PortSIP VoIP SDK Manual for Android

Version 11.2.3

Table of Contents

Welcome to the PortSIP VoIP SDK

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

The award-winning PortSIP VoIP SDK is a powerful and highly versatile set of tools to dramatically accelerate SIP application development. It includes a suite of stacks, SDKs, Sample projects. Each one 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 the PortSIP VoIP SDK Sample projects at our Website, the samples include for VC++, C#, VB.NET, Delphi XE, XCode(for iOS and Mac OS), Eclipse(Java, for Android), the sample project source code is provided(not include SDK source code). The sample projects demonstrate how to create a SIP application base on our SDK, powerful, easy and quick.

Contents

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

SDK User Manual

The starting point for the documentation of PortSIP VoIP SDK is the SDK User Manual page, which gives a brief description of each API functions.

Web Site

Some general interest or often changing PortSIP SDK information lives only on the PortSIP web site. The release contains links to the site, so while browsing it you'll see occasional broken links if you aren't connected to the Internet. But everything needed to use the PortSIP VoIP SDK is contained within the release.

Support

Please send email to Our support if you need any helps.

Machine Requirements

Development using the PortSIP VoIP/IMS SDK for Android requires minimum sdk version API-9

Frequently Asked Questions

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

```
All sample projects of the PortSIP VoIP SDK can be download to test at:
                 http://www.portsip.com/downloads
                 http://www.portsip.com/portsip-voip-sdk.
```

2. How to compile the sample project?

```
1. Download the sample projects 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 2-3 minutes conversation.
```

3. Create a new project base on PortSIP VoIP SDK

```
  1).Download the Sample project and evaluation SDK and extract it to a directory
  2).Run the eclipse and create a new Android Applicatiion 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, example:
          import com.portsip.OnPortSIPEvent;
          import com.portsip.PortSipSdk;
  5).Inherit the interface OnPortSIPEvent to process the callback events.
  6). Initialize sdk, Example:
          mPortSIPSDK = new PortSipSdk();
          mPortSIPSDK.setOnPortSIPEvent(instanceofOnPortSIPEvent);
          mPortSIPSDK.CreateCallManager(context);
          mPortSIPSDK.initialize(...);
  7).More details please read the Sample project source code.
```

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

```
1) Download and extract the SDK sample project .zip file, compile and run the "P2PSample" project.

2) Run the P2Psample on two devices, for example, run it on device A and device B, A IP address is
192.168.1.10   B IP address is 192.168.1.11.

3) Enter a user name and password on A, for example, user name is 111, password is aaa(you can enter
anything for  the password, the SDK will ignore it). Enter a user name and password on B, for example:
user name is 222, password is aaa.

4) Click the "Initialize" button on A and B. If the default port 5060 in using, the P2PSample will
said "Initialize failure". In case please click the "Uninitialize" button and change the local port,
click the "Initialize" button again.

5) The log box will appears "Initialized." if the SDK initialize succeeded.

6) Make call from A to B, enter: sip:222@192.168.1.11 and click "Dial" button; Make call from B to
A, enter: sip:111@192.168.1.10.
```

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

5. Does the SDK is thread safe?

```
Yes, the SDK is thread safe, you can call all the API functions don't need to consider the multiple
threads. Note: the SDK allows call API functions in callback events directly - except the
"onAudioRawCallback", "onVideoRawCallback", "onReceivedRtpPacket", "onSendingRtpPacket" callbacks.
```

Module Index

Modules

Here is a list of all modules:

Hierarchical Index

Class Hierarchy

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

Class Index

Class List

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

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 register to server, this event will be triggered.

Parameters:

reason	The status text.
code	The status code.

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

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

Parameters:

reason	The status text.
code	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 the call is coming, this event was triggered.

Parameters:

sessionId	The session ID of the call.
callerDisplayName	The display name of caller
caller	The caller.
calleeDisplayName	The display name of callee.
callee	The callee.
audioCodecs	The matched audio codecs, it's separated by "#" if have more than one codec.
videoCodecs	The matched video codecs, it's separated by "#" if have more than one codec.
existsAudio	If it's true means this call include the audio.
existsVideo	If it's true means this call include the video.

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

If the outgoing call was processing, this event triggered.

Parameters:

sessionId	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 will be triggered.

Parameters:

sessionId	The session ID of the call.
audioCodecs	The matched audio codecs, it's separated by "#" if have more than one codec.
videoCodecs	The matched video codecs, it's separated by "#" if have more than one codec.
existsEarlyMedia	If it's true means the call has early media.
existsAudio	If it's true means this call include the audio.
existsVideo	If it's true means this call include the video.

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

If the out going call was ringing, this event triggered.

Parameters:

sessionId	The session ID of the call.
statusText	The status text.
statusCode	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 was answered the call, this event triggered.

Parameters:

sessionId	The session ID of the call.
callerDisplayName	The display name of caller
caller	The caller.
calleeDisplayName	The display name of callee.
callee	The callee.
audioCodecs	The matched audio codecs, it's separated by "#" if have more than one codec.
videoCodecs	The matched video codecs, it's separated by "#" if have more than one codec.
existsAudio	If it's true means this call include the audio.
existsVideo	If it's true means this call include the video.

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

If the outgoing call is fails, this event triggered.

Parameters:

sessionId	The session ID of the call.
reason	The failure reason.
code	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 updated this call.

