

PortSIP VoIP SDK Manual for iOS

Version 19.4
5/30/24

Table of Contents

Welcome to PortSIP VoIP SDK For iOS	4
Getting Started	4
Contents	4
SDK User Manual.....	4
Website	4
Support.....	4
Installation Prerequisites	4
Apple's iOS SDK	4
Note:.....	5
Device Requirements	5
Frequently Asked Questions	5
1. Where can I download the PortSIP VoIP SDK for test?	5
2. How can I compile the sample project?.....	5
3. How can I create a new project with PortSIP VoIP SDK?	5
4. Is the SDK thread safe?.....	6
5. Does the SDK support VoIP PUSH?	6
Module Index	7
Hierarchical Index.....	8
Class Index	9
Module Documentation	10
SDK Callback events	10
Register events	11
Call events.....	12
Refer events	16
Signaling events	18
MWI events.....	19
DTMF events	20
INFO/OPTIONS message events	21
Presence events	22
MESSAGE message events	23
Play audio and video file finished events	26
RTP callback events.....	27
Audio and video stream callback events.....	28
SDK functions.....	30
Initialize and register functions.....	31
NIC and local IP functions.....	35
Audio and video codecs functions	36
Additional settings functions	39
Access SIP message header functions	46
Audio and video functions.....	49
Call functions	54
Refer functions.....	59
Send audio and video stream functions	62
RTP packets, audio stream and video stream callback functions	64
Record functions	66
Play audio and video files to remote party.....	68
Conference functions	70
RTP and RTCP QOS functions	72
Media statistics functions.....	74
Audio effect functions.....	75
Send OPTIONS/INFO/MESSAGE functions.....	76
Presence functions	79
Keep awake functions.....	83
Audio Controller	84
Class Documentation	85
<PortSIPEventDelegate>	85

PortSIPSDK	87
PortSIPVideoRenderView	96
Index	98

Welcome to PortSIP VoIP SDK For iOS

Create your SIP-based application for multiple platforms (iOS, Android, Windows, macOS 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 macOS), Android Studio 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 function.

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

Development using the PortSIP VoIP/IMS SDK for iOS requires a Mac running macOS Monterey 12.5 or later., Xcode 14.0 or above.

Apple's iOS SDK

If you are not yet a registered Apple developer, to be able to develop applications for the iOS, you do need to become a registered Apple developer. After registered, Apple grants you free access to a select set of technical resources, tools, and information for developing with iOS,

macOS, and Safari. You can open [registration page](#) and enroll. Once registered, you can then go to the [iOS Dev Center](#), login and download the iOS SDK. The SDK contains documentation, frameworks, tools, and a simulator to help develop iOS applications. XCode (the developer toolset for iOS application development) is included in the download as well so you do not need to purchase any developer tools to build iOS applications - that is included in the enrollment fee. You will need to use a minimum of iOS SDK 11 for developing iPhone and iPod Touch applications. At the time of writing this document, iOS SDK 16 was the most recent version available and supported.

Note:

Beta and GM seed versions of the iOS SDK are generally not supported unless noted otherwise. Regardless of the iOS SDK you're using for development, you can still target your application for devices running on an older iOS version by configuring your Xcode project's iOS Deployment Target build settings. Be sure to add runtime checks where appropriate to ensure that you use only those iOS features available on the target platform/device. If your application attempts to use iOS features that are not available on the device, your application may crash.

Device Requirements

Applications built with PortSIP VoIP SDK for iOS can be run on iPhone 5s or higher, iPod touch 6 or higher, and iPad 5 or higher devices. These devices must be running iOS 11 or higher. We strongly recommend that you test your applications on actual devices to ensure that they work as expected and perform well. Testing on the simulator alone does not provide a good measure of how the application will perform on the physical device.

Frequently Asked Questions

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

All sample projects of the PortSIP VoIP SDK can be found and downloaded at: <https://www.portsip.com/download-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 Xcode.
4. Compile the sample project directly.

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

1. Download the Sample project and extract it to a directory.
2. Run the Xcode and create a new iOS Project.
3. Drag and drop PortSIPVoIPSDK.framework from Finder to XCode->Frameworks.
4. Add depend Frameworks: Build Phases->Link Binary With Libraries, add libresolv.tbd, VideoToolbox.framework, Network.framework, MetalKit.framework.
5. Add "-ObjC" to "Build Settings"-> "Other Linker Flags"
6. Add the code in .h file to import the SDK, example:
`#import <PortSIPVoIPSDK/PortSIPVoIPSDK.h>`
7. Inherit the interface [PortSIPEventDelegate](#) to process the callback events. For example:

```
@interface AppDelegate : UIResponder <UIApplicationDelegate,PortSIPEventDelegate>{
    PortSIPSDK* mPortSIPSDK;
}
@end
```

8. **Initialize sdk. For example:**

```
mPortSIPSDK = [[PortSIPSDK alloc] init];
mPortSIPSDK.delegate = self;
```

9. For more details, please read the Sample project source code.

4. 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", "onRTPPacketCallback" callbacks.

5. Does the SDK support VoIP PUSH?

Yes, please refer to <https://www.portsip.com/knowledge-base/> for more details.

Module Index

Modules

Here is a list of all modules:

SDK Callback events.....	10
Register events.....	11
Call events	12
Refer events	16
Signaling events.....	18
MWI events	19
DTMF events.....	20
INFO/OPTIONS message events	21
Presence events.....	22
MESSAGE message events.....	23
Play audio and video file finished events	26
RTP callback events	27
Audio and video stream callback events	28
 SDK functions	 30
Initialize and register functions	31
NIC and local IP functions	35
Audio and video codecs functions.....	36
Additional settings functions.....	39
Access SIP message header functions.....	46
Audio and video functions.....	49
Call functions.....	54
Refer functions	59
Send audio and video stream functions	62
RTP packets, audio stream and video stream callback functions.....	64
Record functions.....	66
Play audio and video files to remote party	68
Conference functions.....	70
RTP and RTCP QOS functions	72
Media statistics functions	74
Audio effect functions	75
Send OPTIONS/INFO/MESSAGE functions	76
Presence functions	79
Keep awake functions.....	83
Audio Controller.....	84

Hierarchical Index

Class Hierarchy

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

- <NSObject>
 - <PortSIPEventDelegate>..... 85
 - PortSIPSDK..... 87
- UIView
 - PortSIPVideoRenderView..... 96

Class Index

Class List

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

<u><PortSIPEventDelegate></u> (PortSIP SDK Callback events Delegate)	85
<u>PortSIPSDK</u> (PortSIP VoIP SDK functions class)	87
<u>PortSIPVideoRenderView</u> (PortSIP VoIP SDK Video Render View class)	96

Module Documentation

SDK Callback events

Modules

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

Detailed Description

SDK Callback events

Register events

Functions

- (void) - [<PortSIPEventDelegate>::onRegisterSuccess:statusCode:sipMessage:](#)
- (void) - [<PortSIPEventDelegate>::onRegisterFailure:statusCode:sipMessage:](#)

Detailed Description

Register events

Function Documentation

- (void) **onRegisterSuccess: (NSString *) *statusText* statusCode: (int) *statusCode* sipMessage: (NSString *) *sipMessage***

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

Parameters

<i>statusText</i>	The status text.
<i>statusCode</i>	The status code.
<i>sipMessage</i>	The SIP message received.

- (void) **onRegisterFailure: (NSString *) *statusText* statusCode: (int) *statusCode* sipMessage: (NSString *) *sipMessage***

If registration to SIP server fails, this event will be triggered.

Parameters

<i>statusText</i>	The status text.
<i>statusCode</i>	The status code.
<i>sipMessage</i>	The SIP message received.

Call events

Functions

- (void) - [<PortSIPEventDelegate>::onInviteIncoming:callerDisplayName:caller:calleeDisplayName:callee:audioCodecs:videoCodecs:existsAudio:existsVideo:sipMessage:](#)
- (void) - [<PortSIPEventDelegate>::onInviteTrying:](#)
- (void) - [<PortSIPEventDelegate>::onInviteSessionProgress:audioCodecs:videoCodecs:existsEarlyMedia:existsAudio:existsVideo:sipMessage:](#)
- (void) - [<PortSIPEventDelegate>::onInviteRinging:statusText:statusCode:sipMessage:](#)
- (void) - [<PortSIPEventDelegate>::onInviteAnswered:callerDisplayName:caller:calleeDisplayName:callee:audioCodecs:videoCodecs:existsAudio:existsVideo:sipMessage:](#)
- (void) - [<PortSIPEventDelegate>::onInviteFailure:callerDisplayName:caller:calleeDisplayName:callee:reason:code:sipMessage:](#)
- (void) - [<PortSIPEventDelegate>::onInviteUpdated:audioCodecs:videoCodecs:screenCodecs:existsAudio:existsVideo:existsScreen:sipMessage:](#)
- (void) - [<PortSIPEventDelegate>::onInviteConnected:](#)
- (void) - [<PortSIPEventDelegate>::onInviteBeginningForward:](#)
- (void) - [<PortSIPEventDelegate>::onInviteClosed:sipMessage:](#)
- (void) - [<PortSIPEventDelegate>::onDialogStateUpdated:BLFDialogState:BLFDialogId:BLFDialogDirection:](#)
- (void) - [<PortSIPEventDelegate>::onRemoteHold:](#)
- (void) - [<PortSIPEventDelegate>::onRemoteUnHold:audioCodecs:videoCodecs:existsAudio:existsVideo:](#)

Detailed Description

Function Documentation

- (void) onInviteIncoming: (long) *sessionId* callerDisplayName: (NSString *) *callerDisplayName* caller: (NSString *) *caller* calleeDisplayName: (NSString *) *calleeDisplayName* callee: (NSString *) *callee* audioCodecs: (NSString *) *audioCodecs* videoCodecs: (NSString *) *videoCodecs* existsAudio: (BOOL) *existsAudio* existsVideo: (BOOL) *existsVideo* sipMessage: (NSString *) *sipMessage*

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

Parameters

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

	codecs.
<i>existsAudio</i>	By setting to true, it indicates that this call includes the audio.
<i>existsVideo</i>	By setting to true, it indicates that this call includes the video.
<i>sipMessage</i>	The SIP message received.

- (void) onInviteTrying: (long) *sessionId*

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

Parameters

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

- (void) onInviteSessionProgress: (long) *sessionId* audioCodecs: (NSString *) *audioCodecs* videoCodecs: (NSString *) *videoCodecs* existsEarlyMedia: (BOOL) *existsEarlyMedia* existsAudio: (BOOL) *existsAudio* existsVideo: (BOOL) *existsVideo* sipMessage: (NSString *) *sipMessage*

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

Parameters

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

- (void) onInviteRinging: (long) *sessionId* statusText: (NSString *) *statusText* statusCode: (int) *statusCode* sipMessage: (NSString *) *sipMessage*

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

Parameters

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

- (void) onInviteAnswered: (long) *sessionId* callerDisplayName: (NSString *) *callerDisplayName* caller: (NSString *) *caller* calleeDisplayName: (NSString *) *calleeDisplayName* callee: (NSString *) *callee* audioCodecs: (NSString *) *audioCodecs* videoCodecs: (NSString *) *videoCodecs* existsAudio: (BOOL) *existsAudio* existsVideo: (BOOL) *existsVideo* sipMessage: (NSString *) *sipMessage*

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

Parameters

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

<i>existsAudio</i>	By setting to true, it indicates that this call includes the audio.
<i>existsVideo</i>	By setting to true, it indicates that this call includes the video.
<i>sipMessage</i>	The SIP message received.

- (void) onInviteFailure: (long) sessionId callerDisplayName: (NSString *) callerDisplayName caller: (NSString *) caller calleeDisplayName: (NSString *) calleeDisplayName callee: (NSString *) callee reason: (NSString *) reason code: (int) code sipMessage: (NSString *) sipMessage

If the outgoing/incoming call fails, this event will be triggered.

Parameters

<i>sessionId</i>	The session ID of the call.
<i>callerDisplayName</i>	The display name of caller
<i>caller</i>	The caller.
<i>calleeDisplayName</i>	The display name of callee.
<i>callee</i>	The callee.
<i>reason</i>	The failure reason.
<i>code</i>	The failure code.
<i>sipMessage</i>	The SIP message received.

- (void) onInviteUpdated: (long) sessionId audioCodecs: (NSString *) audioCodecs videoCodecs: (NSString *) videoCodecs screenCodecs: (NSString *) screenCodecs existsAudio: (BOOL) existsAudio existsVideo: (BOOL) existsVideo existsScreen: (BOOL) existsScreen sipMessage: (NSString *) sipMessage

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

Parameters

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

- (void) 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 is connected, otherwise it will return error.

