PortSIP VoIP SDK Manual for Android

Version 16.7
6/8/20

Table of Contents

Welcome to the PortSIP VoIP SDK

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

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

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

Getting Started

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

Contents

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

SDK User Manual

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

Website

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

Support

Please send email to our Support team if you need any help.

Installation Prerequisites

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

Frequently Asked Questions

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

```
All sample projects of the PortSIP VoIP SDK can be found and downloaded at:
      https://www.portsip.com/download-portsip-voip-sdk/
      https://www.PortSIP.com/portsip-voip-sdk.
```

2. How can I compile the sample project?

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

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

```
  1. Download the sample project and evaluation SDK and extract it to a specified directory
  2. Run Eclipse and create a new Android Application Project
  3. Copy all files form libs directory under extracted directory to the libs directory of your new
application.
  4. Import the dependent class form the SDK. For example:
         import com.portsip.OnPortSIPEvent;
         import com.portsip.PortSipSdk;
  5. Inherit the interface OnPortSIPEvent to process the callback events.
  6. Initialize SDK. For example:
         mPortSIPSDK = new PortSipSdk();
         mPortSIPSDK.setOnPortSIPEvent(instanceofOnPortSIPEvent);
         mPortSIPSDK.CreateCallManager(context);
         mPortSIPSDK.initialize(...);
  For more details please refer to the Sample project source code.
```

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

```
  1. Download and extract the SDK sample project ZIP file into local. Compile and run the "P2PSample"
project.

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

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

  4. Click the "Initialize" button on A and B. If the default port 5060 is already in use, the P2PSample
will prompt "Initialize failure". In case of this,please click the "Uninitialize" button and change
the local port, and click the "Initialize" button again.

  5. The log box will appear "Initialized" if the SDK is successfully initialized.

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

```
Note: If the local sip port is changed to other port, for example, A is using local port5080, and B
is using local port 6021, to make call from A to B, please enter"sip:222@192.168.1.11:6021" and dial;
while to make call from B to A, enter"sip:111@192.168.1.10:5080".
```

5. Is the SDK thread safe?

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