Parameters:

sessionId	The session ID of the call.
audioCodecs	The matched audio codecs, it's separated by "#" if have more than one codec.
videoCodecs	The matched video codecs, it's separated by "#" if have more than one codec.
existsAudio	If it's true means this call include the audio.
existsVideo	If it's true means this call include 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 called only after the call connected, otherwise the functions will return error.

Parameters:

sessionId	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 trigger this event.

Parameters:

forwardTo	The forward target SIP URI.

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

This event is triggered once remote side close the call.

Parameters:

sessionId	The session ID of the call.

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

If the remote side has placed the call on hold, this event triggered.

Parameters:

sessionId	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 was un-hold the call, this event triggered

Parameters:

sessionId	The session ID of the call.
audioCodecs	The matched audio codecs, it's separated by "#" if have more than one codec.
videoCodecs	The matched video codecs, it's separated by "#" if have more than one codec.
existsAudio	If it's true means this call include the audio.
existsVideo	If it's true means this call include 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:

sessionId	The session ID of the call.
referId	The ID of the REFER message, pass it to acceptRefer or rejectRefer
to	The refer target.
from	The sender of REFER message.
referSipMessage	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 called "acceptRefer" to accept the REFER

Parameters:

sessionId	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 called "rejectRefer" to reject the REFER

Parameters:

sessionId	The session ID of the call.
reason	Reject reason.
code	Reject code.

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

When the refer call is processing, this event trigged.

Parameters:

sessionId	The session ID of the call.

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

When the refer call is ringing, this event trigged.

Parameters:

sessionId	The session ID of the call.

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

When the refer call is succeeds, this event will be triggered. The ACTV means Active. For example: A established the call with B, A transfer B to C, C accepted the refer call, A received this event.

Parameters:

sessionId	The session ID of the call.

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

When the refer call is fails, this event will be triggered. The ACTV means Active. For example: A established the call with B, A transfer B to C, C rejected this refer call, A will received this event.

Parameters:

sessionId	The session ID of the call.
reason	The error reason.
code	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 received a SIP message.

Parameters:

sessionId	The session ID of the call.
message	The SIP message which is received.

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

This event will be triggered when sent a SIP message.

Parameters:

sessionId	The session ID of the call.
message	The SIP message which is sent.

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 has the waiting voice message(MWI), then this event will be triggered.

Parameters:

messageAccount	Voice message account
urgentNewMessageCount	Urgent new message count.
urgentOldMessageCount	Urgent old message count.
newMessageCount	New message count.
oldMessageCount	Old message count.

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

If has the waiting fax message(MWI), then this event will be triggered.

Parameters:

messageAccount	Fax message account
urgentNewMessageCount	Urgent new message count.
urgentOldMessageCount	Urgent old message count.
newMessageCount	New message count.
oldMessageCount	Old message count.

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 received a DTMF tone from remote side.

Parameters:

sessionId	The session ID of the call.
tone	

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 received the OPTIONS message.

Parameters:

optionsMessage	The received whole OPTIONS message in text format.

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

This event will be triggered when received the INFO message.

Parameters:

infoMessage	The received whole INFO message 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 received the SUBSCRIBE request from a contact.

Parameters:

subscribeId	The id of SUBSCRIBE request.
fromDisplayName	The display name of contact.
from	The contact who send the SUBSCRIBE request.
subject	The subject of the SUBSCRIBE request.

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

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

Parameters:

fromDisplayName	The display name of contact.
from	The contact who send the SUBSCRIBE request.
stateText	The presence status text.

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

When the contact is went offline then this event will be triggered.

Parameters:

fromDisplayName	The display name of contact.
from	The contact who send the SUBSCRIBE request

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

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

Parameters:

sessionId	The session ID of the call.
mimeType	The message mime type.
subMimeType	The message sub mime type.
messageData	The received message body, it's 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", then "messageData" is text messsage body. if the mimeType is "application" and subMimeType is "vnd.3gpp.sms", then "messageData" is binary messsage body.messageDataLength
messageDataLength	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 received a MESSAGE message out of dialog, for example: pager message.

Parameters:

fromDisplayName	The display name of sender.
from	The message sender.

toDisplayName	The display name of receiver.
to	The receiver.
mimeType	The message mime type.
subMimeType	The message sub mime type.
messageData	The received message body, it's 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", then "messageData" is text messsage body. if the mimeType is "application" and subMimeType is "vnd.3gpp.sms", then "messageData" is binary messsage body.
messageDataLength	The length of "messageData".

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

If the message was sent succeeded in dialog, this event will be triggered.

Parameters:

sessionId	The session ID of the call.
messageId	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 was sent failure out of dialog, this event will be triggered.

Parameters:

sessionId	The session ID of the call.
messageId	The message ID, it's equals the return value of sendMessage function.
reason	The failure reason.
code	Failure code.

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

If the message was sent succeeded out of dialog, this event will be triggered.

Parameters:

messageId	The message ID, it's equals the return value of SendOutOfDialogMessage function.
fromDisplayName	The display name of message sender.
from	The message sender.
toDisplayName	The display name of message receiver.
to	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 was sent failure out of dialog, this event will be triggered.

Parameters:

messageId	The message ID, it's equals the return value of SendOutOfDialogMessage function.
fromDisplayName	The display name of message sender
from	The message sender.
toDisplayName	The display name of message receiver.
to	The message receiver.
reason	The failure reason.

code	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:

sessionId	The session ID of the call.
fileName	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:

sessionId	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 called setRTPCallback function to enabled the RTP callback, this event will be triggered once received a RTP packet.