Parameters

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

- (void) onInviteBeginingForward: (NSString *) forwardTo

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

Parameters

<i>forwardTo</i>	The target SIP URI for forwarding.
------------------	------------------------------------

- (void) onInviteClosed: (long) *sessionId* sipMessage: (NSString *) *sipMessage*

This event will be triggered once remote side closes the call.

Parameters

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

- (void) onDialogStateUpdated: (NSString *) *BLFMonitoredUri* BLFDialogState: (NSString *) *BLFDialogState* BLFDialogId: (NSString *) *BLFDialogId* BLFDialogDirection: (NSString *) *BLFDialogDirection*

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

Parameters

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

- (void) onRemoteHold: (long) *sessionId*

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

Parameters

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

- (void) onRemoteUnHold: (long) *sessionId* audioCodecs: (NSString *) *audioCodecs* videoCodecs: (NSString *) *videoCodecs* existsAudio: (BOOL) *existsAudio* existsVideo: (BOOL) *existsVideo*

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

Parameters

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

Refer events

Functions

- (void) - [<PortSIPEventDelegate>::onReceivedRefer:referId:to:from:referSipMessage:](#)
- (void) - [<PortSIPEventDelegate>::onReferAccepted:](#)
- (void) - [<PortSIPEventDelegate>::onReferRejected:reason:code:](#)
- (void) - [<PortSIPEventDelegate>::onTransferTrying:](#)
- (void) - [<PortSIPEventDelegate>::onTransferRinging:](#)
- (void) - [<PortSIPEventDelegate>::onACTVTransferSuccess:](#)
- (void) - [<PortSIPEventDelegate>::onACTVTransferFailure:reason:code:](#)

Detailed Description

Function Documentation

- (void) **onReceivedRefer:** (long) *sessionId* referId: (long) *referId* to: (NSString *) *to* from: (NSString *) *from* referSipMessage: (NSString *) *referSipMessage*

This event will be triggered once receiving a REFER message.

Parameters

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

- (void) **onReferAccepted:** (long) *sessionId*

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

Parameters

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

- (void) **onReferRejected:** (long) *sessionId* reason: (NSString *) *reason* code: (int) *code*

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

Parameters

<i>sessionId</i>	The session ID of the call.
<i>reason</i>	Reason for rejecting.
<i>code</i>	Rejecting code.

- (void) **onTransferTrying:** (long) *sessionId*

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

Parameters

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

- (void) **onTransferRinging:** (long) *sessionId*

When the refer call rings, this event will be triggered.

Parameters

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

- (void) onACTVTransferSuccess: (long) *sessionId*

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

Parameters

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

- (void) onACTVTransferFailure: (long) *sessionId* reason: (NSString *) *reason* code: (int) *code*

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

Parameters

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

Signaling events

Functions

- (void) - [<PortSIPEventDelegate>::onReceivedSignaling:message:](#)
- (void) - [<PortSIPEventDelegate>::onSendingSignaling:message:](#)

Detailed Description

Function Documentation

- (void) onReceivedSignaling: (long) *sessionId* message: (NSString *) *message*

This event will be triggered when receiving an SIP message. This event is disabled by default. To enable, use `enableCallbackSignaling`.

Parameters

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

- (void) onSendingSignaling: (long) *sessionId* message: (NSString *) *message*

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

Parameters

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

MWI events

Functions

- (void) -
[<PortSIPEventDelegate>::onWaitingVoiceMessage:urgentNewMessageCount:urgentOldMessageCount:newMessageCount:oldMessageCount:](#)
- (void) -
[<PortSIPEventDelegate>::onWaitingFaxMessage:urgentNewMessageCount:urgentOldMessageCount:newMessageCount:oldMessageCount:](#)

Detailed Description

Function Documentation

- (void) onWaitingVoiceMessage: (NSString *) *messageAccount*
urgentNewMessageCount: (int) *urgentNewMessageCount* urgentOldMessageCount:
(int) *urgentOldMessageCount* newMessageCount: (int) *newMessageCount*
oldMessageCount: (int) *oldMessageCount*

If there are any waiting voice messages (MWI), this event will be triggered.

Parameters

<i>messageAccount</i>	Account for voice message.
<i>urgentNewMessageCount</i>	Count of new urgent messages.
<i>urgentOldMessageCount</i>	Count of old urgent messages.
<i>newMessageCount</i>	Count of new messages.
<i>oldMessageCount</i>	Count of old messages.

- (void) onWaitingFaxMessage: (NSString *) *messageAccount*
urgentNewMessageCount: (int) *urgentNewMessageCount* urgentOldMessageCount:
(int) *urgentOldMessageCount* newMessageCount: (int) *newMessageCount*
oldMessageCount: (int) *oldMessageCount*

If there are any waiting fax messages (MWI), this event will be triggered.

Parameters

<i>messageAccount</i>	Account for fax message.
<i>urgentNewMessageCount</i>	Count of new urgent messages.
<i>urgentOldMessageCount</i>	Count of old urgent messages.
<i>newMessageCount</i>	Count of new messages.
<i>oldMessageCount</i>	Count of old messages.

DTMF events

Functions

- (void) - [<PortSIPEventDelegate>::onRecvDtmfTone:tone:](#)
-

Detailed Description

Function Documentation

- (void) onRecvDtmfTone: (long) *sessionId* tone: (int) *tone*

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

Parameters

<i>sessionId</i>	The session ID of the call.
<i>tone</i>	DTMF 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) - [<PortSIPEventDelegate>::onRecvOptions:](#)
- (void) - [<PortSIPEventDelegate>::onRecvInfo:](#)
- (void) - [<PortSIPEventDelegate>::onRecvNotifyOfSubscription:notifyMessage:messageData:messageDataLength:](#)

Detailed Description

Function Documentation

- (void) onRecvOptions: (NSString *) *optionsMessage*

This event will be triggered when receiving the OPTIONS message.

Parameters

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

- (void) onRecvInfo: (NSString *) *infoMessage*

This event will be triggered when receiving the INFO message.

Parameters

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

- (void) onRecvNotifyOfSubscription: (long) *subscribeId* notifyMessage: (NSString *) *notifyMessage* messageData: (unsigned char *) *messageData* messageDataLength: (int) *messageDataLength*

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

Parameters

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

Presence events

Functions

- (void) - [`<PortSIPEventDelegate>::onPresenceRecvSubscribe:fromDisplayName:from:subject:`](#)
- (void) - [`<PortSIPEventDelegate>::onPresenceOnline:from:stateText:`](#)
- (void) - [`<PortSIPEventDelegate>::onPresenceOffline:from:`](#)

Detailed Description

Function Documentation

- (void) **onPresenceRecvSubscribe:** (long) *subscribeId* **fromDisplayName:** (NSString *) *fromDisplayName* **from:** (NSString *) *from* **subject:** (NSString *) *subject*

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

Parameters

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

- (void) **onPresenceOnline:** (NSString *) *fromDisplayName* **from:** (NSString *) *from* **stateText:** (NSString *) *stateText*

This event will be triggered when the contact is online or changes presence status.

Parameters

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

- (void) **onPresenceOffline:** (NSString *) *fromDisplayName* **from:** (NSString *) *from*

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

Parameters

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

MESSAGE message events

Functions

- (void) - [<PortSIPEventDelegate>::onRecvMessage:mimeType:subMimeType:messageData:messageDataLength:](#)
- (void) - [<PortSIPEventDelegate>::onRecvOutOfDialogMessage:from:toDisplayName:to:mimeType:subMimeType:messageData:messageDataLength:sipMessage:](#)
- (void) - [<PortSIPEventDelegate>::onSendMessageSuccess:messageId:sipMessage:](#)
- (void) - [<PortSIPEventDelegate>::onSendMessageFailure:messageId:reason:code:sipMessage:](#)
- (void) - [<PortSIPEventDelegate>::onSendOutOfDialogMessageSuccess:fromDisplayName:from:toDisplayName:to:sipMessage:](#)
- (void) - [<PortSIPEventDelegate>::onSendOutOfDialogMessageFailure:fromDisplayName:from:toDisplayName:to:reason:code:sipMessage:](#)
- (void) - [<PortSIPEventDelegate>::onSubscriptionFailure:statusCode:](#)
- (void) - [<PortSIPEventDelegate>::onSubscriptionTerminated:](#)

Detailed Description

Function Documentation

- (void) onRecvMessage: (long) *sessionId* mimeType: (NSString *) *mimeType* subMimeType: (NSString *) *subMimeType* messageData: (unsigned char *) *messageData* messageDataLength: (int) *messageDataLength*

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

Parameters

<i>sessionId</i>	The session ID of the call.
<i>mimeType</i>	The message mime type.
<i>subMimeType</i>	The message sub mime type.
<i>messageData</i>	The received message body. It can be either text or binary data.
<i>messageDataLength</i>	The length of "messageData".

- (void) onRecvOutOfDialogMessage: (NSString *) *fromDisplayName* from: (NSString *) *from* toDisplayName: (NSString *) *toDisplayName* to: (NSString *) *to* mimeType: (NSString *) *mimeType* subMimeType: (NSString *) *subMimeType* messageData: (unsigned char *) *messageData* messageDataLength: (int) *messageDataLength* sipMessage: (NSString *) *sipMessage*

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

Parameters

<i>fromDisplayName</i>	The display name of sender.
<i>from</i>	The message sender.
<i>toDisplayName</i>	The display name of receiver.
<i>to</i>	The recipient.
<i>mimeType</i>	The message mime type.
<i>subMimeType</i>	The message sub mime type.

<i>messageData</i>	The received message body. It can be text or binary data.
<i>messageDataLength</i>	The length of "messageData".
<i>sipMessage</i>	The SIP message received.

- (void) onSendMessageSuccess: (long) sessionId messageId: (long) messageId sipMessage: (NSString *) sipMessage

This event will be triggered when the message is sent successfully in dialog.

Parameters

<i>sessionId</i>	The session ID of the call.
<i>messageId</i>	The message ID. It's equal to the return value of sendMessage function.
<i>sipMessage</i>	The SIP message received.

- (void) onSendMessageFailure: (long) sessionId messageId: (long) messageId reason: (NSString *) reason code: (int) code sipMessage: (NSString *) sipMessage

This event will be triggered when the message fails to be sent out of dialog.

Parameters

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

- (void) onSendOutOfDialogMessageSuccess: (long) messageId fromDisplayName: (NSString *) fromDisplayName from: (NSString *) from toDisplayName: (NSString *) toDisplayName to: (NSString *) to sipMessage: (NSString *) sipMessage

This event will be triggered when the message is sent successfully out of dialog.

Parameters

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

- (void) onSendOutOfDialogMessageFailure: (long) messageId fromDisplayName: (NSString *) fromDisplayName from: (NSString *) from toDisplayName: (NSString *) toDisplayName to: (NSString *) to reason: (NSString *) reason code: (int) code sipMessage: (NSString *) sipMessage

This event will be triggered when the message fails to be sent out of dialog.

Parameters

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

- (void) **onSubscriptionFailure:** (long) *subscribeId* statusCode: (int) *statusCode*

This event will be triggered on sending SUBSCRIBE failure.

Parameters

<i>subscribeId</i>	The ID of SUBSCRIBE request.
<i>statusCode</i>	The status code.

- (void) **onSubscriptionTerminated:** (long) *subscribeId*

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

Parameters

<i>subscribeId</i>	The ID of SUBSCRIBE request.
--------------------	------------------------------

Play audio and video file finished events

Functions

- (void) - [<PortSIPEventDelegate>::onPlayFileFinished:fileName:](#)
- (void) - [<PortSIPEventDelegate>::onStatistics:stat:](#)

Detailed Description

Function Documentation

- (void) onPlayFileFinished: (long) *sessionId* fileName: (NSString *) *fileName*

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

Parameters

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

- (void) onStatistics: (long) *sessionId* stat: (NSString *) *stat*

If getStatistics function is called, this event will be triggered once receiving a RTP statistics .

Parameters

<i>sessionId</i>	The session ID of the call.
<i>stat</i>	RTP statistics of a json format.

RTP callback events

Functions

- (void) - [<PortSIPEventDelegate>::onRTPPacketCallback:mediaType:direction:RTPPacket:packetSize:](#)

Detailed Description

Function Documentation

- (void) onRTPPacketCallback: (long) *sessionId* mediaType: (int) *mediaType* direction: (DIRECTION_MODE) *direction* RTPPacket: (unsigned char *) *RTPPacket* packetSize: (int) *packetSize*

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

Parameters

<i>sessionId</i>	The session ID of the call.
<i>mediaType</i>	If the received RTP packet is audio this parameter is 0 video this parameter is 1 screen this parameter is 2
<i>direction</i>	The RTP stream callback direction.