```
the "onAudioRawCallback", "onVideoRawCallback", "onReceivedRtpPacket", "onSendingRtpPacket"
callbacks.
```

Module Index

Modules

Here is a list of all modules:

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

Detailed Description

Register events

Function Documentation

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

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

Parameters:

reason	The status text.
code	The status code.
sipMessage	The SIP message received.

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

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

Parameters:

reason	The status text.
code	The status code.
sipMessage	The SIP message received.

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, String sipMessage)
- void com.portsip.OnPortSIPEvent.onInviteTrying (long sessionId)
- void com.portsip.OnPortSIPEvent.onInviteSessionProgress (long sessionId, String audioCodecs, String videoCodecs, boolean existsEarlyMedia, boolean existsAudio, boolean existsVideo, String sipMessage)
- void com.portsip.OnPortSIPEvent.onInviteRinging (long sessionId, String statusText, int statusCode, String sipMessage)
- void com.portsip.OnPortSIPEvent.onInviteAnswered (long sessionId, String callerDisplayName, String caller, String calleeDisplayName, String callee, String audioCodecs, String videoCodecs, boolean existsAudio, boolean existsVideo, String sipMessage)
- void com.portsip.OnPortSIPEvent.onInviteFailure (long sessionId, String reason, int code, String sipMessage)
- void com.portsip.OnPortSIPEvent.onInviteUpdated (long sessionId, String audioCodecs, String videoCodecs, boolean existsAudio, boolean existsVideo, String sipMessage)
- 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.onDialogStateUpdated (String BLFMonitoredUri, String BLFDialogState, String BLFDialogId, String BLFDialogDirection)
- 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*, String *sipMessage*)

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

Parameters:

sessionId	The session ID of the call.
callerDisplayName	The display name of caller
caller	The caller.
calleeDisplayNam	The display name of callee.

e	
callee	The callee.
audioCodecs	The matched audio codecs. It's separated by "#" if there are more than one codecs.
videoCodecs	The matched video codecs. It's separated by "#" if there are more than one codecs.
existsAudio	By setting to true, it means that this call include the audio.
existsVideo	By setting to true, it means that this call include the video.
sipMessage	The SIP message received.

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

If the outgoing call is being processed, this event will be 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*, String *sipMessage*)

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

Parameters:

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

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

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

Parameters:

sessionId	The session ID of the call.
statusText	The status text.
statusCode	The status code.
sipMessage	The SIP message received.

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

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

Parameters:

sessionId	The session ID of the call.
callerDisplayName	The display name of caller
caller	The caller.
calleeDisplayNam	The display name of callee.

e	
callee	The callee.
audioCodecs	The matched audio codecs. It's separated by "#" if there are more than one codecs.
videoCodecs	The matched video codecs. It's separated by "#" if there are more than one codecs.
existsAudio	By setting to true, this call includes the audio.
existsVideo	By setting to true, this call includes the video.
sipMessage	The SIP message received.

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

This event will be triggered if the outgoing call fails.

Parameters:

sessionId	The session ID of the call.
reason	The failure reason.
code	The failure code.
sipMessage	The SIP message received.

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

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

Parameters:

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

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

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

Parameters:

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 this event will be triggered.

Parameters:

forwardTo	The target SIP URI of the call forwarding.

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

This event is triggered once remote side ends the call.

Parameters:

sessionId	The session ID of the call.

void com.portsip.OnPortSIPEvent.onDialogStateUpdated (String *BLFMonitoredUri*, String *BLFDialogState*, String *BLFDialogId*, String *BLFDialogDirection*)

If a user subscribed and his dialog status monitored, when the monitored user is holding a call or is being rang, this event will be triggered.

Parameters:

BLFMonitoredUri	the monitored user's URI
BLFDialogState	- the status of the call
BLFDialogId	- the id of the call
BLFDialogDirection	- the direction of the call

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

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

Parameters:

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 un-holds the call, this event will be triggered

Parameters:

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

Refer events

Functions

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

Detailed Description

Function Documentation

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

This event will be triggered once received a REFER message.

Parameters:

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 calls "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 calls "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 being processed, this event will be triggered.

Parameters:

sessionId	The session ID of the call.

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

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

Parameters:

sessionId	The session ID of the call.

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

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

Parameters:

sessionId	The session ID of the call.

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

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

Parameters:

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 receiving a SIP message. This event is disabled by default. To enable, use enableCallbackSignaling.

Parameters:

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

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

This event will be triggered when sent a SIP message. This event is disabled by default. To enable, use enableCallbackSignaling.

Parameters:

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

MWI events

Functions

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

Detailed Description

Function Documentation

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

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

Parameters:

messageAccount	Voice message account
urgentNewMessageCount	Count of new urgent messages.
urgentOldMessageCount	Count of history urgent message.
newMessageCount	Count of new messages.
oldMessageCount	Count of history messages.

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

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

Parameters:

messageAccount	Fax message account
urgentNewMessageCount	Count of new urgent messages.
urgentOldMessageCount	Count of history urgent messages.
newMessageCount	Count of new messages.
oldMessageCount	Count of old messages.

DTMF events

Functions

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

Detailed Description

Function Documentation

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

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

Parameters:

sessionId	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)
- void com.portsip.OnPortSIPEvent.onRecvNotifyOfSubscription (long subscribeId, String notifyMessage, byte[] messageData, int messageDataLength)

Detailed Description

Function Documentation

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

This event will be triggered when receiving 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 receiving the INFO message.

Parameters:

infoMessage	The whole INFO message received in text format.

void com.portsip.OnPortSIPEvent.onRecvNotifyOfSubscription (long *subscribeId*, String *notifyMessage*, byte[] *messageData*, int *messageDataLength*)

This event will be triggered when receiving a NOTIFY message of the subscription.

Parameters:

subscribeId	The ID of SUBSCRIBE request.
notifyMessage	The received INFO message in text format.
messageData	The received message body. It's can be either text or binary data.
messageDataLength	The length of "messageData".

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, String sipMessage)
- 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)
- void com.portsip.OnPortSIPEvent.onSubscriptionFailure (long subscribeId, int statusCode)
- void com.portsip.OnPortSIPEvent.onSubscriptionTerminated (long subscribeId)

Detailed Description

Function Documentation

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

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

Parameters:

subscribeId	The ID of SUBSCRIBE request.
fromDisplayName	The display name of contact.
from	The contact who sends 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 changes presence status, this event will be triggered.

Parameters:

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

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

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

Parameters:

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

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

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

Parameters:

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

This event will be triggered when receiving 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 can be text or binary data. Use the mimeType and subMimeType to differentiate them. For example, if the mimeType is "text" and subMimeType is "plain", "messageData" is text message body. If the mimeType is "application" and subMimeType is "vnd.3gpp.sms", "messageData" is binary message body.
messageDataLength	The length of "messageData".
sipMessage	The SIP message received.

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

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

Parameters:

sessionId	The session ID of the call.
messageId	The message ID. It's equal to the return value of sendMessage function.

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

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

Parameters:

sessionId	The session ID of the call.
