If the function succeeds, it will return a message ID that allows to track the message sending state in `onSendMessageSuccess` and `onSendMessageFailure`. If the function fails, it will return a specific error code that is less than 0.

Remarks:

Example 1: Send a plain text message. Note: to send other languages text, please use the UTF8 to encode the message before sending.

```
sendMessage(sessionId, "text", "plain", "hello", 6);
```

Example 2: Send a binary message.

```
sendMessage(sessionId, "application", "vnd.3gpp.sms", binData, binDataSize);
```

long com.portsip.PortSipSdk.sendOutOfDialogMessage (String to, String mimeType, String subMimeType, byte [] message, int messageLength)

Send a out of dialog MESSAGE message to remote side.

Parameters:

<i>to</i>	The message receiver. Likes sip: receiver@portsip.com
<i>mimeType</i>	The mime type of MESSAGE message.
<i>subMimeType</i>	The sub mime type of MESSAGE message.
<i>message</i>	The contents that will be sent with MESSAGE message. Binary data allowed.
<i>messageLength</i>	The message size.

Returns:

If the function succeeds, it will return a message ID that allows to track the message sending state in `onSendOutOfMessageSuccess` and `onSendOutOfMessageFailure`. If the function fails, it will return a specific error code that is less than 0.

Remarks:

Example 1: Send a plain text message. Note: to send other languages text, please use the UTF8 to encode the message before sending.

```
sendOutOfDialogMessage("sip:user1@sip.portsip.com", "text", "plain", "hello", 6);
```

Example 2: Send a binary message.

```
sendOutOfDialogMessage("sip:user1@sip.portsip.com", "application", "vnd.3gpp.sms", binData, binDataSize);
```

long com.portsip.PortSipSdk.presenceSubscribeContact (String contact, String subject)

Send a SUBSCRIBE message for presence to a contact.

Parameters:

<i>contact</i>	The target contact, it must be in the format of sip: contact001@sip.portsip.com .
<i>subject</i>	This subject text will be inserted into the SUBSCRIBE message. For example: "Hello, I'm Jason". The subject maybe is in UTF8 format. You should use UTF8 to decode it.

Returns:

If the function succeeds, it will return value 0. If the function fails, it will return a specific error code.

int com.portsip.PortSipSdk.presenceRejectSubscribe (long subscribelId)

Reject a presence SUBSCRIBE request received from contact.

Parameters:

<i>subscribelId</i>	Subscription ID. When receiving a SUBSCRIBE request from contact, the
---------------------	---

	event onPresenceRecvSubscribe will be triggered. The event includes the subscription ID.
--	--

Returns:

If the function succeeds, it will return value 0. If the function fails, it will return a specific error code.

int com.portsip.PortSipSdk.presenceAcceptSubscribe (long *subscribeId*)

Accept the presence SUBSCRIBE request which received from contact.

Parameters:

<i>subscribeId</i>	Subscription ID. When receiving a SUBSCRIBE request from contact, the event onPresenceRecvSubscribe will be triggered. The event includes the subscription ID.
--------------------	--

Returns:

If the function succeeds, it will return value 0. If the function fails, it will return a specific error code.

int com.portsip.PortSipSdk.presenceOnline (long *subscribeId*, String *statusText*)

Send a NOTIFY message to contact to notify that the presence status is online/changed.

Parameters:

<i>subscribeId</i>	Subscription ID. When receiving a SUBSCRIBE request from contact, the event onPresenceRecvSubscribe will be triggered. The event includes the subscription ID.
<i>statusText</i>	The state text of presende online, for example "I'm here"

Returns:

If the function succeeds, it will return value 0. If the function fails, it will return a specific error code.

int com.portsip.PortSipSdk.presenceOffline (long *subscribeId*)

Send a NOTIFY message to contact to notify that the presence status is offline.

Parameters:

<i>subscribeId</i>	Subscription ID. When receiving a SUBSCRIBE request from contact, the event onPresenceRecvSubscribe will be triggered. The event includes the subscription ID.
--------------------	--

Returns:

If the function succeeds, it will return value 0. If the function fails, it will return a specific error code.

Device Management functions.

Functions

- int [com.portsip.PortSipSdk.getNumOfRecordingDevices](#) ()
- int [com.portsip.PortSipSdk.getNumOfPlayoutDevices](#) ()
- String [com.portsip.PortSipSdk.getRecordingDeviceName](#) (int index)
- String [com.portsip.PortSipSdk.getPlayoutDeviceName](#) (int index)
- int [com.portsip.PortSipSdk.setSpeakerVolume](#) (int volume)
- int [com.portsip.PortSipSdk.getSpeakerVolume](#) ()

- int [com.portsip.PortSipSdk.setSystemOutputMute](#) (boolean mute)
- boolean [com.portsip.PortSipSdk.getSystemOutputMute](#) ()
- int [com.portsip.PortSipSdk.setMicVolume](#) (int volume)
- int [com.portsip.PortSipSdk.getMicVolume](#) ()
- int [com.portsip.PortSipSdk.setSystemInputMute](#) (boolean mute)
- boolean [com.portsip.PortSipSdk.getSystemInputMute](#) ()
- void [com.portsip.PortSipSdk.audioPlayLoopbackTest](#) (boolean enable)
- int [com.portsip.PortSipSdk.getNumOfVideoCaptureDevices](#) ()
- String [com.portsip.PortSipSdk.getVideoCaptureDeviceName](#) (int index)

Detailed Description

Function Documentation

int com.portsip.PortSipSdk.getNumOfRecordingDevices ()

Gets the number of audio devices available for audio recording.