Type	Description
DIRECTION_SEND	Callback the send RTP stream for one channel based on the given sessionId.
DIRECTION_RECV	Callback the received RTP stream for one channel based on the given sessionId.

Parameters

<i>RTPPacket</i>	The memory of whole RTP packet.
<i>packetSize</i>	The size of received RTP Packet.

Note

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.

Audio and video stream callback events

Functions

- (void) - [<PortSIPEventDelegate>::onAudioRawCallback:audioCallbackMode:data:dataLength:samplingFreqHz:](#)
- (int) - [<PortSIPEventDelegate>::onVideoRawCallback:videoCallbackMode:width:height:data:dataLength:](#)

Detailed Description

Function Documentation

- (void) **onAudioRawCallback:** (long) *sessionId* **audioCallbackMode:** (int) *audioCallbackMode* **data:** (unsigned char *) *data* **dataLength:** (int) *dataLength* **samplingFreqHz:** (int) *samplingFreqHz*

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

Parameters

<i>sessionId</i>	The session ID of the call.
<i>audioCallbackMode</i>	The type that is passed in enableAudioStreamCallback function.
<i>data</i>	The memory of audio stream. It's in PCM format.
<i>dataLength</i>	The data size.
<i>samplingFreqHz</i>	The audio stream sample in HZ. For example, it could be 8000 or 16000.

Note

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.

- (int) **onVideoRawCallback:** (long) *sessionId* **videoCallbackMode:** (int) *videoCallbackMode* **width:** (int) *width* **height:** (int) *height* **data:** (unsigned char *) *data* **dataLength:** (int) *dataLength*

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

Parameters

<i>sessionId</i>	The session ID of the call.
<i>videoCallbackMode</i>	The type passed in enableVideoStreamCallback function.
<i>width</i>	The width of video image.
<i>height</i>	The height of video image.
<i>data</i>	The memory of video stream. It's in YUV420 format, such as YV12.
<i>dataLength</i>	The data size.

Returns

If you changed the sent video data, dataLength should be returned, otherwise 0.

Note

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.

SDK functions

Modules

- [Initialize and register functions](#)
- [NIC and local IP 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 functions](#)
- [Record functions](#)
- [Play audio and video files to remote party](#)
- [Conference functions](#)
- [RTP and RTCP QOS functions](#)
- [Media statistics functions](#)
- [Audio effect functions](#)
- [Send OPTIONS/INFO/MESSAGE functions](#)
- [Presence functions](#)
- [Keep awake functions](#)
- [Audio Controller](#)

Detailed Description

SDK functions

Initialize and register functions

Functions

- (int) - [PortSIPSDK::initialize:localIP:localSIPPort:loglevel:logPath:maxLine:agent:audioDeviceLayer:videoDeviceLayer:TLSCertificatesRootPath:TLSCipherList:verifyTLSCertificate:dnsServers:](#)
Initialize the SDK.
- (int) - [PortSIPSDK::setInstanceId:](#)
Set the instance Id, the outbound instanceId((RFC5626)) used in contact headers.
- (void) - [PortSIPSDK::unInitialize](#)
Un-initialize the SDK and release resources.
- (int) - [PortSIPSDK::setUser:displayName:authName:password:userDomain:SIPServer:SIPServerPort:STUNServer:STUNServerPort:outboundServer:outboundServerPort:](#)
Set user account info.
- (void) - [PortSIPSDK::removeUser](#)
Remove user account info.
- (int) - [PortSIPSDK::setDisplayNames:](#)
Set the display name of user.
- (int) - [PortSIPSDK::registerServer:retryTimes:](#)
Register to SIP proxy server (login to server)
- (int) - [PortSIPSDK::refreshRegistration:](#)
Refresh the registration manually after successfully registered.
- (int) - [PortSIPSDK::unRegisterServer:](#)
Un-register from the SIP proxy server.
- (int) - [PortSIPSDK::setLicenseKey:](#)
Set the license key. It must be called before setUser function.

Detailed Description

Initialize and register functions

Function Documentation

- (int) initialize: (TRANSPORT_TYPE) *transport* localIP: (NSString *) *localIP*
localSIPPort: (int) *localSIPPort* loglevel: (PORTSIP_LOG_LEVEL) *logLevel* logPath:
(NSString *) *logFilePath* maxLine: (int) *maxCallLines* agent: (NSString *) *sipAgent*

audioDeviceLayer: (int) *audioDeviceLayer* videoDeviceLayer: (int) *videoDeviceLayer*
 TLSCertificatesRootPath: (NSString *) *TLSCertificatesRootPath* TLSCipherList:
 (NSString *) *TLSCipherList* verifyTLSCertificate: (BOOL) *verifyTLSCertificate*
 dnsServers: (NSString *) *dnsServers*

Initialize the SDK.

Parameters

<i>transport</i>	Transport for SIP signaling. TRANSPORT_PERS_UDP/TRANSPORT_PERS_TCP is the PortSIP private transport for anti SIP blocking. It must be used with PERS Server.
<i>localIP</i>	The local computer IP address to be bounded (for example: 192.168.1.108). It will be used for sending and receiving SIP messages and RTP packets. If this IP is transferred in IPv6 format, the SDK will use IPv6. If you want the SDK to choose correct network interface (IP) automatically, please pass the "0.0.0.0"(for IPv4) or "::" (for IPv6).
<i>localSIPPort</i>	The SIP message transport listener port (for example: 5060).
<i>logLevel</i>	Set the application log level. The SDK will generate "PortSIP_Log_datetime.log" file if the log enabled.
<i>logFilePath</i>	The log file path. The path (folder) MUST be existent.
<i>maxCallLines</i>	Theoretically, unlimited lines are supported depending on the device capability. For SIP client recommended value ranges 1 - 100.
<i>sipAgent</i>	The User-Agent header to be inserted in SIP messages.
<i>audioDeviceLayer</i>	0 = Use OS default device 1 = Set to 1 to use the virtual audio device if the no sound device installed.
<i>videoDeviceLayer</i>	0 = Use OS default device 1 = Set to 1 to use the virtual video device if no camera installed.
<i>TLSCertificatesRootPath</i>	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.
<i>TLSCipherList</i>	Specify the TLS cipher list. This parameter is usually passed as empty so that the SDK will offer all available ciphers.
<i>verifyTLSCertificate</i>	Indicate if SDK will verify the TLS certificate. By setting to false, the SDK will not verify the validity of TLS certificate.
<i>dnsServers</i>	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 will return value 0. If the function fails, it will return a specific error code.

- (int) setInstanceId: (NSString *) *instanceId*

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

Parameters

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

Returns

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

- (int) setUser: (NSString *) *userName* displayName: (NSString *) *displayName* authName: (NSString *) *authName* password: (NSString *) *password* userDomain: (NSString *) *userDomain* SIPServer: (NSString *) *sipServer* SIPServerPort: (int) *sipServerPort* STUNServer: (NSString *) *stunServer* STUNServerPort: (int) *stunServerPort* outboundServer: (NSString *) *outboundServer* outboundServerPort: (int) *outboundServerPort*

Set user account info.

Parameters

<i>userName</i>	Account (username) of the SIP. It's usually provided by an IP-Telephony service provider.
<i>displayName</i>	The display name of user. You can set it as your like, such as "James Kend". It's optional.
<i>authName</i>	Authorization user name (usually equal to the username).
<i>password</i>	The password of user. It's optional.
<i>userDomain</i>	User domain. This parameter is optional. It allows to pass an empty string if you are not using domain.
<i>sipServer</i>	SIP proxy server IP or domain. For example: xx.xxx.xx.x or sip.xxx.com.
<i>sipServerPort</i>	Port of the SIP proxy server. For example: 5060.
<i>stunServer</i>	Stun server, used for NAT traversal. It's optional and can pass an empty string to disable STUN.
<i>stunServerPort</i>	STUN server port. It will be ignored if the outboundServer is empty.
<i>outboundServer</i>	Outbound proxy server. For example: sip.domain.com. It's optional and allows to pass an empty string if not using outbound server.
<i>outboundServerPort</i>	Outbound proxy server port. It will be ignored if the outboundServer is empty.

Returns

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

- (int) setDisplayName: (NSString *) *displayName*

Set the display name of user.

Parameters

<i>displayName</i>	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) registerServer: (int) *expires* retryTimes: (int) *retryTimes*

Register to SIP proxy server (login to server)

Parameters

<i>expires</i>	Registration refreshment interval in seconds. Maximum of 3600 allowed. It will be inserted into SIP REGISTER message headers.
<i>retryTimes</i>	The retry times if failed to refresh the registration. Once set to <= 0, the retry will be disabled and onRegisterFailure callback triggered for retry failure.

Returns

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

- (int) refreshRegistration: (int) *expires*

Refresh the registration manually after successfully registered.

Parameters

<i>expires</i>	Registration refreshment interval in seconds. Maximum of 3600 supported. It will be inserted into SIP REGISTER message header. If it's set to 0, default value will be used.
----------------	--

Returns

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

- (int) unregisterServer: (int) *waitMS*

Un-register from the SIP proxy server.

Parameters

<i>waitMS</i>	Wait for the server to reply that the un-registration is successful, waitMS is the longest waiting milliseconds, 0 means not waiting.
---------------	---

Returns

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

- (int) setLicenseKey: (NSString *) *key*

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

Parameters

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

Returns

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

NIC and local IP functions

Functions

- (int) - [PortSIPSDK::getNICNums](#)
Get the Network Interface Card numbers.
- (NSString *) - [PortSIPSDK::getLocalIpAddress:](#)
Get the local IP address by Network Interface Card index.

Detailed Description

Function Documentation

- (int) getNICNums

Get the Network Interface Card numbers.

Returns

If the function succeeds, it will return NIC numbers, which are greater than or equal to 0. If the function fails, it will return a specific error code.

- (NSString*) getLocalIpAddress: (int) *index*

Get the local IP address by Network Interface Card index.

Parameters

<i>index</i>	The IP address index. For example, suppose the PC has two NICs. If we want to obtain the second NIC IP, please set this parameter as 1, and the first NIC IP index 0.
--------------	---

Returns

The buffer for receiving the IP.

Audio and video codecs functions

Functions

- (int) - [PortSIPSDK::addAudioCodec:](#)
Enable an audio codec. It will appear in SDP.
- (int) - [PortSIPSDK::addVideoCodec:](#)
Enable a video codec. It will appear in SDP.
- (BOOL) - [PortSIPSDK::isAudioCodecEmpty](#)
Detect if the enabled audio codecs is empty.
- (BOOL) - [PortSIPSDK::isVideoCodecEmpty](#)
Detect if enabled video codecs is empty or not.
- (int) - [PortSIPSDK::setAudioCodecPayloadType:payloadType:](#)
Set the RTP payload type for dynamic audio codec.
- (int) - [PortSIPSDK::setVideoCodecPayloadType:payloadType:](#)
Set the RTP payload type for dynamic Video codec.
- (void) - [PortSIPSDK::clearAudioCodec](#)
Remove all enabled audio codecs.
- (void) - [PortSIPSDK::clearVideoCodec](#)
Remove all enabled video codecs.
- (int) - [PortSIPSDK::setAudioCodecParameter:parameter:](#)
Set the codec parameter for audio codec.
- (int) - [PortSIPSDK::setVideoCodecParameter:parameter:](#)
Set the codec parameter for video codec.

Detailed Description

Function Documentation

- (int) **addAudioCodec: (AUDIOCODEC_TYPE) codecType**

Enable an audio codec. It will appear in SDP.

Parameters

<i>codecType</i>	Audio codec type.
------------------	-------------------

Returns

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

- (int) addVideoCodec: (VIDEOCODEC_TYPE) *codecType*

Enable a video codec. It will appear in SDP.

Parameters

<i>codecType</i>	Video codec type.
------------------	-------------------

Returns

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

- (BOOL) isAudioCodecEmpty

Detect if the enabled audio codecs is empty.

Returns

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

- (BOOL) isVideoCodecEmpty

Detect if enabled video codecs is empty or not.

Returns

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

- (int) setAudioCodecPayloadType: (AUDIOCODEC_TYPE) *codecType* payloadType: (int) *payloadType*

Set the RTP payload type for dynamic audio codec.

Parameters

<i>codecType</i>	Audio codec type defined in the PortSIPTypes file.
<i>payloadType</i>	The new RTP payload type that you want to set.

Returns

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

- (int) setVideoCodecPayloadType: (VIDEOCODEC_TYPE) *codecType* payloadType: (int) *payloadType*

Set the RTP payload type for dynamic Video codec.

Parameters

<i>codecType</i>	Video codec type defined in the PortSIPTypes file.
<i>payloadType</i>	The new RTP payload type that you want to set.

Returns

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

- (int) setAudioCodecParameter: (AUDIOCODEC_TYPE) *codecType* parameter: (NSString *) *parameter*

Set the codec parameter for audio codec.