messageId	The message ID. It's equal to 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 is sent successfully out of dialog, this event will be triggered.

Parameters:

messageId	The message ID. It's equal to 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 failed to be sent out of dialog, this event would be triggered.

Parameters:

messageId	The message ID. It's equal to 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.

void com.portsip.OnPortSIPEvent.onSubscriptionFailure (long *subscribeId*, int *statusCode*)

This event will be triggered on sending SUBSCRIBE failure.

Parameters:

subscribeId	The ID of SUBSCRIBE request.
statusCode	The status code.

void com.portsip.OnPortSIPEvent.onSubscriptionTerminated (long *subscribeId*)

This event will be triggered when a SUBSCRIPTION is terminated or expired.

Parameters:

subscribeId	The ID of SUBSCRIBE request.

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)

Detailed Description

Function Documentation

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

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

Parameters:

sessionId	The session ID of the call.
isAudio	If the received RTP packet is of audio, this parameter returns 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 is time-consuming, you should post a message to another thread and execute SDK API functions or other code in another thread.

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

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

Parameters:

sessionId	The session ID of the call.
isAudio	If the received RTP packet is of audio, this parameter returns 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 is time-consuming, you should post a message to another thread and execute SDK API functions or other code in another thread.

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

This event will be triggered once receiving the audio packets if called enableAudioStreamCallback function.

Parameters:

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

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

See also:

PortSipSdk::enableAudioStreamCallback

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

This event will be triggered once receiving the video packets if enableVideoStreamCallback function is called.

Parameters:

sessionId	The session ID of the call.
enum_videoCallbackMode	The type which is passed in enableVideoStreamCallback function. Below types allowed: 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 in YUV420 format, YV12.
dataLength	The data size.

See also:

PortSipSdk::enableVideoStreamCallback

SDK functions

Modules

- Initialize and register functions
- Audio and video codecs functions
- Additional settings functions
- Access SIP message header functions
- Audio and video functions
- Call functions

- Refer functions
- Send audio and video stream functions
- RTP packets, Audio stream and video stream callback
- Record functions
- Play audio and video file to remote functions
- Conference functions
- RTP and RTCP QOS functions
- RTP statistics functions
- Audio effect functions
- Send OPTIONS/INFO/MESSAGE functions

Detailed Description

Initialize and register functions

Functions

- void com.portsip.PortSipSdk.CreateCallManager (Context context)
- void **com.portsip.PortSipSdk.setAudioManagerEvents** (AppRTCAudioManager.AudioManagerEvents audioManagerEvents)
- PortSipEnumDefine.AudioDevice **com.portsip.PortSipSdk.getSelectedAudioDevice** ()
- void com.portsip.PortSipSdk.DeleteCallManager ()
- int com.portsip.PortSipSdk.initialize (int enum_transport, String localIP, int localSIPPort, int enum_LogLevel, String LogPath, int maxLines, String agent, int audioDeviceLayer, int videoDeviceLayer, String TLSCertificatesRootPath, String TLSCipherList, boolean verifyTLSCertificate, String dnsServers)
- int com.portsip.PortSipSdk.setInstanceId (String instanceId)
- int com.portsip.PortSipSdk.setUser (String userName, String displayName, String authName, String password, String userDomain, String SIPServer, int SIPServerPort, String STUNServer, int STUNServerPort, String outboundServer, int outboundServerPort)
- void com.portsip.PortSipSdk.removeUser ()
  remove user account info.
- int com.portsip.PortSipSdk.registerServer (int expires, int retryTimes)
- int com.portsip.PortSipSdk.refreshRegistration (int expires)
- int com.portsip.PortSipSdk.unRegisterServer ()
- int com.portsip.PortSipSdk.setDisplayName (String displayName)
- int com.portsip.PortSipSdk.setLicenseKey (String key)

Detailed Description

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 callback Handlers.

int com.portsip.PortSipSdk.initialize (int *enum_transport*, String *localIP*, int *localSIPPort*, int *enum_LogLevel*, String *LogPath*, int *maxLines*, String *agent*, int *audioDeviceLayer*, int *videoDeviceLayer*, String *TLSCertificatesRootPath*, String *TLSCipherList*, boolean *verifyTLSCertificate*, String *dnsServers*)

Initialize the SDK.

Parameters:

enum_transport	Transport for SIP signaling, which can be set as: ENUM_TRANSPORT_UDP, ENUM_TRANSPORT_TCP, ENUM_TRANSPORT_TLS, ENUM_TRANSPORT_PERS. The ENUM_TRANSPORT_PERS_UDP is the private PortSIP transport for anti-SIP blocking, which must work with the PERS. ENUM_TRANSPORT_PERS. The ENUM_TRANSPORT_PERS_TCP is the private PortSIP transport for anti-SIP blocking, which must work with the PERS.
localIP	The local PC IP address (for example: 192.168.1.108). It will be used for sending and receiving SIP messages and RTP packets. If the local IP is provided in IPv6 format, the SDK will use IPv6. If you want the SDK to choose correct network interface (IP) automatically, please use "0.0.0.0" for IPv4, or "::" for IPv6.
localSIPPort	The listening port for SIP message transmission, for example 5060.
enum_LogLevel	Set the application log level. The SDK will generate "PortSIP_Log_datatime.log" file if the log is enabled. ENUM_LOG_LEVEL_NONE ENUM_LOG_LEVEL_DEBUG ENUM_LOG_LEVEL_ERROR ENUM_LOG_LEVEL_WARNING ENUM_LOG_LEVEL_INFO ENUM_LOG_LEVEL_DEBUG
LogPath	The path for storing log file. The path (folder) specified MUST be existent.
maxLines	Theoretically, unlimited count of lines are supported depending on the device capability. For SIP client, it is recommended to limit it as ranging 1 - 100.
agent	The User-Agent header to be inserted in to SIP messages.
audioDeviceLayer	Specifies the audio device layer that should be using: 0 = Use the OS defaulted device. 1 = Virtual device, usually use this for the device that has no sound device installed.
videoDeviceLayer	Specifies the video device layer that should be using: 0 = Use the OS defaulted device. 1 = Use Virtual device, usually use this for the device that has no camera installed.
TLSCertificatesRootPath	Specify the TLS certificate path, from which the SDK will load the certificates automatically. Note: On Windows, this path will be ignored, and SDK will read the certificates from Windows certificates stored area instead.
TLSCipherList	Specify the TLS cipher list. This parameter is usually passed as empty so that the SDK will offer all available ciphers.
verifyTLSCertificate	Indicate if SDK will verify the TLS certificate or not. By setting to false, the SDK will not verify the validity of TLS certificate.
dnsServers	Additional Nameservers DNS servers. Value null indicates system DNS Server. Multiple servers will be split by ";", e.g "8.8.8.8;8.8.4.4"

Returns:

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

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

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

Parameters:

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

Returns:

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

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

Set user account info.

Parameters:

userName	Account (username) of the SIP, usually provided by an IP-Telephony service provider.
displayName	The name displayed. You can set it as your like, such as "James Kend". It's optional.
authName	Authorization user name (usually equal to the username).
password	User's password. It's optional.
userDomain	User domain; this parameter is optional, which allows to transfer an empty string if you are not using the 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 for NAT traversal. It's optional and can be used to transfer 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 and allows to transfer an empty string if not using the outbound server.
outboundServerPort	Outbound proxy server port, it will be ignored if the outboundServer is empty.

Returns:

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

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

Register to SIP proxy server (login to server)

Parameters:

expires	Time interval for registration refreshment, in seconds. The maximum of supported value is 3600. It will be inserted into SIP REGISTER message headers.
retryTimes	The maximum of retry attempts if failed to refresh the registration. By setting to <= 0, the attempt will be disabled and onRegisterFailure callback will be triggered when facing retry failure.

Returns:

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

If the registration to server succeeds, onRegisterSuccess will be triggered; otherwise onRegisterFailure will be triggered.

int com.portsip.PortSipSdk.refreshRegistration (int *expires*)

Refresh the registration manually after successfully registered.

Parameters:

expires	Time interval for registration refreshment, in seconds. The maximum of supported value is 3600. It will be inserted into SIP REGISTER message headers.

Returns:

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

int com.portsip.PortSipSdk.unRegisterServer ()

Un-register from the SIP proxy server.

Returns:

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

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

Set the display name of user.

Parameters:

displayName	That will appear in the From/To Header.

Returns:

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

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

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

Parameters:

key	The SDK license key. Please purchase from PortSIP.

Returns:

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

Audio and video codecs functions

Functions

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

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

Detailed Description

Function Documentation

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

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

Parameters:

enum_audiocodec	Audio codec type, including: ENUM_AUDIOCODEC_G729, ENUM_AUDIOCODEC_PCMA, ENUM_AUDIOCODEC_PCMU, ENUM_AUDIOCODEC_GSM, ENUM_AUDIOCODEC_G722, ENUM_AUDIOCODEC_ILBC, ENUM_AUDIOCODEC_AMR, ENUM_AUDIOCODEC_AMRWB, ENUM_AUDIOCODEC_SPEEX, ENUM_AUDIOCODEC_SPEEXWB, ENUM_AUDIOCODEC_ISACWB, ENUM_AUDIOCODEC_ISACSWB, ENUM_AUDIOCODEC_OPUS, ENUM_AUDIOCODEC_DTMF.