Returns:

If the function succeeds, it will return the number of recording devices. If the function fails, it will return a specific error code that is less than 0.

int com.portsip.PortSipSdk.getNumOfPlayoutDevices ()

Gets the number of audio devices available for audio playout.

Returns:

If the function succeeds, it will return value 0. If the function fails, it will return a specific error code.

String com.portsip.PortSipSdk.getRecordingDeviceName (int *index*)

Gets the name of a specific recording device given by an index.

Parameters:

<i>index</i>	Device index (0, 1, 2, ..., N-1), of which N is given by getNumOfRecordingDevices (). Also -1 is a valid value and it will return the name of the default recording device.
--------------	---

Returns:

String The name of a specific recording device given by an index.

String com.portsip.PortSipSdk.getPlayoutDeviceName (int *index*)

Gets the name of a specific playout device given by an index.

Parameters:

<i>index</i>	Device index (0, 1, 2, ..., N-1), of which N is given by getNumOfPlayoutDevices (). Also -1 is a valid value and it will return the name of the default playout device.
--------------	---

Returns:

String The name of a specific playout device given by an index.

int com.portsip.PortSipSdk.setSpeakerVolume (int *volume*)

Set the speaker volume.

Parameters:

<i>volume</i>	Volume level of speaker. Valid value ranges 0 to 255.
---------------	---

Returns:

If the function succeeds, it will return value 0. If the function fails, it will return a specific error code.

int com.portsip.PortSipSdk.getSpeakerVolume ()

Gets the speaker volume level.

Returns:

If the function succeeds, it will return the value of speaker volume. If the function fails, it will return a specific error code.

int com.portsip.PortSipSdk.setSystemOutputMute (boolean *mute*)

Mute the speaker device completely in the operating system.

Parameters:

<i>mute</i>	If set to true, the device output is muted. If set to false, the output is unmuted.
-------------	---

Returns:

If the function succeeds, it will return value 0. If the function fails, it will return a specific error code.

boolean com.portsip.PortSipSdk.getSystemOutputMute ()

Retrieves the output device mute state in the operating system.

Returns:

If return value is true, the output device is muted. If false, the output device is not muted.

int com.portsip.PortSipSdk.setMicVolume (int *volume*)

Sets the microphone volume.

Parameters:

<i>volume</i>	The microphone volume. The valid value ranges from 0 to 255.
---------------	--

Returns:

If the function succeeds, it will return value 0. If the function fails, it will return a specific error code.

int com.portsip.PortSipSdk.getMicVolume ()

Retrieves the current microphone volume.

Returns:

If the function succeeds, it will return the value of microphone volume. If the function fails, it will return a specific error code.

int com.portsip.PortSipSdk.setSystemInputMute (boolean *mute*)

Mute the microphone input device completely in the operating system.

Parameters:

<i>mute</i>	If set to true, the input device is muted, or false to unmute it.
-------------	---

Returns:

If the function succeeds, it will return value 0. If the function fails, it will return a specific error code.

boolean com.portsip.PortSipSdk.getSystemInputMute ()

Gets the mute state of the input device in the operating system.

Returns:

If it returns true, the input device is muted; If false, the input device is not muted.

void com.portsip.PortSipSdk.audioPlayLoopbackTest (boolean *enable*)

Used for testing loopback for the audio device.

Parameters:

<i>enable</i>	Set to true to start testing audio loopback test; or set to false to stop.
---------------	--

int com.portsip.PortSipSdk.getNumOfVideoCaptureDevices ()

Gets the number of available capture devices.

Returns:

If the function succeeds, it will return the number of video capturing devices. Or if the function fails, it will return a specific error code that is less than 0.

String com.portsip.PortSipSdk.getVideoCaptureDeviceName (int *index*)

Gets the name of a specific video capture device given by an index.

Parameters:

<i>index</i>	Device index (0, 1, 2, ..., N-1), of which N is given by getNumOfVideoCaptureDevices (). Also -1 is a valid value and will return the name of the default capturing device.
--------------	---

Returns:

The name of a specific video capturing device given by an index.

Class Documentation

com.portsip.AndroidGLES20 Class Reference

Inherits GLSurfaceView, and Renderer.

Public Member Functions

- **AndroidGLES20** (Context context)
- **AndroidGLES20** (Context context, boolean translucent, int depth, int stencil)
- void **onDrawFrame** (GL10 gl)
- void **onSurfaceChanged** (GL10 gl, int width, int height)
- void **onSurfaceCreated** (GL10 gl, EGLConfig config)
- void **RegisterNativeObject** (long nativeObject)
- void **DeRegisterNativeObject** ()
- void **ReDraw** ()

Static Public Member Functions

- static boolean **UseOpenGL2** (Object renderWindow)
- static boolean **IsSupported** (Context context)

The documentation for this class was generated from the following file:

- AndroidGLES20.java

com.portsip.OnPortSIPEvent Interface Reference

Public Member Functions

- void [onRegisterSuccess](#) (String reason, int code)
- void [onRegisterFailure](#) (String reason, int code)
- void [onInviteIncoming](#) (long sessionId, String callerDisplayName, String caller, String calleeDisplayName, String callee, String audioCodecs, String videoCodecs, boolean existsAudio, boolean existsVideo)
- void [onInviteTrying](#) (long sessionId)
- void [onInviteSessionProgress](#) (long sessionId, String audioCodecs, String videoCodecs, boolean existsEarlyMedia, boolean existsAudio, boolean existsVideo)
- void [onInviteRinging](#) (long sessionId, String statusText, int statusCode)
- void [onInviteAnswered](#) (long sessionId, String callerDisplayName, String caller, String calleeDisplayName, String callee, String audioCodecs, String videoCodecs, boolean existsAudio, boolean existsVideo)
- void [onInviteFailure](#) (long sessionId, String reason, int code)
- void [onInviteUpdated](#) (long sessionId, String audioCodecs, String videoCodecs, boolean existsAudio, boolean existsVideo)
- void [onInviteConnected](#) (long sessionId)
- void [onInviteBeginingForward](#) (String forwardTo)
- void [onInviteClosed](#) (long sessionId)
- void [onRemoteHold](#) (long sessionId)
- void [onRemoteUnHold](#) (long sessionId, String audioCodecs, String videoCodecs, boolean existsAudio, boolean existsVideo)
- void [onReceivedRefer](#) (long sessionId, long referId, String to, String from, String referSipMessage)
- void [onReferAccepted](#) (long sessionId)
- void [onReferRejected](#) (long sessionId, String reason, int code)
- void [onTransferTrying](#) (long sessionId)
- void [onTransferRinging](#) (long sessionId)
- void [onACTVTransferSuccess](#) (long sessionId)
- void [onACTVTransferFailure](#) (long sessionId, String reason, int code)
- void [onReceivedSignaling](#) (long sessionId, String message)
- void [onSendingSignaling](#) (long sessionId, String message)
- void [onWaitingVoiceMessage](#) (String messageAccount, int urgentNewMessageCount, int urgentOldMessageCount, int newMessageCount, int oldMessageCount)
- void [onWaitingFaxMessage](#) (String messageAccount, int urgentNewMessageCount, int urgentOldMessageCount, int newMessageCount, int oldMessageCount)
- void [onRecvDtmfTone](#) (long sessionId, int tone)
- void [onRecvOptions](#) (String optionsMessage)
- void [onRecvInfo](#) (String infoMessage)
- void [onPresenceRecvSubscribe](#) (long subscribeId, String fromDisplayName, String from, String subject)
- void [onPresenceOnline](#) (String fromDisplayName, String from, String stateText)
- void [onPresenceOffline](#) (String fromDisplayName, String from)
- void [onRecvMessage](#) (long sessionId, String mimeType, String subMimeType, byte[] messageData, int messageDataLength)
- void [onRecvOutOfDialogMessage](#) (String fromDisplayName, String from, String toDisplayName, String to, String mimeType, String subMimeType, byte[] messageData, int messageDataLength)
- void [onSendMessageSuccess](#) (long sessionId, long messageId)
- void [onSendMessageFailure](#) (long sessionId, long messageId, String reason, int code)
- void [onSendOutOfDialogMessageSuccess](#) (long messageId, String fromDisplayName, String from, String toDisplayName, String to)
- void [onSendOutOfDialogMessageFailure](#) (long messageId, String fromDisplayName, String from, String toDisplayName, String to, String reason, int code)

- void [onPlayAudioFileFinished](#) (long sessionId, String fileName)
- void [onPlayVideoFileFinished](#) (long sessionId)
- void [onReceivedRTPPacket](#) (long sessionId, boolean isAudio, byte[] RTPPacket, int packetSize)
- void [onSendingRTPPacket](#) (long sessionId, boolean isAudio, byte[] RTPPacket, int packetSize)
- void [onAudioRawCallback](#) (long sessionId, int enum_audioCallbackMode, byte[] data, int dataLength, int samplingFreqHz)
- void [onVideoRawCallback](#) (long sessionId, int enum_videoCallbackMode, int width, int height, byte[] data, int dataLength)
- void [onVideoDecodedInfoCallback](#) (long sessionId, int width, int height, int framerate, int bitrate)

The documentation for this interface was generated from the following file:

- OnPortSIPEvent.java

com.portsip.PortSipEnumDefine Class Reference

Static Public Attributes

- static final int `ENUM_AUDIOCODEC_G729` = 18
- static final int `ENUM_AUDIOCODEC_PCMA` = 8
- static final int `ENUM_AUDIOCODEC_PCMU` = 0
- static final int `ENUM_AUDIOCODEC_GSM` = 3
- static final int `ENUM_AUDIOCODEC_G722` = 9
- static final int `ENUM_AUDIOCODEC_ILBC` = 97
- static final int `ENUM_AUDIOCODEC_AMR` = 98
- static final int `ENUM_AUDIOCODEC_AMRWB` = 99
- static final int `ENUM_AUDIOCODEC_SPEEX` = 100
- static final int `ENUM_AUDIOCODEC_SPEEXWB` = 102
- static final int `ENUM_AUDIOCODEC_ISACWB` = 103
- static final int `ENUM_AUDIOCODEC_ISACSWB` = 104
- static final int `ENUM_AUDIOCODEC_OPUS` = 105
- static final int `ENUM_AUDIOCODEC_DTMF` = 101
- static final int `ENUM_VIDEOCODEC_NONE` = -1
- static final int `ENUM_VIDEOCODEC_I420` = 133
- static final int `ENUM_VIDEOCODEC_H263` = 34
- static final int `ENUM_VIDEOCODEC_H263_1998` = 115
- static final int `ENUM_VIDEOCODEC_H264` = 125
- static final int `ENUM_VIDEOCODEC_VP8` = 120
- static final int `ENUM_RESOLUTION_NONE` = 0
- static final int `ENUM_RESOLUTION_QCIF` = 1
- static final int `ENUM_RESOLUTION_CIF` = 2
- static final int `ENUM_RESOLUTION_VGA` = 3
- static final int `ENUM_RESOLUTION_SVGA` = 4
- static final int `ENUM_RESOLUTION_XVGA` = 5
- static final int `ENUM_RESOLUTION_720P` = 6
- static final int `ENUM_RESOLUTION_QVGA` = 7
- static final int `ENUM_SRTPPOLICY_NONE` = 0
- static final int `ENUM_SRTPPOLICY_FORCE` = 1
- static final int `ENUM_SRTPPOLICY_PREFER` = 2
- static final int `ENUM_TRANSPORT_UDP` = 0
- static final int `ENUM_TRANSPORT_TLS` = 1
- static final int `ENUM_TRANSPORT_TCP` = 2
- static final int `ENUM_TRANSPORT_PERS` = 3
- static final int `ENUM_LOG_LEVEL_NONE` = -1
- static final int `ENUM_LOG_LEVEL_ERROR` = 1
- static final int `ENUM_LOG_LEVEL_WARNING` = 2
- static final int `ENUM_LOG_LEVEL_INFO` = 3
- static final int `ENUM_LOG_LEVEL_DEBUG` = 4
- static final int `ENUM_DTMF_MOTHOD_RFC2833` = 0
- static final int `ENUM_DTMF_MOTHOD_INFO` = 1
- static final int `ENUM_ROTATE_CAPTURE_FRAME_0` = 0
- static final int `ENUM_ROTATE_CAPTURE_FRAME_90` = 90
- static final int `ENUM_ROTATE_CAPTURE_FRAME_180` = 180
- static final int `ENUM_ROTATE_CAPTURE_FRAME_270` = 270
- static final int `ENUM_AUDIOSTREAM_NONE` = 0
- static final int `ENUM_AUDIOSTREAM_LOCAL_MIX` = 1
- static final int `ENUM_AUDIOSTREAM_LOCAL_PER_CHANNEL` = 2
- static final int `ENUM_AUDIOSTREAM_REMOTE_MIX` = 3
- static final int `ENUM_AUDIOSTREAM_REMOTE_PER_CHANNEL` = 4

- static final int [ENUM_VIDEOSTREAM_NONE](#) = 0
- static final int [ENUM_VIDEOSTREAM_LOCAL](#) = 1
- static final int [ENUM_VIDEOSTREAM_REMOTE](#) = 2
- static final int [ENUM_VIDEOSTREAM_BOTH](#) = 3
- static final int [ENUM_RECORD_MODE_RECV](#) = 1
- static final int [ENUM_RECORD_MODE_SEND](#) = 2
- static final int [ENUM_RECORD_MODE_BOTH](#) = 3
- static final int [ENUM_AUDIO_FILE_FORMAT_WAVE](#) = 1
- static final int [ENUM_AUDIO_FILE_FORMAT_AMR](#) = 2
- static final int [ENUM_EC_NONE](#) = 0
Type of Echo Control.
- static final int [ENUM_EC_DEFAULT](#) = 1
Disable AEC.
- static final int [ENUM_EC_AECM_1](#) = 4
Platform default AEC.
- static final int [ENUM_EC_AECM_2](#) = 5
Mobile platform (iOS,Android) with most earpiece used.
- static final int [ENUM_EC_AECM_3](#) = 6
Mobile platform (iOS,Android) with loud earpiece or quiet speakerphone used.
- static final int [ENUM_EC_AECM_4](#) = 7
Mobile platform (iOS,Android) with most speakerphone used (Mobile Platform in default)
- static final int [ENUM_AGC_NONE](#) = 0
Mobile platform (iOS,Android) with oud speakerphone.
- static final int [ENUM_AGC_DEFAULT](#) = 1
- static final int [ENUM_AGC_ADAPTIVE_ANALOG](#) = 2
- static final int [ENUM_AGC_ADAPTIVE_DIGITAL](#) = 3
- static final int [ENUM_AGC_FIXED_DIGITAL](#) = 4
- static final int [ENUM_NS_NONE](#) = 0
Type of Noise Suppression.
- static final int [ENUM_NS_DEFAULT](#) = 1
- static final int [ENUM_NS_Conference](#) = 2
- static final int [ENUM_NS_LOW_SUPPRESSION](#) = 3
- static final int [ENUM_NS_MODERATE_SUPPRESSION](#) = 4
- static final int [ENUM_NS_HIGH_SUPPRESSION](#) = 5
- static final int [ENUM_NS_VERY_HIGH_SUPPRESSION](#) = 6

Member Data Documentation

final int com.portsip.PortSipEnumDefine.ENUM_VIDEOCODEC_NONE = -1 [static]

Used in startRecord only

final int com.portsip.PortSipEnumDefine.ENUM_VIDEOCODEC_I420 = 133 [static]