Parameters:

sessionId	The session ID of the call.
isAudio	If the received RTP packet is of audio, this parameter is true, otherwise false.
RTPPacket	The memory of whole RTP packet.
packetSize	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 will spend long time, 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 called setRTPCallback function to enabled the RTP callback, this event will be triggered once sending a RTP packet.

Parameters:

sessionId	The session ID of the call.
isAudio	If the received RTP packet is of audio, this parameter is true, otherwise false.
RTPPacket	The memory of whole RTP packet.
packetSize	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 will spend long time, 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 received the audio packets if called enableAudioStreamCallback function.

Parameters:

sessionId	The session ID of the call.
enum_audioCallbackMode	The type which pasdded in enableAudioStreamCallback function.allow:

ENUM_AUDIOSTREAM_NONE, ENUM_AUDIOSTREAM_LOCAL_MIX, ENUM_AUDIOSTREAM_LOCAL_PER_CHANNEL, ENUM_AUDIOSTREAM_REMOTE_MIX, ENUM_AUDIOSTREAM_REMOTE_PER_CHANNEL.

Parameters:

data	The memory of audio stream, it's PCM format.
dataLength	The data size.
samplingFreqHz	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 will spend long time, 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 received the video packets if called enableVideoStreamCallback function.

Parameters:

sessionId	The session ID of the call.
enum_videoCallbackMode	The type which pasdded in enableVideoStreamCallback function.allow: ENUM_VIDEOSTREAM_NONE, ENUM_VIDEOSTREAM_LOCAL, ENUM_VIDEOSTREAM_REMOTE, ENUM_VIDEOSTREAM_BOTH.
width	The width of video image.
height	The height of video image.
data	The memory of video stream, it's YUV420 format, YV12.
dataLength	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 received the video size is change, if called enableVideoDecoderCallback function.

Parameters:

sessionId	The session ID of the call.
width	The width of received video image.
height	The height of received video image.
framerate	The frame rate value of received video.
bitrate	The bitrate value of received video.

See also:

PortSipSdk::enableVideoDecoderCallback

SDK functions SDK functions

Modules

- Initialize and register functions Initialize and register
- Audio and video codecs functions
- Additional setting 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 remoe functions
- Conference functions
- RTP and RTCP QOS functions
- RTP statistics functions
- Audio effect functions
- Send OPTIONS/INFO/MESSAGE functions
- Device Manage functions.

Detailed Description

Initialize and register functions Initialize and register

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.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

functions

Function Documentation

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

Create the callback handlers.

Parameters:

context	The context of application.

void com.portsip.PortSipSdk.DeleteCallManager ()

Release the callbackHandlers.

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:

enum_transport	Transport for SIP signaling, it can be set as: ENUM_TRANSPORT_UDP, ENUM_TRANSPORT_TCP, ENUM_TRANSPORT_TLS, ENUM_TRANSPORT_PERS, The ENUM_TRANSPORT_PERS is the PortSIP private transport for anti the SIP blocking, it must using with the PERS
enum_LogLevel	Set the application log level, the SDK generate the "PortSIP_Log_datatime.log" file if the log 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
LogPath	The log file path, the path(folder) MUST is exists.
maxLines	In theory support unlimited lines just depends on the device capability, for SIP client recommend less than 1 - 100;
agent	The User-Agent header to insert in SIP messages.
audioDeviceLayer	Specifies which audio device layer should be using: 0 = Use the OS default device. 1 = Virtual device - Virtual device, usually use this for the device which no sound device installed. 2 = AndroidOpenSLES - Use the OpenSL ES for audio device, just valid to Android, if you got bad voice with this optional, please try AndroidAudioTrackJni. 3 = AndroidAudioTrackJni - Use Audio Track JNI for audio device, just valid to Android, if you got bad voice with this optional, please try AndroidOpenSLES.
videoDeviceLayer	Specifies which video device layer should be using: 0 = Use the OS default device. 1 = Use Virtual device, usually use this for the device which no camera installed.

Returns:

If the function succeeds, the return value is 0. If the function fails, the return value is 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:

userName	Account(User name) of the SIP, usually provided by an IP-Telephony service provider.
displayName	The display name of user, you can set it as your like, such as "James Kend". It's optional.
authName	Authorization user name (usually equals the username).
password	The password of user, it's optional.
localIP	The local computer IP address to bind (for example: 192.168.1.108), it will be using for send and receive SIP message and RTP packet. If pass this IP as the IPv6 format then the SDK using IPv6. If you want the SDK choose correct network interface(IP) automatically, please pass the "0.0.0.0"(for IPv4) or "::"(for IPv6).
localSIPPort	The SIP message transport listener port(for example: 5060).
userDomain	User domain; this parameter is optional that allow pass a empty string if you are not use domain.

SIPServer	SIP proxy server IP or domain(for example: xx.xxx.xx.x or sip.xxx.com).
SIPServerPort	Port of the SIP proxy server, (for example: 5060).
STUNServer	Stun server, use for NAT traversal, it's optional and can be pass empty string to disable STUN.
STUNServerPort	STUN server port,it will be ignored if the outboundServer is empty.
outboundServer	Outbound proxy server(for example: sip.domain.com), it's optional that allow pass a empty string if not use outbound server.
outboundServerPort	Outbound proxy server port, it will be ignored if the outboundServer is empty.