Returns:

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

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

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

Parameters:

enum_videocodec	Video codec type. Supported types include ENUM_VIDEOCODEC_H264, ENUM_VIDEOCODEC_VP8. ENUM_VIDEOCODEC_VP9.

Returns:

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

boolean com.portsip.PortSipSdk.isAudioCodecEmpty ()

Detect if the audio codecs are enabled.

Returns:

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

boolean com.portsip.PortSipSdk.isVideoCodecEmpty ()

Detect if the video codecs are enabled.

Returns:

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

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

Set the RTP payload type for dynamic audio codec.

Parameters:

enum_audiocodec	Audio codec types. Supported types include: ENUM_AUDIOCODEC_G729,

	ENUM_AUDIOCODEC_PCMA, ENUM_AUDIOCODEC_PCMU, ENUM_AUDIOCODEC_GSM, ENUM_AUDIOCODEC_G722, ENUM_AUDIOCODEC_ILBC, ENUM_AUDIOCODEC_AMR, ENUM_AUDIOCODEC_AMRWB, ENUM_AUDIOCODEC_SPEEX, ENUM_AUDIOCODEC_SPEEXWB, ENUM_AUDIOCODEC_ISACWB, ENUM_AUDIOCODEC_ISACSWB, ENUM_AUDIOCODEC_OPUS, ENUM_AUDIOCODEC_DTMF
payloadType	The new RTP payload type that you want to set.

Returns:

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

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

Set the RTP payload type for dynamic video codec.

Parameters:

enum_videocodec	Video codec type. Supported types include: ENUM_VIDEOCODEC_H264, ENUM_VIDEOCODEC_VP8. ENUM_VIDEOCODEC_VP9.
payloadType	The new RTP payload type that you want to set.

Returns:

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

void com.portsip.PortSipSdk.clearAudioCodec ()

Remove all the enabled audio codecs.

void com.portsip.PortSipSdk.clearVideoCodec ()

Remove all the enabled video codecs.

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

Set the codec parameter for audio codec.

Parameters:

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

Returns:

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

See also:

PortSipEnumDefine

Remarks:

Example:

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

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

Set the codec parameter for video codec.

Parameters:

enum_videocodec	Video codec types. Supported types include: ENUM_VIDEOCODEC_H264, ENUM_VIDEOCODEC_VP8. ENUM_VIDEOCODEC_VP9.
sdpParameter	The parameter is in string format.

Returns:

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

Remarks:

Example:

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

Additional settings functions

Functions

- String com.portsip.PortSipSdk.getVersion ()
- int com.portsip.PortSipSdk.enableRport (boolean enable)
- int com.portsip.PortSipSdk.enableEarlyMedia (boolean enable)
  Enable/disable rport(RFC3581).
- int com.portsip.PortSipSdk.enableReliableProvisional (boolean enable)
- int com.portsip.PortSipSdk.enable3GppTags (boolean enable)
- void com.portsip.PortSipSdk.enableCallbackSignaling (boolean enableSending, boolean enableReceived)
- 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 state)
- void com.portsip.PortSipSdk.enableAutoCheckMwi (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

String com.portsip.PortSipSdk.getVersion ()

Get the version number of the current SDK.

Returns:

String with version description

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

Enable/Disable rport(RFC3581).

Parameters:

enable	enable Set to true to enable the SDK to support rport. By default it is enabled.

Returns:

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

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

Enable/disable rport(RFC3581).

Parameters:

enable	Set to true to enable the SDK to support rport. By default it is enabled.

Returns:

If the function succeeds, it will return value 0. If the function fails, it will return a specific error code. Enable/Disable Early Media.

Parameters:

enable	Set to true to enable the SDK support Early Media. By default the Early Media is disabled.

Returns:

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

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

Enable/Disable PRACK.

Parameters:

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

Returns:

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

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

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

Parameters:

enable	Set to true to enable 3Gpp tags for SDK.

Returns:

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

void com.portsip.PortSipSdk.enableCallbackSignaling (boolean *enableSending*, boolean *enableReceived*)

Enable/disable the callback of the SIP messages.

Parameters:

enableSending	Set as true to enable to callback the sent SIP messages, or false to disable. Once enabled, the "onSendingSignaling" event will be triggered when the SDK sends a SIP message.
enableReceived	Set as true to enable to callback the received SIP messages, or false to disable. Once enabled, the "onReceivedSignaling" event will be triggered when the SDK receives 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 to set the RTP port 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, it will return value 0. If the function fails, it will return a specific error code.

Remarks:

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

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

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

Parameters:

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

Remarks:

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

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

Enable call forwarding.

Parameters:

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

Returns:

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

int com.portsip.PortSipSdk.disableCallForward ()

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

Returns:

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

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

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

Parameters:

timerSeconds	The value of the refresh interval in seconds. A minimum of 90 seconds required.

Returns:

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

Remarks:

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

void com.portsip.PortSipSdk.disableSessionTimer ()

Disable the session timer.

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

Enable/disable the "Do not disturb" status.

Parameters:

state	If it is set to true, the SDK will reject all incoming calls.

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

Enable/disable the "Auto Check MWI" status.

Parameters:

state	If it is set to true, the SDK will check Mwi automatically.

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

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

Parameters:

state	When it's set to true, it's allowed to send the keep-alive packet during the conversation;
keepAlivePayload Type	The payload type of the keep-alive RTP packet. It's usually set to 126.
deltaTransmitTime MS	The interval for sending keep-alive RTP packet, in millisecond. Recommended value ranges 15000 - 300000.

Returns:

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

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

Enable or disable to send SIP keep-alive packet.

Parameters:

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

Returns:

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

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

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

Parameters:

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

Returns:

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

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

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

Parameters:

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

Returns:

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

Remarks:

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

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

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

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

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

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

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

Access SIP message header functions

Functions

- String com.portsip.PortSipSdk.getSipMessageHeaderValue (String sipMessage, String headerName)
- long com.portsip.PortSipSdk.addSipMessageHeader (long sessionId, String methodName, int msgType, String headerName, String headerValue)
- int com.portsip.PortSipSdk.removeAddedSipMessageHeader (long addedSipMessageId)
- void com.portsip.PortSipSdk.clearAddedSipMessageHeaders ()
- long com.portsip.PortSipSdk.modifySipMessageHeader (long sessionId, String methodName, int msgType, String headerName, String headerValue)
- int com.portsip.PortSipSdk.removeModifiedSipMessageHeader (long modifiedSipMessageId)
- void com.portsip.PortSipSdk.clearModifiedSipMessageHeaders ()

Detailed Description

Function Documentation

String com.portsip.PortSipSdk.getSipMessageHeaderValue (String *sipMessage*, String *headerName*)

Access the SIP header of SIP message.

Parameters:

sipMessage	The SIP message.
headerName	The header of which user wishes to access the SIP message.

Returns:

String. The SIP header of SIP message.

long com.portsip.PortSipSdk.addSipMessageHeader (long *sessionId*, String *methodName*, int *msgType*, String *headerName*, String *headerValue*)

Add the SIP Message header into the specified outgoing SIP message.

Parameters:

sessionId	Add the header to the SIP message with the specified session Id only. By setting to -1, it will be added to all messages.
methodName	Add the header to the SIP message with specified method name only. For example: "INVITE", "REGISTER", "INFO" etc. If "ALL" specified, it will add all SIP messages.
msgType	1 refers to apply to the request message, 2 refers to apply to the response message, 3 refers to apply to both request and response.
headerName	The header name which will appear in SIP message.
headerValue	The custom header value.

Returns:

If the function succeeds, it will return the addedSipMessageId , which is greater than 0. If the function fails, it will return a specific error code.

int com.portsip.PortSipSdk.removeAddedSipMessageHeader (long *addedSipMessageId*)

Remove the headers (custom header) added by addSipMessageHeader.

Parameters:

addedSipMessageId	The addedSipMessageId return by addSipMessageHeader.

Returns:

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

void com.portsip.PortSipSdk.clearAddedSipMessageHeaders ()

Clear the added extension headers (custom headers)

Remarks:

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