Used in startRecord only

final int com.portsip.PortSipEnumDefine.ENUM_AUDIOSTREAM_LOCAL_MIX = 1 [static]

Callback the audio stream from microphone for all channels.

final int com.portsip.PortSipEnumDefine.ENUM_AUDIOSTREAM_LOCAL_PER_CHANNEL = 2[static]

Callback the audio stream from microphone for one channel base on the session ID

final int com.portsip.PortSipEnumDefine.ENUM_AUDIOSTREAM_REMOTE_MIX = 3[static]

Callback the received audio stream that mixed including all channels.

final int com.portsip.PortSipEnumDefine.ENUM_AUDIOSTREAM_REMOTE_PER_CHANNEL = 4[static]

Callback the received audio stream for one channel base on the session ID.

final int com.portsip.PortSipEnumDefine.ENUM_VIDEOSTREAM_NONE = 0[static]

Disable video stream callback

final int com.portsip.PortSipEnumDefine.ENUM_VIDEOSTREAM_LOCAL = 1[static]

Local video stream callback

final int com.portsip.PortSipEnumDefine.ENUM_VIDEOSTREAM_REMOTE = 2[static]

Remote video stream callback

final int com.portsip.PortSipEnumDefine.ENUM_VIDEOSTREAM_BOTH = 3[static]

Both of local and remote video stream callback

final int com.portsip.PortSipEnumDefine.ENUM_RECORD_MODE_RECV = 1[static]

record only received

final int com.portsip.PortSipEnumDefine.ENUM_RECORD_MODE_SEND = 2[static]

record only sent out

final int com.portsip.PortSipEnumDefine.ENUM_RECORD_MODE_BOTH = 3[static]

record both sent out and received

final int com.portsip.PortSipEnumDefine.ENUM_AGC_NONE = 0[static]

Mobile platform (iOS,Android) with out speakerphone.

Type of Automatic Gain Control

The documentation for this class was generated from the following file:

- PortSipEnumDefine.java

com.portsip.PortSipErrorcode Class Reference

Static Public Attributes

- static final int **ECoreErrorNone** = 0
- static final int **INVALID_SESSION_ID** = -1
- static final int **ECoreAlreadyInitialized** = -60000
- static final int **ECoreNotInitialized** = -60001
- static final int **ECoreSDKObjectNull** = -60002
- static final int **ECoreArgumentNull** = -60003
- static final int **ECoreInitializeWinsockFailure** = -60004
- static final int **ECoreUserNameAuthNameEmpty** = -60005
- static final int **ECoreInitiazeStackFailure** = -60006
- static final int **ECorePortOutOfRange** = -60007
- static final int **ECoreAddTcpTransportFailure** = -60008
- static final int **ECoreAddTlsTransportFailure** = -60009
- static final int **ECoreAddUdpTransportFailure** = -60010
- static final int **ECoreNotSupportMediaType** = -60011
- static final int **ECoreNotSupportDTMFValue** = -60012
- static final int **ECoreAlreadyRegistered** = -60021
- static final int **ECoreSIPServerEmpty** = -60022
- static final int **ECoreExpiresValueTooSmall** = -60023
- static final int **ECoreCallIdNotFound** = -60024
- static final int **ECoreNotRegistered** = -60025
- static final int **ECoreCalleeEmpty** = -60026
- static final int **ECoreInvalidUri** = -60027
- static final int **ECoreAudioVideoCodecEmpty** = -60028
- static final int **ECoreNoFreeDialogSession** = -60029
- static final int **ECoreCreateAudioChannelFailed** = -60030
- static final int **ECoreSessionTimerValueTooSmall** = -60040
- static final int **ECoreAudioHandleNull** = -60041
- static final int **ECoreVideoHandleNull** = -60042
- static final int **ECoreCallIsClosed** = -60043
- static final int **ECoreCallAlreadyHold** = -60044
- static final int **ECoreCallNotEstablished** = -60045
- static final int **ECoreCallNotHold** = -60050
- static final int **ECoreSipMessaegEmpty** = -60051
- static final int **ECoreSipHeaderNotExist** = -60052
- static final int **ECoreSipHeaderValueEmpty** = -60053
- static final int **ECoreSipHeaderBadFormed** = -60054
- static final int **ECoreBufferTooSmall** = -60055
- static final int **ECoreSipHeaderValueListEmpty** = -60056
- static final int **ECoreSipHeaderParserEmpty** = -60057
- static final int **ECoreSipHeaderValueListNull** = -60058
- static final int **ECoreSipHeaderNameEmpty** = -60059
- static final int **ECoreAudioSampleNotmultiple** = -60060
- static final int **ECoreAudioSampleOutOfRange** = -60061
- static final int **ECoreInviteSessionNotFound** = -60062
- static final int **ECoreStackException** = -60063
- static final int **ECoreMimeTypeUnknown** = -60064
- static final int **ECoreDataSizeTooLarge** = -60065
- static final int **ECoreSessionNumsOutOfRange** = -60066
- static final int **ECoreNotSupportCallbackMode** = -60067
- static final int **ECoreNotFoundSubscribeId** = -60068
- static final int **ECoreCodecNotSupport** = -60069