Parameters

<i>codecType</i>	Audio codec type defined in the PortSIPTypes file.
<i>parameter</i>	The parameter 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:

```
[myVoIPsdk setAudioCodecParameter:AUDIOCODEC_AMR parameter:"mode-set=0;
octet-align=1; robust-sorting=0"];
```

- (int) setVideoCodecParameter: (VIDEOCODEC_TYPE) *codecType* parameter: (NSString *) *parameter*

Set the codec parameter for video codec.

Parameters

<i>codecType</i>	Video codec type defined in the PortSIPTypes file.
<i>parameter</i>	The parameter in string format.

Returns

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

Remarks

Example:

```
[myVoIPsdk setVideoCodecParameter:VIDEOCODEC_H264
parameter:"profile-level-id=420033; packetization-mode=0"];
```

Additional settings functions

Functions

- (NSString *) - [PortSIPSDK::getVersion](#)
Get the current version number of the SDK.
- (int) - [PortSIPSDK::enableRport](#):
Enable/disable rport(RFC3581).
- (int) - [PortSIPSDK::enableEarlyMedia](#):
Enable/disable Early Media.
- (int) - [PortSIPSDK::enablePriorityIPv6Domain](#):
Enable/disable which allows specifying the preferred protocol when a domain supports both IPV4 and IPV6 simultaneously.
- (int) - [PortSIPSDK::setUriUserEncoding:enable](#):
Modifies the default URI user character needs to be escaped.
- (int) - [PortSIPSDK::setReliableProvisional](#):
Enable/disable PRACK.
- (int) - [PortSIPSDK::enable3GppTags](#):
Enable/disable the 3Gpp tags, including "ims.icsi.mmtel" and "g.3gpp.smsip".
- (void) - [PortSIPSDK::enableCallbackSignaling:enableReceived](#):
Enable/disable to callback the SIP messages.
- (int) - [PortSIPSDK::setSrtpPolicy](#):
Set the SRTP policy.
- (int) - [PortSIPSDK::setRtpPortRange:maximumRtpPort](#):
Set the RTP ports range for audio and video streaming.
- (int) - [PortSIPSDK::enableCallForward:forwardTo](#):
Enable call forwarding.
- (int) - [PortSIPSDK::disableCallForward](#):
Disable the call forwarding. The SDK is not forwarding any incoming calls once this function is called.
- (int) - [PortSIPSDK::enableSessionTimer:refreshMode](#):
Allows to periodically refresh Session Initiation Protocol (SIP) sessions by sending INVITE requests repeatedly.
- (int) - [PortSIPSDK::disableSessionTimer](#):
Disable the session timer.

- (void) - [PortSIPSDK::setDoNotDisturb:](#)
Enable the "Do not disturb" to enable/disable.
 - (void) - [PortSIPSDK::enableAutoCheckMwi:](#)
Enable the CheckMwi to enable/disable.
 - (int) - [PortSIPSDK::setRtpKeepAlive:keepAlivePayloadType:deltaTransmitTimeMS:](#)
Enable or disable to send RTP keep-alive packet when the call is established.
 - (int) - [PortSIPSDK::setKeepAliveTime:](#)
Enable or disable to send SIP keep-alive packet.
 - (int) - [PortSIPSDK::setAudioSamples:maxPtime:](#)
Set the audio capturing sample.
 - (int) - [PortSIPSDK::addSupportedMimeType:mimeType:subMimeType:](#)
-

Detailed Description

Function Documentation

- (NSString*) getVersion

Get the current version number of the SDK.

Returns

Return a current version number MAJOR.MINOR.PATCH of the SDK.

- (int) enableRport: (BOOL) enable

Enable/disable rport(RFC3581).

Parameters

<i>enable</i>	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) enableEarlyMedia: (BOOL) enable

Enable/disable Early Media.

Parameters

<i>enable</i>	Set to true to enable the SDK to 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) **enablePriorityIPv6Domain: (BOOL) *enable***

Enable/disable which allows specifying the preferred protocol when a domain supports both IPV4 and IPV6 simultaneously.

Parameters

<i>enable</i>	Set to true to enable priority IPv6 Domain. with the default priority being IPV4.
---------------	---

Returns

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

- (int) **setUriUserEncoding: (NSString *) *character* enable: (BOOL) *enable***

Modifies the default URI user character needs to be escaped.

Parameters

<i>character</i>	The character to be modified, set one character at a time.
<i>enable</i>	Whether escaping is required, true for allowing escaping, false for disabling escaping.

Returns

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

- (int) **setReliableProvisional: (int) *mode***

Enable/disable PRACK.

Parameters

<i>mode</i>	Modes work as follows: 0 - Never, Disable PRACK,By default the PRACK is disabled. 1 - SupportedEssential, Only send reliable provisionals if sending a body and far end supports. 2 - Supported, Always send reliable provisionals if far end supports. 3 - Required Always send reliable provisionals.
-------------	---

Returns

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

- (int) **enable3GppTags: (BOOL) *enable***

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

Parameters

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

Returns

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

- (void) enableCallbackSignaling: (BOOL) *enableSending* enableReceived: (BOOL) *enableReceived*

Enable/disable to callback the SIP messages.

Parameters

<i>enableSending</i>	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.
<i>enableReceived</i>	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.

- (int) setSrtpPolicy: (SRTP_POLICY) *srtpPolicy*

Set the SRTP policy.

Parameters

<i>srtpPolicy</i>	The SRTP policy.
-------------------	------------------

Returns

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

- (int) setRtpPortRange: (int) *minimumRtpPort* maximumRtpPort: (int) *maximumRtpPort*

Set the RTP ports range for audio and video streaming.

Parameters

<i>minimumRtpPort</i>	The minimum RTP port.
<i>maximumRtpPort</i>	The maximum RTP port.

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) enableCallForward: (BOOL) *forBusyOnly* forwardTo: (NSString *) *forwardTo*

Enable call forwarding.

Parameters

<i>forBusyOnly</i>	If this parameter is set as true, the SDK will forward all incoming calls when currently it's busy. If it's set as false, the SDK forward all incoming calls anyway.
<i>forwardTo</i>	The target of call forwarding. It must 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) disableCallForward

Disable the call forwarding. The SDK is not forwarding any incoming calls once this function is called.

Returns

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

- (int) enableSessionTimer: (int) *timerSeconds* refreshMode: (SESSION_REFRESH_MODE) *refreshMode*

Allows to periodically refresh Session Initiation Protocol (SIP) sessions by sending INVITE requests repeatedly.

Parameters

<i>timerSeconds</i>	The value of the refreshment interval in seconds. Minimum of 90 seconds required.
<i>refreshMode</i>	Allow to set the session refreshment by UAC or UAS: SESSION_REFRESH_UAC or SESSION_REFRESH_UAS;

Returns

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

Remarks

The INVITE requests, or re-INVITEs, are sent repeatedly 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 for remembering incoming and outgoing requests (stateful proxies) may continue to retain call state needlessly. If a UA fails to send a BYE message at the end of a session or if the BYE message is lost because of network problems, a stateful proxy does not know that the session has ended. The re-INVITEs ensure that active sessions stay active and completed sessions are terminated.

- (int) disableSessionTimer

Disable the session timer.

Returns

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

- (void) setDoNotDisturb: (BOOL) *state*

Enable the "Do not disturb" to enable/disable.

Parameters

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

- (void) enableAutoCheckMwi: (BOOL) *state*

Enable the CheckMwi to enable/disable.

Parameters

<i>state</i>	If it is set to true, the SDK will check Mwi automatically.
--------------	---

- (int) setRtpKeepAlive: (BOOL) *state* keepAlivePayloadType: (int) *keepAlivePayloadType* deltaTransmitTimeMS: (int) *deltaTransmitTimeMS*

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

Parameters

<i>state</i>	Set as true to allow to send the keep-alive packet during the conversation.
<i>keepAlivePayloadType</i>	The payload type of the keep-alive RTP packet. It's usually set to 126.
<i>deltaTransmitTimeMS</i>	The keep-alive RTP packet sending interval, in milliseconds. 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) setKeepAliveTime: (int) *keepAliveTime*

Enable or disable to send SIP keep-alive packet.

Parameters

<i>keepAliveTime</i>	This is the SIP keep-alive time interval in seconds. By setting 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) setAudioSamples: (int) *ptime* maxPtime: (int) *maxPtime*

Set the audio capturing sample.

Parameters

<i>ptime</i>	It should be a multiple of 10 between 10 - 60 (with 10 and 60 inclusive).
<i>maxPtime</i>	For the "maxptime" attribute, it should be a multiple of 10 between 10 - 60

	(with 10 and 60 inclusive). It cannot 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.

Remarks

It will appear in the SDP of INVITE and 200 OK message as "ptime" and "maxptime" attribute.

- (int) addSupportedMimeType: (NSString *) methodName mimeType: (NSString *) mimeType subMimeType: (NSString *) subMimeType

```
@brief Set the SDK to receive the SIP message that includes special mime type.

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

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

Remarks

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

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

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

```
[myVoIPSdk addSupportedMimeType:@"INFO" mimeType:@"text" subMimeType:@"plain"];
```

To receive the NOTIFY message with "application/media_control+xml":

```
[myVoIPSdk addSupportedMimeType:@"NOTIFY" mimeType:@"application"
subMimeType:@"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

- (NSString *) - [PortSIPSDK::getSipMessageHeaderValue:headerName:](#)
Access the SIP header of SIP message.
- (long) - [PortSIPSDK::addSipMessageHeader:methodName:msgType:headerName:headerValue:](#)
Add the SIP Message header into the specified outgoing SIP message.
- (int) - [PortSIPSDK::removeAddedSipMessageHeader:](#)
Remove the headers (custom header) added by addSipMessageHeader.
- (void) - [PortSIPSDK::clearAddedSipMessageHeaders](#)
Clear the added extension headers (custom headers)
- (long) - [PortSIPSDK::modifySipMessageHeader:methodName:msgType:headerName:headerValue:](#)
Modify the special SIP header value for every outgoing SIP message.
- (int) - [PortSIPSDK::removeModifiedSipMessageHeader:](#)
Remove the extension header (custom header) from every outgoing SIP message.
- (void) - [PortSIPSDK::clearModifiedSipMessageHeaders](#)
Clear the modified headers value, and do not modify every outgoing SIP message header values any longer.

Detailed Description

Function Documentation

- (NSString*) **getSipMessageHeaderValue: (NSString *) sipMessage headerName: (NSString *) headerName**

Access the SIP header of SIP message.

Parameters

<i>sipMessage</i>	The SIP message.
<i>headerName</i>	The header with which to access the SIP message.

Returns

If the function succeeds, it will return value headerValue. If the function fails, it will return value null.

Remarks

When receiving an SIP message in the onReceivedSignaling callback event, and user wishes to get SIP message header value, use getExtensionHeaderValue:

```
NSString* headerValue = [myVoIPSdk getSipMessageHeaderValue:message  
headerName:name];
```

**- (long) addSipMessageHeader: (long) sessionId methodName: (NSString *)
methodName msgType: (int) msgType headerName: (NSString *) headerName
headerValue: (NSString *) headerValue**

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

Parameters

<i>sessionId</i>	Add the header to the SIP message with the specified session Id only. By setting to -1, it will be added to all messages.
<i>methodName</i>	Just add the header to the SIP message with specified method name. For example: "INVITE", "REGISTER", "INFO" etc. If "ALL" specified, it will add all SIP messages.
<i>msgType</i>	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.
<i>headerName</i>	The header name that will appear in SIP message.
<i>headerValue</i>	The custom header value.

Returns

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

- (int) removeAddedSipMessageHeader: (long) addedSipMessageId

Remove the headers (custom header) added by addSipMessageHeader.

Parameters

<i>addedSipMessageId</i>	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) clearAddedSipMessageHeaders

Clear the added extension headers (custom headers)

Remarks

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

```
[myVoIPSdk addedSipMessageId:-1 methodName:@"ALL" msgType:3  
headerName:@"Blling" headerValue:@"usd100.00"];  
[myVoIPSdk addedSipMessageId:-1 methodName:@"ALL" msgType:3  
headerName:@"ServiceId" headerValue:@"8873456"];  
[myVoIPSdk clearAddedSipMessageHeaders];
```

- (long) modifySipMessageHeader: (long) *sessionId* methodName: (NSString *) *methodName* msgType: (int) *msgType* headerName: (NSString *) *headerName* headerValue: (NSString *) *headerValue*

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

Parameters

<i>sessionId</i>	The header to the SIP message with the specified session Id. By setting to -1, it will be added to all messages.
<i>methodName</i>	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.
<i>msgType</i>	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.
<i>headerName</i>	The SIP header name of which the value will be modified.
<i>headerValue</i>	The header 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.