```
addSipMessageHeader(-1,"ALL",3,"Blling", "usd100.00");
    addSipMessageHeader(-1,"ALL",3,"ServiceId", "8873456");
    clearAddedSipMessageHeaders();
```

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

long com.portsip.PortSipSdk.modifySipMessageHeader (long *sessionId*, String *methodName*, int *msgType*, String *headerName*, String *headerValue*)

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

Parameters:

sessionId	The header to the SIP message with the specified session Id. By setting to -1, it will be added to all messages.
methodName	Modify the header to the SIP message with specified method name only. For example: "INVITE", "REGISTER", "INFO" etc. If "ALL" specified, it will add all SIP messages.
msgType	1 refers to apply to the request message, 2 refers to apply to the response message, 3 refers to apply to both request and response.
headerName	The SIP header name of which the value will be modified.
headerValue	The heaver value to be modified.

Returns:

If the function succeeds, it will return modifiedSipMessageId, which is greater than 0. If the function fails, it will return a specific error code.

Remarks:

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

```
modifySipMessageHeader(-1,"ALL",3, "Expires", "1000");
modifySipMessageHeader(-1,"ALL",3, "User-Agent", "MyTest Softphone 1.0");
```

int com.portsip.PortSipSdk.removeModifiedSipMessageHeader (long *modifiedSipMessageId*)

Remove the headers (custom header) added by modifiedSipMessageId.

Parameters:

modifiedSipMessageId	The modifiedSipMessageId return by modifySipMessageHeader.

Returns:

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

void com.portsip.PortSipSdk.clearModifiedSipMessageHeaders ()

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

Remarks:

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