- static final int **ECoreCodecParameterNotSupport** = -60070
- static final int **ECorePayloadOutOfRange** = -60071
- static final int **ECorePayloadHasExist** = -60072
- static final int **ECoreFixPayloadCantChange** = -60073
- static final int **ECoreCodecTypeInvalid** = -60074
- static final int **ECoreCodecWasExist** = -60075
- static final int **ECorePayloadTypeInvalid** = -60076
- static final int **ECoreArgumentTooLong** = -60077
- static final int **ECoreMiniRtpPortMustIsEvenNum** = -60078
- static final int **ECoreCallInHold** = -60079
- static final int **ECoreNotIncomingCall** = -60080
- static final int **ECoreCreateMediaEngineFailure** = -60081
- static final int **ECoreAudioCodecEmptyButAudioEnabled** = -60082
- static final int **ECoreVideoCodecEmptyButVideoEnabled** = -60083
- static final int **ECoreNetworkInterfaceUnavailable** = -60084
- static final int **ECoreWrongDTMFTone** = -60085
- static final int **ECoreWrongLicenseKey** = -60086
- static final int **ECoreTrialVersionLicenseKey** = -60087
- static final int **ECoreOutgoingAudioMuted** = -60088
- static final int **ECoreOutgoingVideoMuted** = -60089
- static final int **ECoreIVRObjectNull** = -61001
- static final int **ECoreIVRIndexOutOfRange** = -61002
- static final int **ECoreIVRReferFailure** = -61003
- static final int **ECoreIVRWaitingTimeOut** = -61004
- static final int **EAudioFileNameEmpty** = -70000
- static final int **EAudioChannelNotFound** = -70001
- static final int **EAudioStartRecordFailure** = -70002
- static final int **EAudioRegisterRecodingFailure** = -70003
- static final int **EAudioRegisterPlaybackFailure** = -70004
- static final int **EAudioGetStatisticsFailure** = -70005
- static final int **EAudioPlayFileAlreadyEnable** = -70006
- static final int **EAudioPlayObjectNotExist** = -70007
- static final int **EAudioPlaySteamNotEnabled** = -70008
- static final int **EAudioRegisterCallbackFailure** = -70009
- static final int **EAudioCreateAudioConferenceFailure** = -70010
- static final int **EAudioOpenPlayFileFailure** = -70011
- static final int **EAudioPlayFileModeNotSupport** = -70012
- static final int **EAudioPlayFileFormatNotSupport** = -70013
- static final int **EAudioPlaySteamAlreadyEnabled** = -70014
- static final int **EAudioCreateRecordFileFailure** = -70015
- static final int **EAudioCodecNotSupport** = -70016
- static final int **EAudioPlayFileNotEnabled** = -70017
- static final int **EAudioPlayFileUnknowSeekOrigin** = -70018
- static final int **EAudioCantSetDeviceIdDuringCall** = -70019
- static final int **EVideoFileNameEmpty** = -80000
- static final int **EVideoGetDeviceNameFailure** = -80001
- static final int **EVideoGetDeviceIdFailure** = -80002
- static final int **EVideoStartCaptureFailure** = -80003
- static final int **EVideoChannelNotFound** = -80004
- static final int **EVideoStartSendFailure** = -80005
- static final int **EVideoGetStatisticsFailure** = -80006
- static final int **EVideoStartPlayAviFailure** = -80007
- static final int **EVideoSendAviFileFailure** = -80008
- static final int **EVideoRecordUnknowCodec** = -80009
- static final int **EDeviceGetDeviceNameFailure** = -90001

The documentation for this class was generated from the following file:

- PortSipErrorcode.java

com.portsip.PortSipSdk Class Reference

Classes

- class **MainHandler**

Public Member Functions

- void [CreateCallManager](#) (Context context)
- void [DeleteCallManager](#) ()
- int [initialize](#) (int enum_transport, int enum_LogLevel, String LogPath, int maxLines, String agent, int audioDeviceLayer, int videoDeviceLayer)
- int [setInstanceId](#) (String instanceId)
- int [setUser](#) (String userName, String displayName, String authName, String password, String localIP, int localSIPPort, String userDomain, String SIPServer, int SIPServerPort, String STUNServer, int STUNServerPort, String outboundServer, int outboundServerPort)
- int [registerServer](#) (int expires, int retryTimes)
- int [unRegisterServer](#) ()
- int [setLicenseKey](#) (String key)
- int [addAudioCodec](#) (int enum_audiocodec)
- int [addVideoCodec](#) (int enum_videocodec)
- boolean [isAudioCodecEmpty](#) ()
- boolean [isVideoCodecEmpty](#) ()
- int [setAudioCodecPayloadType](#) (int enum_audiocodec, int payloadType)
- int [setVideoCodecPayloadType](#) (int enum_videocodec, int payloadType)
- void [clearAudioCodec](#) ()
- void [clearVideoCodec](#) ()
- int [setAudioCodecParameter](#) (int enum_audiocodec, String sdpParameter)
- int [setVideoCodecParameter](#) (int enum_videocodec, String sdpParameter)
- int [setDisplayname](#) (String displayName)
- String [getVersion](#) ()
- int [enableReliableProvisional](#) (boolean enable)
- int [enable3GppTags](#) (boolean enable)
- void [enableCallbackSendingSignaling](#) (boolean enable)
- void [setSrtpPolicy](#) (int enum_srtpolicy)
- int [setRtpPortRange](#) (int minimumRtpAudioPort, int maximumRtpAudioPort, int minimumRtpVideoPort, int maximumRtpVideoPort)
- int [setRtcpPortRange](#) (int minimumRtcpAudioPort, int maximumRtcpAudioPort, int minimumRtcpVideoPort, int maximumRtcpVideoPort)
- int [enableCallForward](#) (boolean forBusyOnly, String forwardTo)
- int [disableCallForward](#) ()
- int [enableSessionTimer](#) (int timerSeconds)
- void [disableSessionTimer](#) ()
- void [setDoNotDisturb](#) (boolean forBusyOnly)
- int [detectMwi](#) ()
- int [enableCheckMwi](#) (boolean state)
- int [setRtpKeepAlive](#) (boolean state, int keepAlivePayloadType, int deltaTransmitTimeMS)
- int [setKeepAliveTime](#) (int keepAliveTime)
- int [setAudioSamples](#) (int ptime, int maxptime)
- int [addSupportedMimeType](#) (String methodName, String mimeType, String subMimeType)
- String [getExtensionHeaderValue](#) (String sipMessage, String headerName)
- int [addExtensionHeader](#) (String headerName, String headerValue)
- int [clearAddExtensionHeaders](#) ()
- int [modifyHeaderValue](#) (String headerName, String headerValue)
- int [clearModifyHeaders](#) ()