- (int) removeModifiedSipMessageHeader: (long) *modifiedSipMessageId*

Remove the extension header (custom header) from every outgoing SIP message.

Parameters

<i>modifiedSipMessageId</i>	The modifiedSipMessageId is returned by modifySipMessageHeader.
-----------------------------	---

Returns

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

- (void) clearModifiedSipMessageHeaders

Clear the modified headers value, and do not modify every outgoing SIP message header values any longer.

Remarks

For example, to modify two headers' value for every outgoing SIP message and wish to clear it:

```
[myVoIPSdk removeModifiedSipMessageHeader:-1 methodName:@"ALL" msgType:3
headerName:@"Expires" headerValue:@"1000"];
[myVoIPSdk removeModifiedSipMessageHeader:-1 methodName:@"ALL" msgType:3
headerName:@"User-Agent" headerValue:@"MyTest Softphone 1.0"];
[myVoIPSdk clearModifiedSipMessageHeaders];
```


Audio and video functions

Functions

- (int) - [PortSIPSDK::setVideoDeviceId:](#)
Set the video device that will be used for video call.
- (int) - [PortSIPSDK::setVideoOrientation:](#)
Set the video Device Orientation.
- (int) - [PortSIPSDK::setVideoResolution:height:](#)
Set the video capturing resolution.
- (int) - [PortSIPSDK::setAudioBitrate:codecType:bitrateKbps:](#)
Set the audio bit rate.
- (int) - [PortSIPSDK::setVideoBitrate:bitrateKbps:](#)
Set the video bitrate.
- (int) - [PortSIPSDK::setVideoFrameRate:frameRate:](#)
Set the video frame rate.
- (int) - [PortSIPSDK::sendVideo:sendState:](#)
Send the video to remote side.
- (int) - [PortSIPSDK::setRemoteVideoWindow:remoteVideoWindow:](#)
Set the window for a session to display the received remote video image.
- (int) - [PortSIPSDK::setRemoteScreenWindow:remoteScreenWindow:](#)
Set the window for a session to display the received remote screen image.
- (int) - [PortSIPSDK::displayLocalVideo:mirror:localVideoWindow:](#)
Start/stop displaying the local video image.
- (int) - [PortSIPSDK::setVideoNackStatus:](#)
Enable/disable the NACK feature (RFC4585) to help to improve the video quality.
- (void) - [PortSIPSDK::muteMicrophone:](#)
Mute the device microphone. It's unavailable for Android and iOS.
- (void) - [PortSIPSDK::muteSpeaker:](#)
Mute the device speaker. It's unavailable for Android and iOS.
- (int) - [PortSIPSDK::setLoudspeakerStatus:](#)
Set the audio device that will be used for audio call.
- (int) - [PortSIPSDK::setChannelOutputVolumeScaling:scaling:](#)
- (int) - [PortSIPSDK::setChannelInputVolumeScaling:scaling:](#)

Detailed Description

Function Documentation

- (int) **setVideoDeviceId:** (int) *deviceId*

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

Parameters

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

Returns

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

- (int) **setVideoOrientation:** (int) *rotation*

Set the video Device Orientation.

Parameters

<i>rotation</i>	Device Orientation for video device (camera), e.g 0,90,180,270.
-----------------	---

Returns

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

- (int) **setVideoResolution:** (int) *width* height: (int) *height*

Set the video capturing resolution.

Parameters

<i>width</i>	Video width.
<i>height</i>	Video height.

Returns

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

- (int) **setAudioBitrate:** (long) *sessionId* codecType: (AUDIOCODEC_TYPE) *codecType* bitrateKbps: (int) *bitrateKbps*

Set the audio bit rate.

Parameters

<i>sessionId</i>	The session ID of the call.
<i>codecType</i>	Audio codec type.

<i>bitrateKbps</i>	The Audio bit rate in KBPS.
--------------------	-----------------------------

Returns

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

- (int) setVideoBitrate: (long) sessionId bitrateKbps: (int) bitrateKbps

Set the video bitrate.

Parameters

<i>sessionId</i>	The session ID of the call. Set it to -1 for all calls.
<i>bitrateKbps</i>	The video bit rate in KBPS.

Returns

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

- (int) setVideoFrameRate: (long) sessionId frameRate: (int) frameRate

Set the video frame rate.

Parameters

<i>sessionId</i>	The session ID of the call. Set it to -1 for all calls.
<i>frameRate</i>	The frame rate value, with its minimum value 5, and maximum value 30. Greater value renders better video quality but requires more bandwidth.

Returns

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

Remarks

Usually you do not need to call this function to set the frame rate, as the SDK uses default frame rate.

- (int) sendVideo: (long) sessionId sendState: (BOOL) sendState

Send the video to remote side.

Parameters

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

Returns

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

- (int) setRemoteVideoWindow: (long) sessionId remoteVideoWindow: ([PortSIPVideoRenderView](#) *) remoteVideoWindow

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

Parameters

<i>sessionId</i>	The session ID of the call.
<i>remoteVideoWindow</i>	The PortSIPVideoRenderView for displaying 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.

- (int) **setRemoteScreenWindow: (long) sessionId remoteScreenWindow: ([PortSIPVideoRenderView](#) *) remoteScreenWindow**

Set the window for a session to display the received remote screen image.

Parameters

<i>sessionId</i>	The session ID of the call.
<i>remoteScreenWindow</i>	The PortSIPVideoRenderView for displaying received remote screen image.

Returns

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

- (int) **displayLocalVideo: (BOOL) state mirror: (BOOL) mirror localVideoWindow: ([PortSIPVideoRenderView](#) *) localVideoWindow**

Start/stop displaying the local video image.

Parameters

<i>state</i>	Set to true to display local video image.
<i>mirror</i>	Set to true to display the mirror image of local video.
<i>localVideoWindow</i>	The PortSIPVideoRenderView for displaying local video image from camera.

Returns

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

- (int) **setVideoNackStatus: (BOOL) state**

Enable/disable the NACK feature (RFC4585) to help to improve the video quality.

Parameters

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

Returns

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

- (void) **muteMicrophone: (BOOL) mute**

Mute the device microphone. It's unavailable for Android and iOS.

Parameters

<i>mute</i>	If the value is set to true, the microphone is muted, or set to false to be un-muted.
-------------	---

- (void) **muteSpeaker: (BOOL) *mute***

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

Parameters

<i>mute</i>	If the value is set to true, the speaker is muted, or set to false to be un-muted.
-------------	--

- (int) **setLoudspeakerStatus: (BOOL) *enable***

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

Parameters

<i>enable</i>	By setting to true the SDK uses loudspeaker for audio call. This is 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.

Remarks

Allow to switch between earphone and loudspeaker only.

- (int) **setChannelOutputVolumeScaling: (long) *sessionId* scaling: (int) *scaling***

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

Parameters

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

Returns

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

- (int) **setChannelInputVolumeScaling: (long) *sessionId* scaling: (int) *scaling***

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

Parameters

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

Returns

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

Call functions

Functions

- (long) - [PortSIPSDK::call:sendSdp:videoCall:](#)
Make a call.
- (int) - [PortSIPSDK::rejectCall:code:](#)
rejectCall Reject the incoming call.
- (int) - [PortSIPSDK::hangUp:](#)
hangUp Hang up the call.
- (int) - [PortSIPSDK::answerCall:videoCall:](#)
answerCall Answer the incoming call.
- (int) - [PortSIPSDK::updateCall:enableAudio:enableVideo:](#)
Use the re-INVITE to update the established call.
- (int) - [PortSIPSDK::hold:](#)
Place a call on hold.
- (int) - [PortSIPSDK::unHold:](#)
Take off hold.
- (int) - [PortSIPSDK::muteSession:muteIncomingAudio:muteOutgoingAudio:muteIncomingVideo:muteOutgoingVideo:](#)
Mute the specified session audio or video.
- (int) - [PortSIPSDK::forwardCall:forwardTo:](#)
Forward the call to another user once received an incoming call.
- (long) - [PortSIPSDK::pickupBLFCall:videoCall:](#)
This function will be used for picking up a call based on the BLF (Busy Lamp Field) status.
- (int) - [PortSIPSDK::sendDtmf:dtmfMethod:code:dtmfDuration:playDtmfTone:](#)
Send DTMF tone.

Detailed Description

Function Documentation

- (long) call: (NSString *) callee sendSdp: (BOOL) sendSdp videoCall: (BOOL) videoCall

Make a call.

Parameters

<i>callee</i>	The callee. It can be a name only or full SIP URI. For example, user001, sip: user001@sip.iptel.org or sip: user002@sip.yourdomain.com :5068.
<i>sendSdp</i>	If set to false, the outgoing call will not include the SDP in INVITE message.
<i>videoCall</i>	If set to true and at least one video codec was added, the outgoing call will include the video codec into 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 being processed, and you need to detect the final state of calling in onInviteTrying, onInviteRinging, onInviteFailure callback events.

- (int) rejectCall: (long) sessionId code: (int) code

rejectCall Reject the incoming call.

Parameters

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

Returns

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

- (int) hangUp: (long) sessionId

hangUp Hang up the call.

Parameters

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

Returns

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

- (int) answerCall: (long) sessionId videoCall: (BOOL) videoCall

answerCall Answer the incoming call.

Parameters

<i>sessionId</i>	The session ID of the call.
<i>videoCall</i>	If the incoming call is a video call and the video codec is matched, set it to true to answer the video call.

	If set to false, the answer call will 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) updateCall: (long) *sessionId* enableAudio: (BOOL) *enableAudio* enableVideo: (BOOL) *enableVideo*

Use the re-INVITE to update the established call.

Parameters

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

Remarks

Example usage:

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

```
[myVoIPSdk clearVideoCodec];
[myVoIPSdk addVideoCodec:VIDEOCODEC_H264];
[myVoIPSdk updateCall:sessionId enableAudio:true enableVideo:true];
```

Example 2: Remove video stream from current conversation.

```
[myVoIPSdk updateCall:sessionId enableAudio:true enableVideo:false];
```

- (int) hold: (long) *sessionId*

Place a call on hold.

Parameters

<i>sessionId</i>	The 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) unHold: (long) *sessionId*

Take off hold.

Parameters

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

Returns

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

- (int) muteSession: (long) sessionId muteIncomingAudio: (BOOL) muteIncomingAudio muteOutgoingAudio: (BOOL) muteOutgoingAudio muteIncomingVideo: (BOOL) muteIncomingVideo muteOutgoingVideo: (BOOL) muteOutgoingVideo

Mute the specified session audio or video.

Parameters

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

Returns

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

- (int) forwardCall: (long) sessionId forwardTo: (NSString *) forwardTo

Forward the call to another user once received an incoming call.

Parameters

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

Returns

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

- (long) pickupBLFCall: (const char *) replaceDialogId videoCall: (BOOL) videoCall

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

Parameters

<i>replaceDialogId</i>	The ID of the call to be picked up. It comes with onDialogStateUpdated callback.
<i>videoCall</i>	Indicates if it is video call or audio call to be picked up.

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("100", "dialog");`
2. When 100 hold 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) `sendDtmf`: (long) `sessionId` `dtmfMethod`: (DTMF_METHOD) `dtmfMethod` `code`: (int) `code` `dtmfDuration`: (int) `dtmfDuration` `playDtmfTone`: (BOOL) `playDtmfTone`

Send DTMF tone.

Parameters

<i>sessionId</i>	The session ID of the call.
<i>dtmfMethod</i>	Support sending DTMF tone with two methods: DTMF_RFC2833 and DTMF_INFO. The DTMF RFC2833 is recommended.
<i>code</i>	The DTMF tone (0-16).

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

Parameters

<i>dtmfDuration</i>	The DTMF tone samples. Recommended value 160.
<i>playDtmfTone</i>	By setting to true, the SDK plays local DTMF tone sound when sending 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) - [PortSIPSDK::refer:referTo:](#)
Refer the current call to another one.
- (int) - [PortSIPSDK::attendedRefer:replaceSessionId:referTo:](#)
Make an attended refer.
- (int) - [PortSIPSDK::attendedRefer2:replaceSessionId:replaceMethod:target:referTo:](#)
Make an attended refer with specified request line and specified method embedded into the "Refer-To" header.
- (int) - [PortSIPSDK::outOfDialogRefer:replaceMethod:target:referTo:](#)
Send an out of dialog REFER to replace the specified call.
- (long) - [PortSIPSDK::acceptRefer:referSignaling:](#)
Once the REFER request accepted, a new call will be made if called this function. The function is usually called after onReceivedRefer callback event.
- (int) - [PortSIPSDK::rejectRefer:](#)
Reject the REFER request.

Detailed Description

Function Documentation

- (int) refer: (long) *sessionId* referTo: (NSString *) *referTo*

Refer the current call to another one.