Returns:

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

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

Register to SIP proxy server(login to server)

Parameters:

expires	Registration refresh Interval in seconds, maximum is 3600, it will be inserted into SIP REGISTER message headers.
retryTimes	The retry times if failed to refresh the registration, set to <= 0 the retry will be disabled and onRegisterFailure callback triggered when retry failure.

Returns:

If the function succeeds, the return value is 0. If the function fails, the return value is a specific error code. if register to server succeeded then onRegisterSuccess will be triggered, otherwise onRegisterFailure triggered.

int com.portsip.PortSipSdk.unRegisterServer ()

Un-register from the SIP proxy server.

Returns:

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

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

Set the license key, must called before setUser function.

Parameters:

key	The SDK license key, please purchase from PortSIP

Returns:

If the function succeeds, the return value is 0. If the function fails, the return value is 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, it will be appears in SDP.

Parameters:

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

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

Enable a video codec, it will be appears in SDP.

Parameters:

enum_videocodec	Video codec type,allow: ENUM_VIDEOCODEC_H263, ENUM_VIDEOCODEC_H263_1998, ENUM_VIDEOCODEC_H264, ENUM_VIDEOCODEC_VP8.

Returns:

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

boolean com.portsip.PortSipSdk.isAudioCodecEmpty ()

Detect enabled audio codecs is empty or not.

Returns:

If no audio codec was enabled the return value is true, otherwise is false.

boolean com.portsip.PortSipSdk.isVideoCodecEmpty ()

Detect enabled video codecs is empty or not.

Returns:

If no video codec was enabled the return value is true, otherwise is false.

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

Set the RTP payload type for dynamic audio codec.

Parameters:

enum_audiocodec	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
payloadType	The new RTP payload type that you want to set.

Returns:

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

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

Set the RTP payload type for dynamic Video codec.

Parameters:

enum_videocodec	Video codec type, allow: ENUM_VIDEOCODEC_H263, ENUM_VIDEOCODEC_H263_1998, ENUM_VIDEOCODEC_H264, ENUM_VIDEOCODEC_VP8.
payloadType	The new RTP payload type that you want to set.

Returns:

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

void com.portsip.PortSipSdk.clearAudioCodec ()

Remove all enabled audio codecs.

void com.portsip.PortSipSdk.clearVideoCodec ()

Remove all enabled video codecs.

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

Set the codec parameter for audio codec.

Parameters:

enum_audiocodec	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

Parameters:

sdpParameter	The parameter in string format.

Returns:

If the function succeeds, the return value is 0. If the function fails, the return value is 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:

enum_videocodec	Video codec type, allow: ENUM_VIDEOCODEC_H263, ENUM_VIDEOCODEC_H263_1998, ENUM_VIDEOCODEC_H264, ENUM_VIDEOCODEC_VP8.
sdpParameter	The parameter in string format.

Returns:

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

Remarks:

Example:

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

Additional setting 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_srtppolicy)
- 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 user display name.

Parameters:

displayName	The display name.

Returns:

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

String com.portsip.PortSipSdk.getVersion ()

Get the current version number of the SDK.

Returns:

String a version description string

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

Enable/disable PRACK.

Parameters:

enable	Set to true to enable the SDK support PRACK, default the PRACK is disabled.

Returns:

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

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

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

Parameters:

enable	Set to true to enable the SDK support 3Gpp tags.

Returns:

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

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

Enable/disable callback the sending SIP messages.

Parameters:

enable	Set as true to enable callback the sent SIP messages, false to disable. Once enabled, the "onSendingSignaling" event will be fired once the SDK sending a SIP message.

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

Set the SRTP policy.

Parameters:

enum_srtppolicy	The SRTP policy.allow: ENUM_SRTPPOLICY_NONE,

	ENUM_SRTPPOLICY_FORCE, ENUM_SRTPPOLICY_PREFER.

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

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

Parameters:

minimumRtpAudioPort	The minimum RTP port for audio stream.
maximumRtpAudioPort	The maximum RTP port for audio stream.
minimumRtpVideoPort	The minimum RTP port for video stream.
maximumRtpVideoPort	The maximum RTP port for video stream.

Returns:

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

Remarks:

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

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

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

Parameters:

minimumRtcpAudioPort	The minimum RTCP port for audio stream.
maximumRtcpAudioPort	The maximum RTCP port for audio stream.
minimumRtcpVideoPort	The minimum RTCP port for video stream.
maximumRtcpVideoPort	The maximum RTCP port for video stream.

Returns:

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

Remarks:

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

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

Enable call forward.

Parameters:

forBusyOnly	If set this parameter as true, the SDK will forward all incoming calls when currently it's busy. If set this as false, the SDK forward all inconing calls anyway.
forwardTo	The call forward target, it's must likes sip:xxxx@sip.portsip.com.

Returns:

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

int com.portsip.PortSipSdk.disableCallForward ()

Disable the call forward, the SDK is not forward any incoming call after this function is called.

Returns:

If the function succeeds, the return value is 0. If the function fails, the return value is 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:

timerSeconds	The value of the refresh interval in seconds. Minimum requires 90 seconds.

Returns:

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

Remarks:

The repeated INVITE requests, or re-INVITEs, are sent during an active call leg to allow user agents (UA) or proxies to determine the status of a SIP session. Without this keepalive mechanism, proxies that remember incoming and outgoing requests (stateful proxies) may continue to retain call state needlessly. If a UA fails to send a BYE message at the end of a session or if the BYE message is lost because of network problems, a stateful proxy does 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 the "Do not disturb" to enable/disable.