- int [setVideoDeviceId](#) (int deviceId)
- int [setVideoResolution](#) (int width, int height)
- int [setAudioBitrate](#) (long sessionId, int enum_codec, int bitrateKbps)
- int [setVideoBitrate](#) (int bitrateKbps)
- int [setVideoFrameRate](#) (int frameRate)
- int [sendVideo](#) (long sessionId, boolean send)
- int [setVideoOrientation](#) (int enum_rotation)
- void [setLocalVideoWindow](#) (Object localVideoView)
- int [setRemoteVideoWindow](#) (long sessionId, Object remoteVideoView)
- void [displayLocalVideo](#) (boolean state)
- int [setVideoNackStatus](#) (boolean state)
- void [muteMicrophone](#) (boolean mute)
- void [muteSpeaker](#) (boolean mute)
- int [getDynamicSpeakerVolumeLevel](#) ()
- int [getDynamicMicrophoneVolumeLevel](#) ()
- int [setChannelOutputVolumeScaling](#) (long sessionId, int scaling)
- int [setLoudspeakerStatus](#) (boolean useSpeaker)
- long [call](#) (String callee, boolean sendSdp, boolean videoCall)
- int [rejectCall](#) (long sessionId, int code)
- int [hangUp](#) (long sessionId)
- int [answerCall](#) (long sessionId, boolean videoCall)
- int [updateCall](#) (long sessionId, boolean enableAudio, boolean enableVideo)
- int [hold](#) (long sessionId)
- int [unHold](#) (long sessionId)
- int [muteSession](#) (long sessionId, boolean muteIncomingAudio, boolean muteOutgoingAudio, boolean muteIncomingVideo, boolean muteOutgoingVideo)
- int [forwardCall](#) (long sessionId, String forwardTo)
- int [sendDtmf](#) (long sessionId, int enum_dtmfMethod, int code, int dtmfDuration, boolean playDtmfTone)
- int [refer](#) (long sessionId, String referTo)
- int [attendedRefer](#) (long sessionId, long replaceSessionId, String referTo)
- long [acceptRefer](#) (long referId, String referSignaling)
- int [rejectRefer](#) (long referId)
- int [enableSendPcmStreamToRemote](#) (long sessionId, boolean state, int streamSamplesPerSec)
- int [sendPcmStreamToRemote](#) (long sessionId, byte[] data, int dataLength)
- int [enableSendVideoStreamToRemote](#) (long sessionId, boolean state)
- int [sendVideoStreamToRemote](#) (long sessionId, byte[] data, int dataLength, int width, int height)
- void [setRtpCallback](#) (boolean enable)
- void [enableAudioStreamCallback](#) (long sessionId, boolean enable, int enum_audioCallbackMode)
- void [enableVideoStreamCallback](#) (long sessionId, int enum_videoCallbackMode)
- void [enableVideoDecoderCallback](#) (boolean enable)
- int [startRecord](#) (long sessionId, String recordFilePath, String recordFileName, boolean appendTimeStamp, int enum_audioFileFormat, int enum_audioRecordMode, int enum_videocodec, int enum_videoRecordMode)
- int [stopRecord](#) (long sessionId)
- int [playVideoFileToRemote](#) (long sessionId, String aviFile, boolean loop, boolean playAudio)
- int [stopPlayVideoFileToRemote](#) (long sessionId)
- int [playAudioFileToRemote](#) (long sessionId, String filename, int fileSamplesPerSec, boolean loop)
- int [stopPlayAudioFileToRemote](#) (long sessionId)
- int [playAudioFileToRemoteAsBackground](#) (long sessionId, String filename, int fileSamplesPerSec)
- int [stopPlayAudioFileToRemoteAsBackground](#) (long sessionId)
- int [createConference](#) (Object conferenceVideoWindow, int videoWidth, int videoHeight, boolean displayLocalVideoInConference)
- void [destroyConference](#) ()
- int [setConferenceVideoWindow](#) (Object conferenceVideoWindow)
- int [joinToConference](#) (long sessionId)