```
modifySipMessageHeader(-1,"ALL",3, "Expires", "1000");
modifySipMessageHeader(-1,"ALL",3, "User-Agent", "MyTest Softphone 1.0");
cleaModifyHeaders();
```

Audio and video functions

Functions

- int com.portsip.PortSipSdk.setVideoDeviceId (int deviceId)
- int com.portsip.PortSipSdk.enableVideoHardwareCodec (boolean enableHWEncoder, boolean enableHWDecoder)
- int com.portsip.PortSipSdk.setVideoResolution (int width, int height)
- int com.portsip.PortSipSdk.setVideoCropAndScale (boolean enable)
- int com.portsip.PortSipSdk.setAudioBitrate (long sessionId, int enum_audiocodec, int bitrateKbps)
- int com.portsip.PortSipSdk.setVideoBitrate (long sessionId, int bitrateKbps)
- int com.portsip.PortSipSdk.setVideoFrameRate (long sessionId, int frameRate)
- int com.portsip.PortSipSdk.sendVideo (long sessionId, boolean send)
- void com.portsip.PortSipSdk.setLocalVideoWindow (PortSIPVideoRenderer renderer)
- int com.portsip.PortSipSdk.setRemoteVideoWindow (long sessionId, PortSIPVideoRenderer renderer)
- void com.portsip.PortSipSdk.displayLocalVideo (boolean state, boolean mirror)
- int com.portsip.PortSipSdk.setVideoNackStatus (boolean state)
- int com.portsip.PortSipSdk.setChannelOutputVolumeScaling (long sessionId, int scaling)
- int com.portsip.PortSipSdk.setChannelInputVolumeScaling (long sessionId, int scaling)
- Set< PortSipEnumDefine.AudioDevice > com.portsip.PortSipSdk.getAudioDevices ()
- int com.portsip.PortSipSdk.setAudioDevice (PortSipEnumDefine.AudioDevice defaultDevice)

Detailed Description

Function Documentation

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

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

Parameters:

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

Returns:

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

int com.portsip.PortSipSdk.enableVideoHardwareCodec (boolean *enableHWEncoder*, boolean *enableHWDecoder*)

Set enable/disable video Hardware codec.

Parameters:

enableHWEncoder	If it is set to true, the SDK will use video hardware encoder when available. By default it is true.
enableHWDecoder	If it is set to true, the SDK will use video hardware decoder when available. By default it is true.

Returns:

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

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

Set the video capturing resolution.

Parameters:

width	Video resolution, width
height	Video resolution, height

Returns:

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

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

When the camera does not support specified resolution, enable or disable SDK to crop and scale to the specified resolution.

Parameters:

enable	Enable or disable to crop or scale the video to fit in specified resolution. By default it is disabled.

Returns:

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

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

Set the audio bitrate.

Parameters:

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

Returns:

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

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

Set the video bitrate.

Parameters:

sessionId	The session ID of the call.
bitrateKbps	The video bitrate in KBPS.

Returns:

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

int com.portsip.PortSipSdk.setVideoFrameRate (long *sessionId*, int *frameRate*)

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

Parameters:

sessionId	The session ID of the call.
frameRate	The frame rate value, with its minimum of 5, and maximum value of 30. The greater the value is, the better video quality enabled and more bandwidth required;

Returns:

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

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

Send the video to remote side.

Parameters:

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

Returns:

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

void com.portsip.PortSipSdk.setLocalVideoWindow (PortSIPVideoRenderer *renderer*)

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

Parameters:

renderer	SurfaceView a SurfaceView for displaying local video image from camera.

int com.portsip.PortSipSdk.setRemoteVideoWindow (long *sessionId*, PortSIPVideoRenderer *renderer*)

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

Parameters:

sessionId	The session ID of the call.
renderer	SurfaceView a SurfaceView for displaying the received remote video image.

Returns:

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

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

Start/stop displaying the local video image.

Parameters:

state	Set to true to display local video image.
mirror	Set to true to display the mirror image of local video.

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

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

Parameters:

state	Set to true to enable.

Returns:

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

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

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

Parameters:

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

Returns:

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

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

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

Parameters:

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

Returns:

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

Set<PortSipEnumDefine.AudioDevice> com.portsip.PortSipSdk.getAudioDevices ()

Get current set of available/selectable audio devices.

Returns:

Current set of available/selectable audio devices.

int com.portsip.PortSipSdk.setAudioDevice (PortSipEnumDefine.AudioDevice *defaultDevice*)

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

Parameters:

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

Returns:

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

Call functions

Functions

- long com.portsip.PortSipSdk.call (String callee, boolean sendSdp, boolean videoCall)
- int com.portsip.PortSipSdk.rejectCall (long sessionId, int code)
- int com.portsip.PortSipSdk.hangUp (long sessionId)
- int com.portsip.PortSipSdk.answerCall (long sessionId, boolean videoCall)
- int com.portsip.PortSipSdk.updateCall (long sessionId, boolean enableAudio, boolean enableVideo)
- int com.portsip.PortSipSdk.hold (long sessionId)
- int com.portsip.PortSipSdk.unHold (long sessionId)
- int com.portsip.PortSipSdk.muteSession (long sessionId, boolean muteIncomingAudio, boolean muteOutgoingAudio, boolean muteIncomingVideo, boolean muteOutgoingVideo)
- int com.portsip.PortSipSdk.forwardCall (long sessionId, String forwardTo)
- long com.portsip.PortSipSdk.pickupBLFCall (String replaceDialogId, boolean videoCall)
- 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 a name only or full SIP URI, for example: user001 or sip:user001@sip.iptel.org or sip:user002@sip.yourdomain.com:5068
sendSdp	If it is set to false, the outgoing call will not include the SDP in INVITE message.
videoCall	If it is set to true and at least one video codec was added, the outgoing call will include the video codec into SDP. Otherwise no video codec will be added into outgoing SDP.

Returns:

If the function succeeds, it will return the session ID of the call, which is greater than 0. If the function fails, it will return a specific error code.
Note: the function success just means the outgoing call is processing, you need to detect the call final state in onInviteTrying, onInviteRinging, onInviteFailure callback events.

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

rejectCall Reject the incoming call.

Parameters:

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

Returns:

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

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

hangUp Hang up the call.

Parameters:

sessionId	Session ID of the call.

Returns:

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

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

answerCall Answer the incoming call.

Parameters:

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 does not include video codec answer anyway.

Returns:

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

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

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

Parameters:

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

Returns:

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

Remarks:

Example usage:

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

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

Example 2: Remove video stream from the current conversation.

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

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

To place a call on hold.

Parameters:

sessionId	The session ID of call.

Returns:

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

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

Take off hold.

Parameters:

sessionId	The session ID of call.

Returns:

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

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

Mute the specified audio or video session.

Parameters:

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

Returns:

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

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

Forward call to another one when receiving the incoming call.

Parameters:

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

Returns:

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

long com.portsip.PortSipSdk.pickupBLFCall (String *replaceDialogId*, boolean *videoCall*)

This function will be used for picking up a call based on the BLF (Busy Lamp Field) status.

Parameters:

replaceDialogId	The ID of the call which will be pickup. It comes with onDialogStateUpdated callback.
videoCall	Indicates pickup video call or audio call

Returns:

If the function succeeds, it will return a session ID that is greater than 0 to the new call, otherwise returns a specific error code that is less than 0.

Remarks:

The scenario is:

1. User 101 subscribed the user 100's call status: sendSubscription(mSipLib, "100", "dialog");
2. When 100 holds a call or 100 is ringing, onDialogStateUpdated callback will be triggered, and 101 will receive this callback. Now 101 can use pickupBLFCall function to pick the call rather than 100 to talk with caller.

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	DTMF tone could be sent via two methods: DTMF_RFC2833 or DTMF_INFO. The DTMF_RFC2833 is recommend.
code	The DTMF tone. Values include:

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

Parameters:

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

Returns:

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

Refer functions

Functions

- int com.portsip.PortSipSdk.refer (long sessionId, String referTo)
- int com.portsip.PortSipSdk.attendedRefer (long sessionId, long replaceSessionId, String referTo)
- int com.portsip.PortSipSdk.attendedRefer2 (long sessionId, long replaceSessionId, String replaceMethod, String target, String referTo)
- int com.portsip.PortSipSdk.outOfDialogRefer (long replaceSessionId, String replaceMethod, String target, String referTo)

- long com.portsip.PortSipSdk.acceptRefer (long referId, String referSignaling)
- int com.portsip.PortSipSdk.rejectRefer (long referId)

Detailed Description

Function Documentation

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

Transfer the current call to another callee.

Parameters:

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

Returns:

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

Remarks:

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

You can refer to the video on Youtube at:
https://www.youtube.com/watch?v=_2w9EGgr3FY, which will demonstrate how to complete the transfer.

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

Make an attended refer.

Parameters:

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

Returns:

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

Remarks:

Please read the sample project source code to get more details, or you can refer to the video on YouTube at:
https://www.youtube.com/watch?v=_2w9EGgr3FY
Note: Please use Windows Media Player to play the AVI file, which demonstrates how to complete the transfer.

int com.portsip.PortSipSdk.attendedRefer2 (long *sessionId*, long *replaceSessionId*, String *replaceMethod*, String *target*, String *referTo*)

Make an attended refer.

Parameters:

sessionId	The session ID of the call.
replaceSessionId	The session ID of the session to be replaced.
replaceMethod	The SIP method name to be added in the "Refer-To" header, usually INVITE

	or BYE.
target	The target to which the REFER message will be sent.
referTo	The URI to be added into the "Refer-To" header.

Returns:

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

int com.portsip.PortSipSdk.outOfDialogRefer (long *replaceSessionId*, String *replaceMethod*, String *target*, String *referTo*)

Make an attended refer.

Parameters:

replaceSessionId	The session ID of the session which will be replaced.
replaceMethod	The SIP method name which will be added in the "Refer-To" header, usually INVITE or BYE.
target	The target to which the REFER message will be sent.
referTo	The URI which will be added into the "Refer-To" header.

Returns:

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

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

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

Parameters:

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

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

Reject the REFER request.

Parameters:

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

Returns:

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

Send audio and video stream functions

Functions

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

Detailed Description

Function Documentation

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

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

Parameters:

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

Returns:

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

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

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

Parameters:

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

Returns:

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

Remarks:

Usually we should use it like below:

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

You can't have too much audio data at one time as we have 100ms audio buffer only. Once you put too much, data will be lost. It is recommended to send 20ms audio data every 20ms.

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

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

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

Parameters:

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

Returns:

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

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

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

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

Parameters:

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

Returns:

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

RTP packets, Audio stream and video stream callback

Functions

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

Detailed Description

functions

Function Documentation

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

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

Parameters:

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

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

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

Parameters:

sessionId	The session ID of call.
enable	Set to true to enable audio stream callback, or false to stop the callback.
enum_audioCallbackMode	The audio stream callback mode. Supported modes include ENUM_AUDIOSTREAM_NONE, ENUM_AUDIOSTREAM_LOCAL_PER_CHANNEL, ENUM_AUDIOSTREAM_REMOTE_PER_CHANNEL

	ENUM_AUDIOSTREAM_BOTH_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. Supported modes include ENUM_VIDEOSTREAM_NONE, ENUM_VIDEOSTREAM_LOCAL, ENUM_VIDEOSTREAM_REMOTE, ENUM_VIDEOSTREAM_BOTH.

Record functions

Functions

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