Parameters:

forBusyOnly	If set to true, the SDK reject all incoming calls anyway.

int com.portsip.PortSipSdk.detectMwi ()

Use to obtain the MWI status.

Returns:

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

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

Allows enable/disable the check MWI(Message Waiting Indication).

Parameters:

state	If set as true will check MWI automatically once successfully registered to a SIP proxy server;

Returns:

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

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

Enable or disable send RTP keep-alive packet during the call is established.

Parameters:

state	Set to true allow send the keep-alive packet during the conversation;
keepAlivePayload Type	The payload type of the keep-alive RTP packet, usually set to 126.

deltaTransmitTime MS	The keep-alive RTP packet send interval, in millisecond, usually recommend 15000 - 300000.

Returns:

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

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

Enable or disable send SIP keep-alive packet.

Parameters:

keepAliveTime	This is the SIP keep alive time interval in seconds, set to 0 to disable the SIP keep alive, it's in seconds, recommend 30 or 50.

Returns:

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

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

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

Parameters:

ptime	It's should be a multiple of 10, and between 10 - 60(included 10 and 60).
maxptime	For the "maxptime" attribute, should be a multiple of 10, and between 10 - 60(included 10 and 60). Can't less than "ptime".

Returns:

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

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

Set the SDK receive the SIP message that include special mime type.

Parameters:

methodName	Method name of the SIP message, likes INVITE, OPTION, INFO, MESSAGE, UPDATE, ACK etc. More details please read the RFC3261.
mimeType	The mime type of SIP message.
subMimeType	The sub mime type of SIP message.

Returns:

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

Remarks:

Default, the PortSIP VoIP SDK support these media types(mime types) that 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. "application/media_control+xml" in INFO message.
```

The SDK allows received SIP message that included above mime types. Now if remote side send a INFO SIP message, this message "Content-Type" header value is "text/plain", the SDK will reject this INFO message, because "text/plain" of INFO message does not included in the default support list. Then how to let the SDK receive the SIP INFO message that included "text/plain" mime type? We should use addSupportedMimyType to do it:

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

If want to receive the NOTIFY message with "application/media_control+xml", then:

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

About the mime type details, please visit this website: 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:

sipMessage	The SIP message.
headerName	Which header want to access of 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:

headerName	The custom header name which will be appears in every outgoing SIP message.
headerValue	The custom header value.

Returns:

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

int com.portsip.PortSipSdk.clearAddExtensionHeaders ()

Clear the added extension headers(custom headers)

Returns:

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

Remarks:

Example, we have added two custom headers into every outgoing SIP message and want remove them.

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

if called this function, the added extension headers is 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:

headerName	The SIP header name which will be modify it's value.
headerValue	The heaver value want to modify.

Returns:

If the function succeeds, the return value is 0. If the function fails, the return value is 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, no longer modify every outgoing SIP message header values.

Returns:

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

Remarks:

Example, modified two headers value for every outging SIP message and then clear it:

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

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_audiocodec, 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.setLoudspeakerStatus (boolean useSpeaker)

Detailed Description

Function Documentation

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

Set the video device that will use for video call.

Parameters:

deviceId	Device ID(index) for video device(camera).

Returns:

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

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

Set the video capture resolution.

Parameters:

width	Video resolution, width
height	Video resolution, height

Returns:

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

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

Set the Audio bit rate.

Parameters:

sessionId	The session ID of the call.
enum_audiocodec	Audio codec type.allow : ENUM_AUDIOCODEC_OPUS
bitrateKbps	The Audio bit rate in KBPS.

Returns:

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

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

Set the video bit rate.

Parameters:

bitrateKbps	The video bit rate in KBPS.

Returns:

If the function succeeds, the return value is 0. If the function fails, the return value is 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 set the frame rate, the SDK using default frame rate.

Parameters:

frameRate	The frame rate value, minimum is 5, maximum is 30. The bigger value will give you better video quality but require more bandwidth;

Returns:

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

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

Send the video to remote side.

Parameters:

sessionId	The session ID of the call.
send	Set to true to send the video, false to stop send.

Returns:

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

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

Changing the orientation of the video.

Parameters:

enum_rotation	The video rotation that you want to set, it allows: 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, the return value is 0. If the function fails, the return value is a specific error code.

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

Set the the window that using to display the local video image.

Parameters:

localVideoView	SurfaceView a SurfaceView to display local video image from camera.

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

Set the window for a session that using to display the received remote video image.

Parameters:

sessionId	The session ID of the call.
remoteVideoView	SurfaceView a SurfaceView to display received remote video image.

Returns:

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

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

Start/stop to display the local video image.

Parameters:

state	Set to true to display local video iamge.

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

Enable/disable the NACK feature(rfc6642) which help to improve the video quatliy.

Parameters:

state	Set to true to enable.

Returns:

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

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

Mute the device microphone.

Parameters:

mute	If the value is set to true, the microphone is muted, set to false to un-mute it.

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

Mute the device speaker, it's unavailable for Android and iOS.

Parameters:

mute	If the value is set to true, the speaker is muted, set to false to un-mute it.

int com.portsip.PortSipSdk.getDynamicSpeakerVolumeLevel ()

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 speaker volume by this parameter, the range is 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, the range is 0 - 9.

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

Set the audio device that will use for audio call. For Android and iOS, just allow switch between earphone and Loudspeaker.

Parameters:

useSpeaker	Set to true the SDK use loudspeaker for audio call, this just available for mobile platform only.

Returns:

If the function succeeds, the return value is 0. If the function fails, the return value is 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:

callee	The callee, it can be name only or full SIP URI, for example: user001 or sip:user001@sip.iptel.org or sip:user002@sip.yourdomain.com:5068
sendSdp	If set to false then the outgoing call doesn't include the SDP in INVITE message.
videoCall	If set the true and at least one video codec was added, then the outgoing call include the video codec into SDP. Otherwise no video codec to be added into outgoing SDP.

Returns:

If the function succeeds, the return value is the session ID of the call greater than 0. If the function fails, the return value is 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:

sessionId	The sessionId of the call.
code	Reject code, for example, 486, 480 etc.

Returns:

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

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

hangUp Hang up the call.

Parameters:

sessionId	Session ID of the call.

Returns:

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

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

answerCall Answer the incoming call.

Parameters:

sessionId	The session ID of call.
videoCall	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, the return value is 0. If the function fails, the return value is 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:

sessionId	The session ID of call.
enableAudio	Set to true to allow the audio in update call, false for disable audio in update call.
enableVideo	Set to true to allow the video in update call, false for disable video in update call.

Returns:

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

Remarks:

Example usage:

Example 1: A called B with the audio only, B answered A, there has an audio conversation between A, B. Now A want to see B video, A use these functions to do it.