Parameters

<i>sessionId</i>	The session ID of the call.
<i>referTo</i>	Target of the refer. It could 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

```
[myVoIPSdk refer:sessionId referTo:@"sip:testuser12@sip.portsip.com"];
```

You can watch the video on YouTube at <https://www.youtube.com/watch?v=2w9EGgr3FY>. It will demonstrate the transfer.

- (int) attendedRefer: (long) sessionId replaceSessionId: (long) replaceSessionId referTo: (NSString *) referTo

Make an attended refer.

Parameters

<i>sessionId</i>	The session ID of the call.
<i>replaceSessionId</i>	Session ID of the replaced call.
<i>referTo</i>	Target 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 for more details, or you can watch the video on YouTube at <https://www.youtube.com/watch?v=NezhIZW4IV4>, which will demonstrate the transfer.

- (int) attendedRefer2: (long) sessionId replaceSessionId: (long) replaceSessionId replaceMethod: (NSString *) replaceMethod target: (NSString *) target referTo: (NSString *) referTo

Make an attended refer with specified request line and specified method embedded into the "Refer-To" header.

Parameters

<i>sessionId</i>	Session ID of the call.
<i>replaceSessionId</i>	Session ID of the replaced call.
<i>replaceMethod</i>	The SIP method name which will be embedded in the "Refer-To" header, usually INVITE or BYE.
<i>target</i>	The target to which the REFER message will be sent. It appears in the "Request Line" of REFER message.
<i>referTo</i>	Target of the refer that appears in the "Refer-To" header. 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 for more details, or you can watch the video on YouTube at <https://www.youtube.com/watch?v=NezhIZW4IV4>, which will demonstrate the transfer.

- (int) outOfDialogRefer: (long) replaceSessionId replaceMethod: (NSString *) replaceMethod target: (NSString *) target referTo: (NSString *) referTo

Send an out of dialog REFER to replace the specified call.

Parameters

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

- (long) acceptRefer: (long) *referId* referSignaling: (NSString *) *referSignaling*

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

Parameters

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

Returns

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

- (int) rejectRefer: (long) *referId*

Reject the REFER request.

Parameters

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

Returns

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

Send audio and video stream functions

Functions

- (int) - [PortSIPSDK::enableSendPcmStreamToRemote:state:streamSamplesPerSec:](#)
Enable the SDK to send PCM stream data to remote side from another source instead of microphone.
- (int) - [PortSIPSDK::sendPcmStreamToRemote:data:](#)
Send the audio stream in PCM format from another source instead of audio device capturing (microphone).
- (int) - [PortSIPSDK::enableSendVideoStreamToRemote:state:](#)
Enable the SDK to send video stream data to remote side from another source instead of camera.
- (int) - [PortSIPSDK::sendVideoStreamToRemote:data:width:height:](#)
Send the video stream to remote side.

Detailed Description

Function Documentation

- (int) **enableSendPcmStreamToRemote:** (long) *sessionId* state: (BOOL) *state* streamSamplesPerSec: (int) *streamSamplesPerSec*

Enable the SDK to send PCM stream data to remote side from another source instead of microphone.

Parameters

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

Returns

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

Remarks

To send the PCM stream data to another side, this function MUST be called first.

- (int) **sendPcmStreamToRemote:** (long) *sessionId* data: (NSData *) *data*

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

Parameters

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

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:

```
[myVoIPSdk enableSendPcmStreamToRemote:sessionId state:YES  
streamSamplesPerSec:16000];  
[myVoIPSdk sendPcmStreamToRemote:sessionId data:data];
```

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) **enableSendVideoStreamToRemote:** (long) *sessionId* state: (BOOL) *state*

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

Parameters

<i>sessionId</i>	The session ID of call.
<i>state</i>	Set to true to enable the sending 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) **sendVideoStreamToRemote:** (long) *sessionId* data: (NSData *) *data* width: (int) *width* height: (int) *height*

Send the video stream to remote side.

Parameters

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

Returns

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

Remarks

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.

Usually we should use it like below:

```
[myVoIPSdk enableSendVideoStreamToRemote:sessionId state:YES];  
[myVoIPSdk sendVideoStreamToRemote:sessionId data:data width:352 height:288];
```

RTP packets, audio stream and video stream callback functions

Functions

- (int) - [PortSIPSDK::enableRtpCallback:mediaType:mode:](#)
Set the RTP callbacks to allow to access the sent and received RTP packets.
- (int) - [PortSIPSDK::enableAudioStreamCallback:enable:callbackMode:](#)
Enable/disable the audio stream callback.
- (int) - [PortSIPSDK::enableVideoStreamCallback:callbackMode:](#)
Enable/disable the video stream callback.

Detailed Description

Function Documentation

- (int) **enableRtpCallback:** (long) *sessionId* **mediaType:** (int) *mediaType* **mode:** (DIRECTION_MODE) *mode*

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

Parameters

<i>sessionId</i>	The session ID of call.
<i>mediaType</i>	0 -audio 1-video 2-screen.
<i>mode</i>	The RTP stream callback mode.
Type	Description
DIRECTION_SEND	Callback the send RTP stream for one channel based on the given <i>sessionId</i> .
DIRECTION_RECV	Callback the received RTP stream for one channel based on the given <i>sessionId</i> .
DIRECTION_SEND_RECV	Callback both local and remote RTP stream on the given <i>sessionId</i> .

Returns

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

- (int) **enableAudioStreamCallback:** (long) *sessionId* **enable:** (BOOL) *enable* **callbackMode:** (DIRECTION_MODE) *callbackMode*

Enable/disable the audio stream callback.

Parameters

<i>sessionId</i>	The session ID of call.
<i>enable</i>	Set to true to enable audio stream callback, or false to stop the callback.
<i>callbackMode</i>	The audio stream callback mode.

Type	Description
DIRECTION_SEND	Callback the audio stream from microphone for one channel based on the given sessionId.
DIRECTION_RECV	Callback the received audio stream for one channel based on the given sessionId.
DIRECTION_SEND_RECV	Callback both local and remote audio stream on the given sessionId.

Returns

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

Remarks

onAudioRawCallback event will be triggered if the callback is enabled.

- (int) enableVideoStreamCallback: (long) sessionId callbackMode: (DIRECTION_MODE) callbackMode

Enable/disable the video stream callback.

Parameters

<i>sessionId</i>	The session ID of call.
<i>callbackMode</i>	The video stream callback mode.

Mode	Description
DIRECTION_INACTIVE	Disable video stream callback.
DIRECTION_SEND	Local video stream callback.
DIRECTION_RECV	Remote video stream callback.
DIRECTION_SEND_RECV	Both of local and remote video stream callback.

Returns

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

Remarks

The onVideoRawCallback event will be triggered if the callback is enabled.

Record functions

Functions

- (int) - [PortSIPSDK::startRecord:recordFilePath:recordFileName:appendTimeStamp:channels:fileFormat:audioRecordMode:videoRecordMode:](#)
Start recording the call.
- (int) - [PortSIPSDK::stopRecord:](#)
Stop recording.

Detailed Description

Function Documentation

- (int) **startRecord:** (long) *sessionId* **recordFilePath:** (NSString *) *recordFilePath* **recordFileName:** (NSString *) *recordFileName* **appendTimeStamp:** (BOOL) *appendTimeStamp* **channels:** (int) *channels* **fileFormat:** (FILE_FORMAT) *recordFileFormat* **audioRecordMode:** (RECORD_MODE) *audioRecordMode* **videoRecordMode:** (RECORD_MODE) *videoRecordMode*

Start recording the call.

Parameters

<i>sessionId</i>	The session ID of call conversation.
<i>recordFilePath</i>	The filepath to which the recording will be saved. It must be existent.
<i>recordFileName</i>	The filename of the recording. For example audiorecord.wav or videorecord.avi.
<i>appendTimeStamp</i>	Set to true to append the timestamp to the filename of the recording.
<i>channels</i>	Set to record file audio channels, 1 - mono 2 - stereo.
<i>recordFileFormat</i>	The file format for the recording.
<i>audioRecordMode</i>	Allow to set audio recording mode. Support to record received and/or sent audio.
<i>videoRecordMode</i>	Allow to set video recording mode. Support to record received and/or sent video.

Returns

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

- (int) **stopRecord:** (long) *sessionId*

Stop recording.

Parameters

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

Returns

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

Play audio and video files to remote party

Functions

- (int) - [PortSIPSDK::startPlayingFileToRemote:fileUrl:loop:playAudio:](#)
Play a file to remote party.
- (int) - [PortSIPSDK::stopPlayingFileToRemote:](#)
Stop playing file to remote party.
- (int) - [PortSIPSDK::startPlayingFileLocally:loop:playVideoWindow:](#)
Play a file to locally.
- (int) - [PortSIPSDK::stopPlayingFileLocally](#)
Stop playing file to locally.
- (void) - [PortSIPSDK::audioPlayLoopbackTest:](#)
Used for the loop back testing against audio device.

Detailed Description

Function Documentation

- (int) **startPlayingFileToRemote:** (long) *sessionId* **fileUrl:** (NSString *) *fileUrl* **loop:** (BOOL) *loop* **playAudio:** (int) *playAudio*

Play a file to remote party.

Parameters

<i>sessionId</i>	Session ID of the call.
<i>fileUrl</i>	url or file name, such as "/test.mp4", "/test.wav", ..
<i>loop</i>	Set to false to stop playing video file when it is ended, or true to play it repeatedly.
<i>playAudio</i>	If it is set to 0 - Not play file audio. 1 - Play file audio, 2 - Play file audio mix with Mic

Returns

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

- (int) **stopPlayingFileToRemote:** (long) *sessionId*

Stop playing file to remote party.

Parameters

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

Returns

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

- (int) startPlayingFileLocally: (NSString *) *fileUrl* loop: (BOOL) *loop* playVideoWindow: ([PortSIPVideoRenderView](#) *) *playVideoWindow*

Play a file to locally.

Parameters

<i>fileUrl</i>	url or file name, such as "/test.mp4", "/test.wav",.
<i>loop</i>	Set to false to stop playing video file when it is ended, or true to play it repeatedly.
<i>playVideoWindow</i>	The PortSIPVideoRenderView used for displaying the video.

Returns

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

- (int) stopPlayingFileLocally

Stop playing file to locally.

Returns

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

- (void) audioPlayLoopbackTest: (BOOL) *enable*

Used for the loop back testing against audio device.

Parameters

<i>enable</i>	Set to true to start audio look back test, or false to stop.
---------------	--

Conference functions

Functions

- (int) - [PortSIPSDK::createAudioConference](#)
Create an audio conference.
- (int) - [PortSIPSDK::createVideoConference:videoWidth:videoHeight:layout:](#)
Create a video conference.
- (void) - [PortSIPSDK::destroyConference](#)
Destroy the existent conference.
- (int) - [PortSIPSDK::setConferenceVideoWindow:](#)
Set the window for a conference that is used to display the received remote video image.
- (int) - [PortSIPSDK::joinToConference:](#)
Join a session into existent conference. If the call is in hold, please un-hold first.
- (int) - [PortSIPSDK::removeFromConference:](#)
Remove a session from an existent conference.

Detailed Description

Function Documentation

- (int) createAudioConference

Create an audio conference.

Returns

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

- (int) createVideoConference: ([PortSIPVideoRenderView *](#)) *conferenceVideoWindow*
videoWidth: (int) videoWidth videoHeight: (int) videoHeight layout: (int) layout

Create a video conference.

Parameters

<i>conferenceVideoWindow</i>	The PortSIPVideoRenderView used for displaying the conference video.
<i>videoWidth</i>	The conference video width.

<i>videoHeight</i>	The conference video height.
<i>layout</i>	Conference Video layout, default is 0 - Adaptive. 0 - Adaptive(1,3,5,6) 1 - Only Local Video 2 - 2 video, PIP 3 - 2 video, Left and right 4 - 2 video, Up and Down 5 - 3 video 6 - 4 split video 7 - 5 video

Returns

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

**- (int) setConferenceVideoWindow: ([PortSIPVideoRenderView](#) *)
conferenceVideoWindow**

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

Parameters

<i>conferenceVideoWindow</i>	The PortSIPVideoRenderView used to display 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) joinToConference: (long) sessionId

Join a session into existent conference. If the call is in hold, please un-hold first.

Parameters

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

Returns

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

- (int) removeFromConference: (long) sessionId

Remove a session from an existent conference.