- int [removeFromConference](#) (long sessionId)
- int [setAudioRtcpBandwidth](#) (long sessionId, int BitsRR, int BitsRS, int KBitsAS)
- int [setVideoRtcpBandwidth](#) (long sessionId, int BitsRR, int BitsRS, int KBitsAS)
- int [setAudioQos](#) (boolean enable, int DSCPValue, int priority)
- int [setVideoQos](#) (boolean enable, int DSCPValue)
- int [setVideoMTU](#) (int mtu)
- int [getNetworkStatistics](#) (long sessionId, int[] statistics)
- int [getAudioRtpStatistics](#) (long sessionId, int[] statistics)
- int [getAudioRtcpStatistics](#) (long sessionId, int[] statistics)
- int [getVideoRtpStatistics](#) (long sessionId, int[] statistics)
- void [enableVAD](#) (boolean state)
- void [enableAEC](#) (int enum_aecMode)
- void [enableCNG](#) (boolean state)
- void [enableAGC](#) (int enum_agcMode)
- void [enableANS](#) (int enum_nsMode)
- int [sendOptions](#) (String to, String sdp)
- int [sendInfo](#) (long sessionId, String mimeType, String subMimeType, String infoContents)
- long [sendMessage](#) (long sessionId, String mimeType, String subMimeType, byte[] message, int messageLength)
- long [sendOutOfDialogMessage](#) (String to, String mimeType, String subMimeType, byte[] message, int messageLength)
- long [presenceSubscribeContact](#) (String contact, String subject)
- int [presenceRejectSubscribe](#) (long subscribeId)
- int [presenceAcceptSubscribe](#) (long subscribeId)
- int [presenceOnline](#) (long subscribeId, String statusText)
- int [presenceOffline](#) (long subscribeId)
- int [getNumOfRecordingDevices](#) ()
- int [getNumOfPlayoutDevices](#) ()
- String [getRecordingDeviceName](#) (int index)
- String [getPlayoutDeviceName](#) (int index)
- int [setSpeakerVolume](#) (int volume)
- int [getSpeakerVolume](#) ()
- int [setSystemOutputMute](#) (boolean mute)
- boolean [getSystemOutputMute](#) ()
- int [setMicVolume](#) (int volume)
- int [getMicVolume](#) ()
- int [setSystemInputMute](#) (boolean mute)
- boolean [getSystemInputMute](#) ()
- void [audioPlayLoopbackTest](#) (boolean enable)
- int [getNumOfVideoCaptureDevices](#) ()
- String [getVideoCaptureDeviceName](#) (int index)
- void [receiveSIPEvent](#) (int sipCommand)
- void [receivedRTPPacket](#) (long sessionId, boolean isAudio, byte[] RTPPacket, int packetSize)
- void [sendingRTPPacket](#) (long sessionId, boolean isAudio, byte[] RTPPacket, int packetSize)
- void [audioRawCallback](#) (long sessionId, int enum_audioCallbackMode, byte[] data, int dataLength, int samplingFreqHz)
- void [videoRawCallback](#) (long sessionId, int enum_videoCallbackMode, int width, int height, byte[] data, int dataLength)
- void [videoDecodedInfoCallback](#) (long sessionId, int width, int height, int framerate, int bitrate)
- void [setOnPortSIPEvent](#) ([OnPortSIPEvent](#) l)

Detailed Description

Author:

PortSIP Solutions, Inc. All rights reserved.

Version:

11.2

The documentation for this class was generated from the following file:

- PortSipSdk.java

com.portsip.Renderer Class Reference

Static Public Member Functions

- static SurfaceView **CreateRenderer** (Context context)
- static SurfaceView **CreateRenderer** (Context context, boolean useOpenGLES2)
- static SurfaceView **CreateLocalRenderer** (Context context)
- static SurfaceHolder **GetLocalRenderer** ()

The documentation for this class was generated from the following file:

- `Renderer.java`

com.portsip.SurfaceRenderer Class Reference

Inherits Callback.

Public Member Functions

- **SurfaceRenderer** (SurfaceView view)
- void **surfaceChanged** (SurfaceHolder holder, int format, int in_width, int in_height)
- void **surfaceCreated** (SurfaceHolder holder)
- void **surfaceDestroyed** (SurfaceHolder holder)
- Bitmap **CreateBitmap** (int width, int height)
- ByteBuffer **CreateByteBuffer** (int width, int height)
- void **SetCoordinates** (float left, float top, float right, float bottom)
- void **DrawByteBuffer** ()
- void **DrawBitmap** ()

The documentation for this class was generated from the following file:

- SurfaceRenderer.java

com.portsip.VideoCaptureAndroid Class Reference

Inherits PreviewCallback, and Callback.

Public Member Functions

- **VideoCaptureAndroid** (int id, long native_capturer)
- synchronized void **onPreviewFrame** (byte[] data, Camera camera)
- synchronized void **surfaceChanged** (SurfaceHolder holder, int format, int width, int height)
- synchronized void **surfaceCreated** (SurfaceHolder holder)
- synchronized void **surfaceDestroyed** (SurfaceHolder holder)

Static Public Attributes

- static Camera **camera**

Protected Member Functions

- void **finalize** () throws Throwable

The documentation for this class was generated from the following file:

- VideoCaptureAndroid.java

com.portsip.VideoCaptureDeviceInfoAndroid Class Reference

The documentation for this class was generated from the following file:

- VideoCaptureDeviceInfoAndroid.java

Index

INDEX