```
clearVideoCodec();
addVideoCodec(VIDEOCODEC_H264);
updateCall(sessionId, true, true);
```

Example 2: Remove video stream from currently conversation.

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

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

To place a call on hold.

Parameters:

sessionId	The session ID of call.

Returns:

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

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

Take off hold.

Parameters:

sessionId	The session ID of call.

Returns:

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

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

Mute the specified session audio or video.

Parameters:

sessionId	The session ID of the call.
muteIncomingAudio	Set it to true to mute incoming audio stredam, can't hearing remote side audio.
muteOutgoingAudio	Set it to true to mute outgoing audio stredam, the remote side can't hearing audio.
muteIncomingVideo	Set it to true to mute incoming video stredam, can't see remote side video.
muteOutgoingVideo	Set it to true to mute outgoing video stredam, the remote side can't see video.

Returns:

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

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

Forward call to another one when received the incoming call.

Parameters:

sessionId	The session ID of the call.
forwardTo	Target of the forward, it can be "sip:number@sipserver.com" or "number" only.

Returns:

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

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

Send DTMF tone.

Parameters:

sessionId	The session ID of the call.
enum_dtmfMethod	Support send DTMF tone with two methods: DTMF_RFC2833 and DTMF_INFO. The DTMF_RFC2833 is recommend.
code	The DTMF tone, the values is listed below:

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:

dtmfDuration	The DTMF tone samples, recommend 160.
playDtmfTone	Set to true the SDK play local DTMF tone sound during send DTMF.

Returns:

If the function succeeds, the return value is 0. If the function fails, the return value is 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*)

Refer the currently call to another one.

Parameters:

sessionId	The session ID of the call.
referTo	Target of the refer, it can be "sip:number@sipserver.com" or "number" only.

Returns:

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

Remarks:

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

You can watch the video on Youtube at:
https://www.youtube.com/watch?v=_2w9EGgr3FY it will shows how to do the transfer.

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

Make an attended refer.

Parameters:

sessionId	The session ID of the call.
replaceSessionId	Session ID of the replace call.
referTo	Target of the refer, it can be "sip:number@sipserver.com" or "number" only.

Returns:

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

Remarks:

Please read the sample project source code to got more details. Or You can watch the video on Youtube at:
https://www.youtube.com/watch?v=_2w9EGgr3FY
use the Windows Media Player to play the AVI file after it will shows how to do the transfer.

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

Accept the REFER request, a new call will be make if called this function, usuall called after onReceivedRefer callback event.

Parameters:

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

Returns:

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

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

Reject the REFER request.

Parameters:

referId	The ID of REFER request that comes from onReceivedRefer callback event.

Returns:

If the function succeeds, the return value is 0. If the function fails, the return value is 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 to instread of microphone,MUST called this function first

if want to send the PCM stream data to another side.

Parameters:

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

Returns:

If the function succeeds, the return value is 0. If the function fails, the return value is 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 to instead of audio device capture(microphone).

Parameters:

sessionId	Session ID of the call conversation.
data	The PCM audio stream data, must is 16bit, mono.
dataLength	The size of data.

Returns:

If the function succeeds, the return value is 0. If the function fails, the return value is 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 send video stream data to remote side from another source to instread of camera,

MUST called this function first if want to send the video stream data to another side.

Parameters:

sessionId	The session ID of call.
state	Set to true to enable the send stream, false to disable.

Returns:

If the function succeeds, the return value is 0. If the function fails, the return value is 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 to instead of video device capture(camera). </>Before called this funtion,you MUST call the enableSendVideoStreamToRemote function.

Parameters:

sessionId	Session ID of the call conversation.
data	The video video stream data, must is i420 format.
dataLength	The size of data.
width	The video image width.
height	The video image height.

Returns:

If the function succeeds, the return value is 0. If the function fails, the return value is 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 the sending and received RTP packets.

Parameters:

enable	Set to true to enable the RTP callback for received 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:

sessionId	The session ID of call.
enable	Set to true to enable audio stream callback, false to stop the callback.
enum_audioCallbackMode	The audio stream callback mode, allow: 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:

sessionId	The session ID of call.
enum_videoCallbackMode	The video stream callback mode, allow below values 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:

sessionId	The session ID of call.
enable	Set to true to enable video Decoder Info callback, 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 record the call.

Parameters:

sessionId	The session ID of call conversation.
recordFilePath	The file path to save record file, it's must exists.
recordFileName	The file name of record file, for example: audiorecord.wav or videorecord.avi.
appendTimeStamp	Set to true to append the timestamp to the recording file name.
enum_audioFileFormat	The audio record file format, allow below values:
enum_audioRecordMode	The audio record mode, allow below values:
enum_videocodec	The codec which using for compress the video data to save into video record file.
enum_videoRecordMode	Allow set video record mode, support record received video/send video/both received and send.

Returns:

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

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

Stop record.

Parameters:

sessionId	The session ID of call conversation.

Returns:

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

Play audio and video file to remoe 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:

sessionId	Session ID of the call.
aviFile	The file full path name, such as "/mnt/sdcard/test.avi".
loop	Set to false to stop play video file when it is end. Set to true to play it as repeat.
playAudio	If set to true then play audio and video together, set to false just play video only.

Returns:

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

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

Stop play video file to remote side.

Parameters:

sessionId	Session ID of the call.

Returns:

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

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

Play an wave file to remote party.

Parameters:

sessionId	Session ID of the call.
filename	The file full path name, such as "/mnt/sdcard/test.wav".

fileSamplesPerSec	The wave file sample in seconds, should be 8000 or 16000 or 32000.
loop	Set to false to stop play audio file when it is end. Set to true to play it as repeat.

Returns:

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

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

Stop play wave file to remote side.

Parameters:

sessionId	Session ID of the call.

Returns:

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

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

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

Parameters:

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

Returns:

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

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

Stop play an wave file to remote party as conversation background sound.

Parameters:

sessionId	Session ID of the call.

Returns:

If the function succeeds, the return value is 0. If the function fails, the return value is 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's failures if the exists conference isn't destroy yet.

Parameters:

conferenceVideoWindow	SurfaceView The window which using to display the conference video.
videoWidth	width of conference video resolution
videoHeight	height of conference video resolution
displayLocalVideoInConference	displayLocalVideoInConference

Returns:

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

void com.portsip.PortSipSdk.destroyConference ()

Destroy the exist conference.

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

Set the window for a conference that using to display the received remote video image.

Parameters:

conferenceVideoWindow	SurfaceView The window which using to display the conference video

Returns:

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

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

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

Parameters:

sessionId	Session ID of the call.

Returns:

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

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

Remove a session from an exist conference.

Parameters:

sessionId	Session ID of the call.

Returns:

If the function succeeds, the return value is 0. If the function fails, the return value is 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 the RFC3556.

Parameters:

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

Returns:

If the function succeeds, the return value is 0. If the function fails, the return value is 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:

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

Returns:

If the function succeeds, the return value is 0. If the function fails, the return value is 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:

enable	Set to true to enable audio QoS.
DSCPValue	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.
priority	The 802.1p priority(PCP) field in a 802.1Q/VLAN tag. Values 0-7 set the priority, value -1 leaves the priority setting unchanged.

Returns:

If the function succeeds, the return value is 0. If the function fails, the return value is 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:

enable	Set as true to enable QoS, false to disable.
DSCPValue	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, the return value is 0. If the function fails, the return value is a specific error code.

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

Set the MTU size for video RTP packet.

Parameters:

mtu	Set MTU value, allow value is (512-65507), other value will set to the default: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:

sessionId	The session ID of call conversation.
statistics	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, the return value is 0. If the function fails, the return value is a specific error code.

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

Obtain the RTP statisics of audio channel.

Parameters:

sessionId	The session ID of call conversation.
statistics	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, the return value is 0. If the function fails, the return value is a specific error code.

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

Obtain the RTCP statisics of audio channel.

Parameters:

sessionId	The session ID of call conversation.
statistics	Return audio 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. statistics[4] - Fraction of sent lost in percent. statistics[5] - The number of sent cumulative lost packet. statistics[6] - Fraction of received lost in percent. statistics[7] - The number of received cumulative lost packets.

Returns:

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

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

Obtain the RTCP statisics of audio channel.

Parameters:

sessionId	The session ID of call conversation.
statistics	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, the return value is 0. If the function fails, the return value is 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:

state	Set to true to enable VAD, false to disable.

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

Enable/disable AEC (Acoustic Echo Cancellation).

Parameters:

enum_aecMode	

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

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

Enable/disable Comfort Noise Generator(CNG).

Parameters:

state	Set to true to enable CNG, false to disable.

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

Enable/disable Automatic Gain Control(AGC).

Parameters:

enum_agcMode

Mode	Description
AGC_NONE	0 - Disable AGC.
AGC_DEFAULT	1 - Platform 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:

enum_nsMode

Mode	Description
NS_NONE	0 - Disable NS.
NS_DEFAULT	1 - Platform default.
NS_Conference	2 - Conferencing 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:

to	The receiver of OPTIONS message.
sdp	The SDP of OPTIONS message, it's optional if don't want send the SDP with OPTIONS message.

Returns:

If the function succeeds, the return value is 0. If the function fails, the return value is 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:

sessionId	The session ID of call.
mimeType	The mime type of INFO message.
subMimeType	The sub mime type of INFO message.
infoContents	The contents that send with INFO message.

Returns:

If the function succeeds, the return value is 0. If the function fails, the return value is 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:

sessionId	The session ID of call.
mimeType	The mime type of MESSAGE message.
subMimeType	The sub mime type of MESSAGE message.
message	The contents which send with MESSAGE message, allow binary data.
messageLength	The message size.

Returns:

If the function succeeds, the return value is a message ID allows track the message send state in onSendMessageSuccess and .
onSendMessageFailure. If the function fails, the return value is a specific error code 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 send.