Parameters

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

Returns

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

RTP and RTCP QOS functions

Functions

- (int) - [PortSIPSDK::setAudioRtcpBandwidth:BitsRR:BitsRS:KBitsAS:](#)
Set the audio RTCP bandwidth parameters as the RFC3556.
- (int) - [PortSIPSDK::setVideoRtcpBandwidth:BitsRR:BitsRS:KBitsAS:](#)
Set the video RTCP bandwidth parameters as the RFC3556.
- (int) - [PortSIPSDK::enableAudioQos:](#)
Set the DSCP (differentiated services code point) value of QoS (Quality of Service) for audio channel.
- (int) - [PortSIPSDK::enableVideoQos:](#)
Set the DSCP (differentiated services code point) value of QoS (Quality of Service) for video channel.
- (int) - [PortSIPSDK::setVideoMTU:](#)
Set the MTU size for video RTP packet.

Detailed Description

Function Documentation

- (int) **setAudioRtcpBandwidth: (long) sessionId BitsRR: (int) BitsRR BitsRS: (int) BitsRS KBitsAS: (int) KBitsAS**

Set the audio RTCP bandwidth parameters as the RFC3556.

Parameters

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

Returns

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

- (int) **setVideoRtcpBandwidth: (long) sessionId BitsRR: (int) BitsRR BitsRS: (int) BitsRS KBitsAS: (int) KBitsAS**

Set the video RTCP bandwidth parameters as the RFC3556.

Parameters

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

Returns

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

- (int) enableAudioQos: (BOOL) *state*

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

Parameters

<i>state</i>	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) enableVideoQos: (BOOL) *state*

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

Parameters

<i>state</i>	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) setVideoMTU: (int) *mtu*

Set the MTU size for video RTP packet.

Parameters

<i>mtu</i>	Set MTU value. Allowed value ranges 512-65507. Other values will be automatically changed to the default 1400.
------------	--

Returns

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

Media statistics functions

Functions

- (int) - [PortSIPSDK::getStatistics:](#)
Obtain the statistics of channel. the event onStatistics will be triggered.

Detailed Description

Function Documentation

- (int) **getStatistics:** (long) *sessionId*

Obtain the statistics of channel. the event onStatistics will be triggered.

Parameters

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

Returns

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

Audio effect functions

Functions

- (void) - [PortSIPSDK::enableVAD:](#)
Enable/disable Voice Activity Detection (VAD).
- (void) - [PortSIPSDK::enableCNG:](#)
Enable/disable Comfort Noise Generator (CNG).

Detailed Description

Function Documentation

- (void) enableVAD: (BOOL) *state*

Enable/disable Voice Activity Detection (VAD).

Parameters

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

- (void) enableCNG: (BOOL) *state*

Enable/disable Comfort Noise Generator (CNG).

Parameters

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

Send OPTIONS/INFO/MESSAGE functions

Functions

- (int) - [PortSIPSDK::sendOptions:sdp:](#)
Send OPTIONS message.
- (int) - [PortSIPSDK::sendInfo:mimeType:subMimeType:infoContents:](#)
Send an INFO message to remote side in dialog.
- (long) - [PortSIPSDK::sendMessage:mimeType:subMimeType:message:messageLength:](#)
Send a MESSAGE message to remote side in dialog.
- (long) - [PortSIPSDK::sendOutOfDialogMessage:mimeType:subMimeType:isSMS:message:messageLength:](#)
Send an out of dialog MESSAGE message to remote side.

Detailed Description

Function Documentation

- (int) **sendOptions: (NSString *) to sdp: (NSString *) sdp**

Send OPTIONS message.

Parameters

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

Returns

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

- (int) **sendInfo: (long) sessionId mimeType: (NSString *) mimeType subMimeType: (NSString *) subMimeType infoContents: (NSString *) infoContents**

Send an INFO message to remote side in dialog.

Parameters

<i>sessionId</i>	The session ID of call.
<i>mimeType</i>	The mime type of INFO message.
<i>subMimeType</i>	The sub mime type of INFO message.
<i>infoContents</i>	The contents to 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) sendMessage: (long) sessionId mimeType: (NSString *) mimeType
subMimeType: (NSString *) subMimeType message: (NSData *) message
messageLength: (int) messageLength**

Send a MESSAGE message to remote side in dialog.

Parameters

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

Returns

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

Remarks

Example 1: Send a plain text message. Note: to send other languages text, please use the UTF-8 to encode the message before sending.

```
[myVoIPsdk sendMessage:sessionId mimeType:@"text" subMimeType:@"plain"  
message:data messageLength:dataLen];
```

Example 2: Send a binary message.

```
[myVoIPsdk sendMessage:sessionId mimeType:@"application"  
subMimeType:@"vnd.3gpp.sms" message:data messageLength:dataLen];
```

**- (long) sendOutOfDialogMessage: (NSString *) to mimeType: (NSString *)
mimeType subMimeType: (NSString *) subMimeType isSMS: (BOOL) isSMS
message: (NSData *) message messageLength: (int) messageLength**

Send an out of dialog MESSAGE message to remote side.

Parameters

<i>to</i>	The message recipient, such as sip: receiver@portsip.com .
<i>mimeType</i>	The mime type of MESSAGE message.
<i>subMimeType</i>	The sub mime type of MESSAGE message. @isSMS isSMS Set to YES to specify "messagetype=SMS" in the To line, or NO to disable.
<i>message</i>	The contents sent with MESSAGE message. Binary data allowed.
<i>messageLength</i>	The message size.

Returns

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

Remarks

Example 1: Send a plain text message. Note: to send text in other languages, please use UTF-8 to encode the message before sending.

```
[myVoIPsdk sendOutOfDialogMessage:@"sip:user1@sip.portsip.com" mimeType:@"text"  
subMimeType:@"plain" message:data messageLength:dataLen];
```

Example 2: Send a binary message.

```
[myVoIPsdk sendOutOfDialogMessage:@"sip:user1@sip.portsip.com"  
mimeType:@"application" subMimeType:@"vnd.3gpp.sms" isSMS:NO message:data  
messageLength:dataLen];
```

Presence functions

Functions

- (int) - [PortSIPSDK::setPresenceMode:](#)
Indicate that SDK uses the P2P mode for presence or presence agent mode.
- (int) - [PortSIPSDK::setDefaultSubscriptionTime:](#)
Set the default expiration time to be used when creating a subscription.
- (int) - [PortSIPSDK::setDefaultPublicationTime:](#)
Set the default expiration time to be used when creating a publication.
- (long) - [PortSIPSDK::presenceSubscribe:subject:](#)
Send a SUBSCRIBE message for subscribing the contact's presence status.
- (int) - [PortSIPSDK::presenceTerminateSubscribe:](#)
Terminate the given presence subscription.
- (int) - [PortSIPSDK::presenceAcceptSubscribe:](#)
Accept the presence SUBSCRIBE request which is received from contact.
- (int) - [PortSIPSDK::presenceRejectSubscribe:](#)
Reject a presence SUBSCRIBE request which is received from contact.
- (int) - [PortSIPSDK::setPresenceStatus:statusText:](#)
Send a NOTIFY message to contact to notify that presence status is online/offline/changed.
- (long) - [PortSIPSDK::sendSubscription:eventName:](#)
Send a SUBSCRIBE message to remote side.
- (int) - [PortSIPSDK::terminateSubscription:](#)
Terminate the given subscription.

Detailed Description

Function Documentation

- (int) **setPresenceMode: (PRESENCE_MODES) mode**

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

Parameters

<i>mode</i>	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>

- (int) setDefaultSubscriptionTime: (int) secs

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

Parameters

<i>secs</i>	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.

- (int) setDefaultPublicationTime: (int) secs

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

Parameters

<i>secs</i>	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) presenceSubscribe: (NSString *) contact subject: (NSString *) subject

Send a SUBSCRIBE message for subscribing the contact's presence status.

Parameters

<i>contact</i>	The target contact. It must be like sip: contact001@sip.portsip.com .
<i>subject</i>	This subject text will be inserted into the SUBSCRIBE message. For example: "Hello, I'm Jason". The subject maybe in UTF-8 format. You should use UTF-8 to decode it.

Returns

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

- (int) presenceTerminateSubscribe: (long) subscribeId

Terminate the given presence subscription.

Parameters

<i>subscribeId</i>	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) presenceAcceptSubscribe: (long) *subscribeId*

Accept the presence SUBSCRIBE request which is received from contact.

Parameters

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

Returns

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

Remarks

If the P2P presence mode is enabled, when someone subscribes your presence status, you will receive the subscription request in the callback, and you can use this function to reject it.

- (int) presenceRejectSubscribe: (long) *subscribeId*

Reject a presence SUBSCRIBE request which is received from contact.

Parameters

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

Returns

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

Remarks

If the P2P presence mode is enabled, when someone subscribe your presence status, you will receive the subscribe request in the callback, and you can use this function to accept it.

- (int) setPresenceStatus: (long) *subscribeId* statusText: (NSString *) *statusText*

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

Parameters

<i>subscribeId</i>	Subscription ID. When receiving a SUBSCRIBE request from contact, the event onPresenceRecvSubscribe that includes the Subscription ID will be triggered.
<i>statusText</i>	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 value a specific error code.

- (long) **sendSubscription:** (NSString *) **to** **eventName:** (NSString *) **eventName**

Send a SUBSCRIBE message to remote side.

Parameters

<i>to</i>	The subscribe user.
<i>eventName</i>	The event name to be subscribed.

Returns

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

Remarks

Example 1, to subscribe the MWI (Message Waiting notifications), You can use this code:
long mwiSubId = sendSubscription("sip:101@test.com", "message-summary");

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

- (int) **terminateSubscription:** (long) **subscribeId**

Terminate the given subscription.

Parameters

<i>subscribeId</i>	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);
```

Keep awake functions

Functions

- (BOOL) - [PortSIPSDK::startKeepAwake](#)
Keep VoIP awake in the background. @discussion If you want your application to be able to receive the incoming call while it's running in background, you should call this function in applicationDidEnterBackground.
- (BOOL) - [PortSIPSDK::stopKeepAwake](#)
Keep VoIP awake in the background. @discussion Call this function in applicationWillEnterForeground once your application comes back to foreground.

Detailed Description

Function Documentation

- (BOOL) startKeepAwake

Keep VoIP awake in the background. @discussion If you want your application to be able to receive the incoming call while it's running in background, you should call this function in applicationDidEnterBackground.

Returns

If the function succeeds, it will return value true. If the function fails, it will return value false.

- (BOOL) stopKeepAwake

Keep VoIP awake in the background. @discussion Call this function in applicationWillEnterForeground once your application comes back to foreground.

Returns

If the function succeeds, it will return value true. If the function fails, it will return value false.

Audio Controller

Functions

- (BOOL) - [PortSIPSDK::startAudio:](#)
Start Audio Device. @discussion Call it as AVAudioSessionInterruptionTypeEnded.
- (BOOL) - [PortSIPSDK::stopAudio:](#)
Stop Audio Device. @discussion Call it as AVAudioSessionInterruptionTypeBegan.
- (int) - [PortSIPSDK::enableCallKit:](#)
Enable/disable CallKit(Native Integration).

Detailed Description

Function Documentation

- (BOOL) startAudio: (AVAudioSession *) *audioSession*

Start Audio Device. @discussion Call it as AVAudioSessionInterruptionTypeEnded.

Returns

If the function succeeds, it will return value true. If the function fails, it will return value false.

- (BOOL) stopAudio: (AVAudioSession *) *audioSession*

Stop Audio Device. @discussion Call it as AVAudioSessionInterruptionTypeBegan.

Returns

If the function succeeds, it will return value true. If the function fails, it will return value false.

- (int) enableCallKit: (BOOL) *state*

Enable/disable CallKit(Native Integration).

Parameters

<i>state</i>	Set to true to enable CallKit, 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.

Class Documentation

<PortSIPEventDelegate> Protocol Reference

PortSIP SDK Callback events Delegate.