Detailed Description

Function Documentation

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

Start recording the call.

Parameters:

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

Mode	

Returns:

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

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

Stop recording.

Parameters:

sessionId	The session ID of call conversation.

Returns:

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

Play audio and video file to remote functions

Functions

- int com.portsip.PortSipSdk.playVideoFileToRemote (long sessionId, String aviFile, boolean loop, boolean playAudio)
- int com.portsip.PortSipSdk.stopPlayVideoFileToRemote (long sessionId)
- int com.portsip.PortSipSdk.playAudioFileToRemote (long sessionId, String filename, int fileSamplesPerSec, boolean loop)
- int com.portsip.PortSipSdk.stopPlayAudioFileToRemote (long sessionId)
- int com.portsip.PortSipSdk.playAudioFileToRemoteAsBackground (long sessionId, String filename, int fileSamplesPerSec)
- int com.portsip.PortSipSdk.stopPlayAudioFileToRemoteAsBackground (long sessionId)
- void com.portsip.PortSipSdk.audioPlayLoopbackTest (boolean enable)

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

Returns:

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

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

Stop play video file to remote side.

Parameters:

sessionId	Session ID of the call.

Returns:

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

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

Play a wave file to remote party.

Parameters:

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

Returns:

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

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

Stop playing wave file to remote side.

Parameters:

sessionId	Session ID of the call.

Returns:

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

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

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

Parameters:

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

Returns:

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

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

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

Parameters:

sessionId	Session ID of the call.

Returns:

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

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

Used for testing loopback for the audio device.

Parameters:

enable	Set to true to start testing audio loopback test; or set to false to stop.

Conference functions

Functions

- int com.portsip.PortSipSdk.createAudioConference ()
- int com.portsip.PortSipSdk.createVideoConference (PortSIPVideoRenderer conferenceVideoWindow, int videoWidth, int videoHeight, boolean displayLocalVideoInConference)
- void com.portsip.PortSipSdk.destroyConference ()
- int com.portsip.PortSipSdk.setConferenceVideoWindow (PortSIPVideoRenderer conferenceVideoWindow)
- int com.portsip.PortSipSdk.joinToConference (long sessionId)
- int com.portsip.PortSipSdk.removeFromConference (long sessionId)

Detailed Description

Function Documentation

int com.portsip.PortSipSdk.createAudioConference ()

Create an audio conference. It will fail if the existing conference is not ended yet.

Returns:

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

int com.portsip.PortSipSdk.createVideoConference (PortSIPVideoRenderer *conferenceVideoWindow*, int *videoWidth*, int *videoHeight*, boolean *displayLocalVideoInConference*)

Create a video conference. It will fail if the existing conference is not ended yet.

Parameters:

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

Returns:

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

void com.portsip.PortSipSdk.destroyConference ()

End the exist conference.

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

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

Parameters:

conferenceVideoWindow	SurfaceView The window which is used for displaying the conference video

Returns:

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

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

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

Parameters:

sessionId	Session ID of the call.

Returns:

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

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

Remove a session from an existing conference.

Parameters:

sessionId	Session ID of the call.

Returns:

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

RTP and RTCP QOS functions

Functions

- int com.portsip.PortSipSdk.setAudioRtcpBandwidth (long sessionId, int BitsRR, int BitsRS, int KBitsAS)
- int com.portsip.PortSipSdk.setVideoRtcpBandwidth (long sessionId, int BitsRR, int BitsRS, int KBitsAS)
- int com.portsip.PortSipSdk.enableAudioQos (boolean state)
- int com.portsip.PortSipSdk.enableVideoQos (boolean state)
- int com.portsip.PortSipSdk.setVideoMTU (int mtu)

Detailed Description

Function Documentation

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

Set the audio RTCP bandwidth parameters as RFC3556.

Parameters:

sessionId	Set the audio RTCP bandwidth parameters as 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, it will return value 0. If the function fails, it will return a specific error code.

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

Set the video RTCP bandwidth parameters as the RFC3556.

Parameters:

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

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

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

Parameters:

state	Set to YES to enable audio QoS and DSCP value will be 46; or NO to disable audio QoS and DSCP value will be 0.

Returns:

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

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

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

Parameters:

state	Set to YES to enable video QoS and DSCP value will be 34; or NO to disable video QoS and DSCP value will be 0.

Returns:

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

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

Set the MTU size for video RTP packet.

Parameters:

mtu	Set MTU value. Allow values range 512 - 65507. Default is 14000.

Returns:

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

RTP statistics functions

Functions

- int com.portsip.PortSipSdk.getAudioStatistics (long sessionId, int[] statistics)
- int com.portsip.PortSipSdk.getVideoStatistics (long sessionId, int[] statistics)

Detailed Description

Function Documentation

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

Obtain the statistics of audio channel.

Parameters:

sessionId	The session ID of call conversation.
statistics	Return audio statistics statistics[0] - The number of sent bytes. statistics[1] - The number of sent packets. statistics[2] - The number of sent but lost packet. statistics[3] - Fraction of sent but lost packet in percentage. statistics[4] - The round-trip time of the session, in milliseconds. statistics[5] - Sent Audio codec type. statistics[6] - The sent jitter, in milliseconds. statistics[7] - The sent audio level.It ranges 0 - 9. statistics[8] - The number of received bytes. statistics[9] - The number of received packets. statistics[10] - The number of received but lost packets. statistics[11] - Fraction of received but lost packet in percentage. statistics[12] - Received Audio codec type. statistics[13] - The received jitter, in milliseconds. statistics[14] - The received audio level.It ranges 0 - 9.

Returns:

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

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

Obtain the statistics of video channel.

Parameters:

sessionId	The session ID of call conversation.
statistics	Return Video statistics statistics[0] - The number of sent bytes. statistics[1] - The number of sent packets. statistics[2] - The number of sent but lost packet. statistics[3] - Fraction of sent lost in percentage. statistics[4] - The round-trip time of the session, in milliseconds.

	statistics[5] - Send Video codec type. statistics[6] - Frame width for the sent video. statistics[7] - Frame height for the sent video. statistics[8] - Bitrate in BPS for the sent video. statistics[9] - Frame rate for the sent video. statistics[10] - The number of received bytes. statistics[11] - The number of received packets. statistics[12] - The number of received but lost packet. statistics[13] - Fraction of received but lost packet in percentage. statistics[14] - Received Video codec type. statistics[15] - Frame width for the received video. statistics[16] - Frame height for the received video. statistics[17] - (This parameter is not implemented yet)Bitrate in BPS for the received video. statistics[18] - Framerate for the received video. statistics[19] - The number of sent bytes.

Returns:

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

Audio effect functions

Functions

- void com.portsip.PortSipSdk.enableVAD (boolean state)
- void com.portsip.PortSipSdk.enableAEC (boolean state)
- void com.portsip.PortSipSdk.enableCNG (boolean state)
- void com.portsip.PortSipSdk.enableAGC (boolean state)
- void com.portsip.PortSipSdk.enableANS (boolean state)

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, or false to disable.

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

Enable/disable AEC (Acoustic Echo Cancellation).

Parameters:

state	Set to true to enable AEC, or false to disable.

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

Enable/disable Comfort Noise Generator(CNG).

Parameters:

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

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

Enable/disable Automatic Gain Control(AGC).

Parameters:

state	Set to true to enable AEC, or false to disable.

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

Enable/disable Audio Noise Suppression(ANS).

Parameters:

state	Set to true to enable ANS, or false to disable.

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, boolean isSMS, byte[] message, int messageLength)
- long com.portsip.PortSipSdk.setPresenceMode (int mode)
- long com.portsip.PortSipSdk.setDefaultSubscriptionTime (int secs)
- long com.portsip.PortSipSdk.setDefaultPublicationTime (int secs)
- long com.portsip.PortSipSdk.presenceSubscribe (String contact, String subject)
- int com.portsip.PortSipSdk.presenceTerminateSubscribe (long subscribeId)
- int com.portsip.PortSipSdk.presenceAcceptSubscribe (long subscribeId)
- int com.portsip.PortSipSdk.presenceRejectSubscribe (long subscribeId)
- int com.portsip.PortSipSdk.setPresenceStatus (long subscribeId, String statusText)
- long com.portsip.PortSipSdk.sendSubscription (String to, String eventName)
  Send a SUBSCRIBE message to subscribe an event.
- int com.portsip.PortSipSdk.terminateSubscription (long subscribeId)

Detailed Description

Function Documentation

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

Send OPTIONS message.

Parameters:

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

Returns:

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

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

Send a INFO message to remote side in dialog.

Parameters:

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 will be sent with INFO message.

Returns:

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

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

Send a MESSAGE message to remote side in dialog.

Parameters:

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

Returns:

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

Remarks:

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

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

Example 2: Send a binary message.

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

long com.portsip.PortSipSdk.sendOutOfDialogMessage (String *to*, String *mimeType*, String *subMimeType*, boolean *isSMS*, 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.
isSMS	Set to YES to specify "messagetype=SMS" in the To line, or NO to disable.
message	The contents that will be sent with MESSAGE message. Binary data allowed.
messageLength	The message size.

Returns:

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

Remarks:

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

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

Example 2: Send a binary message.

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

long com.portsip.PortSipSdk.setPresenceMode (int *mode*)

Indicate the SDK uses the P2P mode for presence or presence agent mode.

Parameters:

mode	0 - P2P mode; 1 - Presence Agent mode. Default is P2P mode.

Returns:

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

Remarks:

Since presence agent mode requires the PBX/Server support the PUBLISH, please ensure you have your server and PortSIP PBX support this feature. For more details please visit: https://www.portsip.com/portsip-pbx

long com.portsip.PortSipSdk.setDefaultSubscriptionTime (int *secs*)

Set the default expiration time to be used when creating a subscription.

Parameters:

secs	The default expiration time of subscription.

Returns:

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

long com.portsip.PortSipSdk.setDefaultPublicationTime (int *secs*)

Set the default expiration time to be used when creating a publication.

Parameters:

secs	The default expiration time of publication.

Returns:

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

long com.portsip.PortSipSdk.presenceSubscribe (String *contact*, String *subject*)

Send a SUBSCRIBE message for presence to a contact.

Parameters:

contact	The target contact, it must be in the format of sip:contact001@sip.portsip.com.

subject	This subject text will be inserted into the SUBSCRIBE message. For example: "Hello, I'm Jason". The subject maybe is in UTF8 format. You should use UTF8 to decode it.

Returns:

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

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

Terminate the given presence subscription.

Parameters:

subscribeId	The ID of the subscription.

Returns:

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

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

Accept the presence SUBSCRIBE request which received from contact.

Parameters:

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

Returns:

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

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

Reject a presence SUBSCRIBE request received from contact.

Parameters:

subscribeId	Subscription ID. When receiving a SUBSCRIBE request from contact, the event onPresenceRecvSubscribe will be triggered. The event inclues the subscription ID.

Returns:

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

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

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

Parameters:

subscribeId	Subscription ID. When receiving a SUBSCRIBE request from contact, the event onPresenceRecvSubscribe that includes the Subscription ID will be triggered.
statusText	The state text of presence status. For example: "I'm here", offline must use "offline"

Returns:

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

long com.portsip.PortSipSdk.sendSubscription (String *to*, String *eventName*)

Send a SUBSCRIBE message to subscribe an event.

Parameters:

to	The user/extension will be subscribed.
eventName	The event name to be subscribed.

Returns:

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

Remarks:

Example 1, below code indicates that user/extension 101 is subscribed to MWI (Message Waiting notifications) for checking his voicemail: int32 mwiSubId = sendSubscription("sip:101@test.com", "message-summary");

Example 2, to monitor a user/extension call status, You can use code: sendSubscription(mSipLib, "100", "dialog"); Extension 100 refers to the user/extension to be monitored. Once being monitored, when extension 100 hold a call or is ringing, the onDialogStateUpdated callback will be triggered.

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

Terminate the given subscription.

Parameters:

subscribeId	The ID of the subscription.

Returns:

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

Remarks:

For example, if you want stop check the MWI, use below code:

```
terminateSubscription(mwiSubId);
```

Class Documentation

com.portsip.PortSipEnumDefine.AUDIOCODEC Interface Reference

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

- PortSipEnumDefine.java

com.portsip.PortSipEnumDefine.AudioDevice Enum Reference

Public Attributes

- **SPEAKER_PHONE**
- **WIRED_HEADSET**
- **EARPIECE**
- **BLUETOOTH**
- **NONE**

Detailed Description

AudioDevice list possible audio devices that we currently support.

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

- PortSipEnumDefine.java

com.portsip.OnPortSIPEvent Interface Reference

Public Member Functions

- void onRegisterSuccess (String reason, int code, String sipMessage)
- void onRegisterFailure (String reason, int code, String sipMessage)
- void onInviteIncoming (long sessionId, String callerDisplayName, String caller, String calleeDisplayName, String callee, String audioCodecs, String videoCodecs, boolean existsAudio, boolean existsVideo, String sipMessage)
- void onInviteTrying (long sessionId)
- void onInviteSessionProgress (long sessionId, String audioCodecs, String videoCodecs, boolean existsEarlyMedia, boolean existsAudio, boolean existsVideo, String sipMessage)
- void onInviteRinging (long sessionId, String statusText, int statusCode, String sipMessage)
- void onInviteAnswered (long sessionId, String callerDisplayName, String caller, String calleeDisplayName, String callee, String audioCodecs, String videoCodecs, boolean existsAudio, boolean existsVideo, String sipMessage)
- void onInviteFailure (long sessionId, String reason, int code, String sipMessage)
- void onInviteUpdated (long sessionId, String audioCodecs, String videoCodecs, boolean existsAudio, boolean existsVideo, String sipMessage)
- void onInviteConnected (long sessionId)
- void onInviteBeginingForward (String forwardTo)
- void onInviteClosed (long sessionId)
- void onDialogStateUpdated (String BLFMonitoredUri, String BLFDialogState, String BLFDialogId, String BLFDialogDirection)
- 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 onRecvNotifyOfSubscription (long subscribeId, String notifyMessage, byte[] messageData, int messageDataLength)
- 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, String sipMessage)
- 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 onSubscriptionFailure (long subscribeId, int statusCode)
- void onSubscriptionTerminated (long subscribeId)
- 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)

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

- OnPortSIPEvent.java

com.portsip.PortSIPCameraCapturer Class Reference

Inherits CapturerObserver.

Public Member Functions

- void **InitializeVideoCapturer** (Context applicationContext)
- void **UnInitializeVideoCapturer** ()
- void **switchCamera** (int newDeviceId)
- void **onCapturerStarted** (boolean success)
- void **onCapturerStopped** ()
- void **onByteBufferFrameCaptured** (byte[] data, int width, int height, int rotation, long timeStamp)
- void **onTextureFrameCaptured** (int width, int height, int oesTextureId, float[] transformMatrix, int rotation, long timestamp)
- void **onFrameCaptured** (VideoFrame frame)

Public Attributes

- CameraVideoCapturer **capturer**

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

- PortSIPCameraCapturer.java

com.portsip.PortSipEnumDefine Class Reference

Classes

- interface AUDIOCODEC
- enum AudioDevice

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_H264** = 125
- static final int **ENUM_VIDEOCODEC_VP8** = 120
- static final int **ENUM_VIDEOCODEC_VP9** = 122
- 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_UDP** = 3
- static final int **ENUM_TRANSPORT_PERS_TCP** = 4
- 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_AUDIOSTREAM_NONE** = 0
- static final int ENUM_AUDIOSTREAM_LOCAL_PER_CHANNEL = 1
- static final int ENUM_AUDIOSTREAM_REMOTE_PER_CHANNEL = 2
- static final int ENUM_AUDIOSTREAM_BOTH_PER_CHANNEL = 3
- 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_NONE = 0
- 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

Member Data Documentation

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

Used in startRecord only

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

Used in startRecord only

final int com.portsip.PortSipEnumDefine.ENUM_AUDIOSTREAM_LOCAL_PER_CHANNEL = 1 `[static]`

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

final int com.portsip.PortSipEnumDefine.ENUM_AUDIOSTREAM_REMOTE_PER_CHANNEL = 2 `[static]`

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

final int com.portsip.PortSipEnumDefine.ENUM_AUDIOSTREAM_BOTH_PER_CHANNEL = 3 `[static]`

Callback microphone and received audio stream for one channel based 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_NONE = 0 `[static]`

Not Recorded.

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

Only record the received data.

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

Only record the sent data.

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

Record both received and sent data.

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 **ECoreFailedCreateSdp** = -60090
- static final int **ECoreTrialVersionExpired** = -60091
- static final int **ECoreStackFailure** = -60092
- static final int **ECoreTransportExists** = -60093
- static final int **ECoreUnsupportTransport** = -60094
- static final int **ECoreAllowOnlyOneUser** = -60095
- static final int **ECoreUserNotFound** = -60096
- static final int **ECoreTransportsIncorrect** = -60097
- static final int **ECoreCreateTransportFailure** = -60098
- static final int **ECoreTransportNotSet** = -60099
- static final int **ECoreECreateSignalingFailure** = -60100
- static final int **ECoreArgumentIncorrect** = -60101
- 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 **EAudioVolumeOutOfRange** =-70020
- 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 **EVideoCantSetDeviceIdDuringCall** = -80010
- static final int **EVideoUnsupportCaptureRotate** = -80011
- static final int **VideoUnsupportCaptureResolution** = -80012
- static final int **ECameraSwitchTooOften** = -80013
- static final int **EMTUOutOfRange** = -80014
- 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)
- PortSipEnumDefine.AudioDevice **getSelectedAudioDevice** ()
- void DeleteCallManager ()
- int initialize (int enum_transport, String localIP, int localSIPPort, int enum_LogLevel, String LogPath, int maxLines, String agent, int audioDeviceLayer, int videoDeviceLayer, String TLSCertificatesRootPath, String TLSCipherList, boolean verifyTLSCertificate, String dnsServers)
- int setInstanceId (String instanceId)
- int setUser (String userName, String displayName, String authName, String password, String userDomain, String SIPServer, int SIPServerPort, String STUNServer, int STUNServerPort, String outboundServer, int outboundServerPort)
- void removeUser ()
  remove user account info.
- int registerServer (int expires, int retryTimes)
- int refreshRegistration (int expires)
- int unRegisterServer ()
- int setDisplayName (String displayName)
- 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)
- String getVersion ()
- int enableRport (boolean enable)
- int enableEarlyMedia (boolean enable)
  Enable/disable rport(RFC3581).
- int enableReliableProvisional (boolean enable)
- int enable3GppTags (boolean enable)
- void enableCallbackSignaling (boolean enableSending, boolean enableReceived)
- 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 state)
- void enableAutoCheckMwi (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 getSipMessageHeaderValue (String sipMessage, String headerName)
- long addSipMessageHeader (long sessionId, String methodName, int msgType, String headerName, String headerValue)
- int removeAddedSipMessageHeader (long addedSipMessageId)
- void clearAddedSipMessageHeaders ()
- long modifySipMessageHeader (long sessionId, String methodName, int msgType, String headerName, String headerValue)
- int removeModifiedSipMessageHeader (long modifiedSipMessageId)
- void clearModifiedSipMessageHeaders ()
- int setVideoDeviceId (int deviceId)
- int enableVideoHardwareCodec (boolean enableHWEncoder, boolean enableHWDecoder)
- int setVideoResolution (int width, int height)
- int setVideoCropAndScale (boolean enable)
- int setAudioBitrate (long sessionId, int enum_audiocodec, int bitrateKbps)
- int setVideoBitrate (long sessionId, int bitrateKbps)
- int setVideoFrameRate (long sessionId, int frameRate)
- int sendVideo (long sessionId, boolean send)
- void setLocalVideoWindow (PortSIPVideoRenderer renderer)
- int setRemoteVideoWindow (long sessionId, PortSIPVideoRenderer renderer)
- void displayLocalVideo (boolean state, boolean mirror)
- int setVideoNackStatus (boolean state)
- int setChannelOutputVolumeScaling (long sessionId, int scaling)
- int setChannelInputVolumeScaling (long sessionId, int scaling)
- Set< PortSipEnumDefine.AudioDevice > getAudioDevices ()
- int setAudioDevice (PortSipEnumDefine.AudioDevice defaultDevice)
- 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)
- long pickupBLFCall (String replaceDialogId, boolean videoCall)
- 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)
- int attendedRefer2 (long sessionId, long replaceSessionId, String replaceMethod, String target, String referTo)
- int outOfDialogRefer (long replaceSessionId, String replaceMethod, String target, 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)
- 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)
- void audioPlayLoopbackTest (boolean enable)
- int createAudioConference ()
- int createVideoConference (PortSIPVideoRenderer conferenceVideoWindow, int videoWidth, int videoHeight, boolean displayLocalVideoInConference)
- void destroyConference ()
- int setConferenceVideoWindow (PortSIPVideoRenderer 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 enableAudioQos (boolean state)
- int enableVideoQos (boolean state)
- int setVideoMTU (int mtu)
- int getAudioStatistics (long sessionId, int[] statistics)
- int getVideoStatistics (long sessionId, int[] statistics)
- void enableVAD (boolean state)
- void enableAEC (boolean state)
- void enableCNG (boolean state)
- void enableAGC (boolean state)
- void enableANS (boolean state)
- 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, boolean isSMS, byte[] message, int messageLength)
- long setPresenceMode (int mode)
- long setDefaultSubscriptionTime (int secs)
- long setDefaultPublicationTime (int secs)
- long presenceSubscribe (String contact, String subject)
- int presenceTerminateSubscribe (long subscribeId)
- int presenceAcceptSubscribe (long subscribeId)
- int presenceRejectSubscribe (long subscribeId)
- int setPresenceStatus (long subscribeId, String statusText)
- long sendSubscription (String to, String eventName)

  Send a SUBSCRIBE message to subscribe an event.
- int terminateSubscription (long subscribeId)
- void **receiveSIPEvent** (long 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 **setOnPortSIPEvent** (OnPortSIPEvent l)

Protected Member Functions

- void **setAudioManagerEvents** (AppRTCAudioManager.AudioManagerEvents audioManagerEvents)

Detailed Description

Author:

PortSIP Solutions, Inc. All rights reserved.

Version:

16

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

- PortSipSdk.java

com.portsip.PortSIPVideoRenderer Class Reference

Inherits SurfaceViewRenderer.

Classes

- enum ScalingType

Public Member Functions

- PortSIPVideoRenderer (Context context)
- PortSIPVideoRenderer (Context context, AttributeSet attrs)
- void **setScalingType** (ScalingType scalingType)
- void release ()
- void **surfaceDestroyed** (SurfaceHolder holder)
- void **renderFrame** (VideoRenderer.I420Frame frame)

Public Attributes

- long **nativeVideoRenderer** = 0

Detailed Description

Display the video stream on a SurfaceView.

Constructor & Destructor Documentation

com.portsip.PortSIPVideoRenderer.PortSIPVideoRenderer (Context *context*)

Standard View constructor. In order to render something, you must first call init().

com.portsip.PortSIPVideoRenderer.PortSIPVideoRenderer (Context *context*, AttributeSet *attrs*)

Standard View constructor. In order to render something, you must first call init().

Member Function Documentation

void com.portsip.PortSIPVideoRenderer.release ()

Block until any pending frame is returned and all GL resources released, even if an interrupt occurs. If an interrupt occurs during release(), the interrupt flag will be set. This function should be called before the Activity is destroyed and the EGLContext is still valid. If you don't call this function, the GL resources might leak.

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

- PortSIPVideoRenderer.java

com.portsip.PortSIPVideoRenderer.ScalingType Enum Reference

Public Attributes

- **SCALE_ASPECT_FIT**
- **SCALE_ASPECT_FILL**
- **SCALE_ASPECT_BALANCED**

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

- PortSIPVideoRenderer.java

Index