```
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:

to	The message receiver. Likes sip:receiver@portsip.com
mimeType	The mime type of MESSAGE message.
subMimeType	The sub mime type of MESSAGE message.
message	The contents which send with MESSAGE message, allow binary data.
messageLength	The message size.

Returns:

If the function succeeds, the return value is a message ID allows track the message send state in onSendOutOfMessageSuccess and
onSendOutOfMessageFailure. If the function fails, the return value is a specific error code 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 send.

```
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:

contact	The target contact, it must likes sip:contact001@sip.portsip.com.
subject	This subject text will be insert into the SUBSCRIBE message. For example: "Hello, I'm Jason". The subject maybe is UTF8 format, you should use UTF8 to decode it.

Returns:

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

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

Reject a presence SUBSCRIBE request which received from contact.

Parameters:

subscribeId	Subscribe id, when received a SUBSCRIBE request from contact, the event onPresenceRecvSubscribe will be triggered, the event inclues the subscribe id.

Returns:

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

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

Accept the presence SUBSCRIBE request which received from contact.

Parameters:

subscribeId	Subscribe id, when received a SUBSCRIBE request from contact, the event onPresenceRecvSubscribe will be triggered,<bt> the event inclues the subscribe id.

Returns:

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

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

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

Parameters:

subscribeId	Subscribe id, when received a SUBSCRIBE request from contact, the event onPresenceRecvSubscribe will be triggered, the event inclues the subscribe id.
statusText	The state text of presende online, for example: "I'm here"

Returns:

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

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

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

Parameters:

subscribeId	Subscribe id, when received a SUBSCRIBE request from contact, the event onPresenceRecvSubscribe will be triggered, the event inclues the subscribe id.

Returns:

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

Device Manage 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:

The return value is number of recording devices,. If the function fails, the return value is a specific error code less than 0.

int com.portsip.PortSipSdk.getNumOfPlayoutDevices ()

Gets the number of audio devices available for audio playout.

Returns:

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

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

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

Parameters:

index	Device index (0, 1, 2, ..., N-1), where N is given by getNumOfRecordingDevices (). Also -1 is a valid value and 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:

index	Device index (0, 1, 2, ..., N-1), where N is given by getNumOfPlayoutDevices (). Also -1 is a valid value and 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 level.

Parameters:

volume	Volume level of speaker, valid range is 0 - 255.

Returns:

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

int com.portsip.PortSipSdk.getSpeakerVolume ()

Gets the speaker volume level.

Returns:

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

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

Mutes the speaker device completely in the OS.

Parameters:

mute	If set to true, the device output is muted. If set to false, the output is unmuted.

Returns:

If the function succeeds, the return value is 0. If the function fails, the return value is 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 level.

Parameters:

volume	The microphone volume level, the valid value is 0 - 255.

Returns:

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

int com.portsip.PortSipSdk.getMicVolume ()

Retrieves the current microphone volume.

Returns:

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

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

Mute the microphone input device completely in the OS.

Parameters:

mute	If set to true, the input device is muted. Set to false is unmuted.

Returns:

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

boolean com.portsip.PortSipSdk.getSystemInputMute ()

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

Returns:

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

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

Use to do the audio device loop back test.

Parameters:

enable	Set to true start audio look back test; Set to fase to stop.

int com.portsip.PortSipSdk.getNumOfVideoCaptureDevices ()

Gets the number of available capture devices.

Returns:

The return value is number of video capture devices, if fails the return value is a specific error code less than 0.

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

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

Parameters:

index	Device index (0, 1, 2, ..., N-1), where N is given by getNumOfVideoCaptureDevices (). Also -1 is a valid value and will return the name of the default capture device.

Returns:

the name of a specific video capture 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_RESULUTION_NONE** = 0
- static final int **ENUM_RESULUTION_QCIF** = 1
- static final int **ENUM_RESULUTION_CIF** = 2
- static final int **ENUM_RESULUTION_VGA** = 3
- static final int **ENUM_RESULUTION_SVGA** = 4
- static final int **ENUM_RESULUTION_XVGA** = 5
- static final int **ENUM_RESULUTION_720P** = 6
- static final int **ENUM_RESULUTION_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) most earpiece use.
- static final int ENUM_EC_AECM_3 = 6

  Mobile platform(iOS,Android) Loud earpiece or quiet speakerphone use.
- static final int ENUM_EC_AECM_4 = 7

  Mobile platform(iOS,Android) most speakerphone use (Mobile Platform default)
- static final int ENUM_AGC_NONE = 0

  Mobile platform(iOS,Android) Loud 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 only in startRecord

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

used only in startRecord

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) Loud 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 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_srtppolicy)
- 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_audiocodec, 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 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.

Version:

11.2.1

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