```
#import <PortSIPEventDelegate.h>
```

Inherits <NSObject>.

Instance Methods

- (void) - [onRegisterSuccess:statusCode:sipMessage:](#)
- (void) - [onRegisterFailure:statusCode:sipMessage:](#)
- (void) - [onInviteIncoming:callerDisplayName:caller:calleeDisplayName:callee:audioCodecs:videoCodecs:existsAudio:existsVideo:sipMessage:](#)
- (void) - [onInviteTrying:](#)
- (void) - [onInviteSessionProgress:audioCodecs:videoCodecs:existsEarlyMedia:existsAudio:existsVideo:sipMessage:](#)
- (void) - [onInviteRinging:statusText:statusCode:sipMessage:](#)
- (void) - [onInviteAnswered:callerDisplayName:caller:calleeDisplayName:callee:audioCodecs:videoCodecs:existsAudio:existsVideo:sipMessage:](#)
- (void) - [onInviteFailure:callerDisplayName:caller:calleeDisplayName:callee:reason:code:sipMessage:](#)
- (void) - [onInviteUpdated:audioCodecs:videoCodecs:screenCodecs:existsAudio:existsVideo:existsScreen:sipMessage:](#)
- (void) - [onInviteConnected:](#)
- (void) - [onInviteBeginningForward:](#)
- (void) - [onInviteClosed:sipMessage:](#)
- (void) - [onDialogStateUpdated:BLFDialogState:BLFDialogId:BLFDialogDirection:](#)
- (void) - [onRemoteHold:](#)
- (void) - [onRemoteUnHold:audioCodecs:videoCodecs:existsAudio:existsVideo:](#)
- (void) - [onReceivedRefer:referId:to:from:referSipMessage:](#)
- (void) - [onReferAccepted:](#)
- (void) - [onReferRejected:reason:code:](#)
- (void) - [onTransferTrying:](#)
- (void) - [onTransferRinging:](#)
- (void) - [onACTVTransferSuccess:](#)
- (void) - [onACTVTransferFailure:reason:code:](#)
- (void) - [onReceivedSignaling:message:](#)
- (void) - [onSendingSignaling:message:](#)
- (void) - [onWaitingVoiceMessage:urgentNewMessageCount:urgentOldMessageCount:newMessageCount:oldMessageCount:](#)
- (void) - [onWaitingFaxMessage:urgentNewMessageCount:urgentOldMessageCount:newMessageCount:oldMessageCount:](#)
- (void) - [onRecvDtmfTone:tone:](#)
- (void) - [onRecvOptions:](#)
- (void) - [onRecvInfo:](#)
- (void) - [onRecvNotifyOfSubscription:notifyMessage:messageData:messageDataLength:](#)
- (void) - [onPresenceRecvSubscribe:fromDisplayName:from:subject:](#)
- (void) - [onPresenceOnline:from:stateText:](#)
- (void) - [onPresenceOffline:from:](#)
- (void) - [onRecvMessage:mimeType:subMimeType:messageData:messageDataLength:](#)

- (void) - [onRecvOutOfDialogMessage:from:toDisplayName:to:mimeType:subMimeType:messageData:messageDataLength:sipMessage:](#)
- (void) - [onSendMessageSuccess:messageId:sipMessage:](#)
- (void) - [onSendMessageFailure:messageId:reason:code:sipMessage:](#)
- (void) - [onSendOutOfDialogMessageSuccess:fromDisplayName:from:toDisplayName:to:sipMessage:](#)
- (void) - [onSendOutOfDialogMessageFailure:fromDisplayName:from:toDisplayName:to:reason:code:sipMessage:](#)
- (void) - [onSubscriptionFailure:statusCode:](#)
- (void) - [onSubscriptionTerminated:](#)
- (void) - [onPlayFileFinished:fileName:](#)
- (void) - [onStatistics:stat:](#)
- (void) - [onRTPPacketCallback:mediaType:direction:RTPPacket:packetSize:](#)
- (void) - [onAudioRawCallback:audioCallbackMode:data:dataLength:samplingFreqHz:](#)
- (int) - [onVideoRawCallback:videoCallbackMode:width:height:data:dataLength:](#)

Detailed Description

PortSIP SDK Callback events Delegate.

Author

Copyright (c) 2006-2022 PortSIP Solutions, Inc. All rights reserved.

Version

19

See also

<http://www.PortSIP.com>

PortSIP SDK Callback events Delegate description.

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

- PortSIPEventDelegate.h

PortSIPSDK Class Reference

PortSIP VoIP SDK functions class.

```
#import <PortSIPSDK.h>
```

Inherits <NSObject>.

Instance Methods

- (int) - [initialize:localIP:localSIPPort:loglevel:logPath:maxLine:agent:audioDeviceLayer:videoDeviceLayer:TLSCertificatesRootPath:TLSCipherList:verifyTLSCertificate:dnsServers:](#)
Initialize the SDK.
- (int) - [setInstanceId:](#)
Set the instance Id, the outbound instanceId((RFC5626)) used in contact headers.
- (void) - [unInitialize](#)
Un-initialize the SDK and release resources.
- (int) - [setUser:displayName:authName:password:userDomain:SIPServer:SIPServerPort:STUNServer:STUNServerPort:outboundServer:outboundServerPort:](#)
Set user account info.
- (void) - [removeUser](#)
Remove user account info.
- (int) - [setDisplayNames:](#)
Set the display name of user.
- (int) - [registerServer:retryTimes:](#)
Register to SIP proxy server (login to server)
- (int) - [refreshRegistration:](#)
Refresh the registration manually after successfully registered.
- (int) - [unRegisterServer:](#)
Un-register from the SIP proxy server.
- (int) - [setLicenseKey:](#)
Set the license key. It must be called before setUser function.
- (int) - [getNICNums](#)
Get the Network Interface Card numbers.
- (NSString *) - [getLocalIpAddress:](#)
Get the local IP address by Network Interface Card index.
- (int) - [addAudioCodec:](#)

Enable an audio codec. It will appear in SDP.

- (int) - [addVideoCodec:](#)
Enable a video codec. It will appear in SDP.
- (BOOL) - [isAudioCodecEmpty](#)
Detect if the enabled audio codecs is empty.
- (BOOL) - [isVideoCodecEmpty](#)
Detect if enabled video codecs is empty or not.
- (int) - [setAudioCodecPayloadType:payloadType:](#)
Set the RTP payload type for dynamic audio codec.
- (int) - [setVideoCodecPayloadType:payloadType:](#)
Set the RTP payload type for dynamic Video codec.
- (void) - [clearAudioCodec](#)
Remove all enabled audio codecs.
- (void) - [clearVideoCodec](#)
Remove all enabled video codecs.
- (int) - [setAudioCodecParameter:parameter:](#)
Set the codec parameter for audio codec.
- (int) - [setVideoCodecParameter:parameter:](#)
Set the codec parameter for video codec.
- (NSString *) - [getVersion](#)
Get the current version number of the SDK.
- (int) - [enableRport:](#)
Enable/disable rport(RFC3581).
- (int) - [enableEarlyMedia:](#)
Enable/disable Early Media.
- (int) - [enablePriorityIPv6Domain:](#)
Enable/disable which allows specifying the preferred protocol when a domain supports both IPV4 and IPV6 simultaneously.
- (int) - [setUriUserEncoding:enable:](#)
Modifies the default URI user character needs to be escaped.
- (int) - [setReliableProvisional:](#)
Enable/disable PRACK.

- (int) - [enable3GppTags:](#)
Enable/disable the 3Gpp tags, including "ims.icsi.mmtel" and "g.3gpp.smsip".
- (void) - [enableCallbackSignaling:enableReceived:](#)
Enable/disable to callback the SIP messages.
- (int) - [setSrtpPolicy:](#)
Set the SRTP policy.
- (int) - [setRtpPortRange:maximumRtpPort:](#)
Set the RTP ports range for audio and video streaming.
- (int) - [enableCallForward:forwardTo:](#)
Enable call forwarding.
- (int) - [disableCallForward](#)
Disable the call forwarding. The SDK is not forwarding any incoming calls once this function is called.
- (int) - [enableSessionTimer:refreshMode:](#)
Allows to periodically refresh Session Initiation Protocol (SIP) sessions by sending INVITE requests repeatedly.
- (int) - [disableSessionTimer](#)
Disable the session timer.
- (void) - [setDoNotDisturb:](#)
Enable the "Do not disturb" to enable/disable.
- (void) - [enableAutoCheckMwi:](#)
Enable the CheckMwi to enable/disable.
- (int) - [setRtpKeepAlive:keepAlivePayloadType:deltaTransmitTimeMS:](#)
Enable or disable to send RTP keep-alive packet when the call is established.
- (int) - [setKeepAliveTime:](#)
Enable or disable to send SIP keep-alive packet.
- (int) - [setAudioSamples:maxPtime:](#)
Set the audio capturing sample.
- (int) - [addSupportedMimeType:mimeType:subMimeType:](#)
- (NSString *) - [getSipMessageHeaderValue:headerName:](#)
Access the SIP header of SIP message.
- (long) - [addSipMessageHeader:methodName:msgType:headerName:headerValue:](#)
Add the SIP Message header into the specified outgoing SIP message.

- (int) - [removeAddedSipMessageHeader:](#)
Remove the headers (custom header) added by addSipMessageHeader.
- (void) - [clearAddedSipMessageHeaders](#)
Clear the added extension headers (custom headers)
- (long) - [modifySipMessageHeader:methodName:msgType:headerName:headerValue:](#)
Modify the special SIP header value for every outgoing SIP message.
- (int) - [removeModifiedSipMessageHeader:](#)
Remove the extension header (custom header) from every outgoing SIP message.
- (void) - [clearModifiedSipMessageHeaders](#)
Clear the modified headers value, and do not modify every outgoing SIP message header values any longer.
- (int) - [setVideoDeviceId:](#)
Set the video device that will be used for video call.
- (int) - [setVideoOrientation:](#)
Set the video Device Orientation.
- (int) - [setVideoResolution:height:](#)
Set the video capturing resolution.
- (int) - [setAudioBitrate:codecType:bitrateKbps:](#)
Set the audio bit rate.
- (int) - [setVideoBitrate:bitrateKbps:](#)
Set the video bitrate.
- (int) - [setVideoFrameRate:frameRate:](#)
Set the video frame rate.
- (int) - [sendVideo:sendState:](#)
Send the video to remote side.
- (int) - [setRemoteVideoWindow:remoteVideoWindow:](#)
Set the window for a session to display the received remote video image.
- (int) - [setRemoteScreenWindow:remoteScreenWindow:](#)
Set the window for a session to display the received remote screen image.
- (int) - [displayLocalVideo:mirror:localVideoWindow:](#)
Start/stop displaying the local video image.
- (int) - [setVideoNackStatus:](#)
Enable/disable the NACK feature (RFC4585) to help to improve the video quality.

- (void) - [muteMicrophone:](#)
Mute the device microphone. It's unavailable for Android and iOS.
- (void) - [muteSpeaker:](#)
Mute the device speaker. It's unavailable for Android and iOS.
- (int) - [setLoudspeakerStatus:](#)
Set the audio device that will be used for audio call.
- (int) - [setChannelOutputVolumeScaling:scaling:](#)
- (int) - [setChannelInputVolumeScaling:scaling:](#)
- (long) - [call:sendSdp:videoCall:](#)
Make a call.
- (int) - [rejectCall:code:](#)
rejectCall Reject the incoming call.
- (int) - [hangUp:](#)
hangUp Hang up the call.
- (int) - [answerCall:videoCall:](#)
answerCall Answer the incoming call.
- (int) - [updateCall:enableAudio:enableVideo:](#)
Use the re-INVITE to update the established call.
- (int) - [hold:](#)
Place a call on hold.
- (int) - [unHold:](#)
Take off hold.
- (int) - [muteSession:muteIncomingAudio:muteOutgoingAudio:muteIncomingVideo:muteOutgoingVideo:](#)
Mute the specified session audio or video.
- (int) - [forwardCall:forwardTo:](#)
Forward the call to another user once received an incoming call.
- (long) - [pickupBLFCall:videoCall:](#)
This function will be used for picking up a call based on the BLF (Busy Lamp Field) status.
- (int) - [sendDtmf:dtmfMethod:code:dtmfDratation:playDtmfTone:](#)
Send DTMF tone.
- (int) - [refer:referTo:](#)
Refer the current call to another one.

- (int) - [attendedRefer:replaceSessionId:referTo:](#)
Make an attended refer.
- (int) - [attendedRefer2:replaceSessionId:replaceMethod:target:referTo:](#)
Make an attended refer with specified request line and specified method embedded into the "Refer-To" header.
- (int) - [outOfDialogRefer:replaceMethod:target:referTo:](#)
Send an out of dialog REFER to replace the specified call.
- (long) - [acceptRefer:referSignaling:](#)
Once the REFER request accepted, a new call will be made if called this function. The function is usually called after onReceivedRefer callback event.
- (int) - [rejectRefer:](#)
Reject the REFER request.
- (int) - [enableSendPcmStreamToRemote:state:streamSamplesPerSec:](#)
Enable the SDK to send PCM stream data to remote side from another source instead of microphone.
- (int) - [sendPcmStreamToRemote:data:](#)
Send the audio stream in PCM format from another source instead of audio device capturing (microphone).
- (int) - [enableSendVideoStreamToRemote:state:](#)
Enable the SDK to send video stream data to remote side from another source instead of camera.
- (int) - [sendVideoStreamToRemote:data:width:height:](#)
Send the video stream to remote side.
- (int) - [enableRtpCallback:mediaType:mode:](#)
Set the RTP callbacks to allow to access the sent and received RTP packets.
- (int) - [enableAudioStreamCallback:enable:callbackMode:](#)
Enable/disable the audio stream callback.
- (int) - [enableVideoStreamCallback:callbackMode:](#)
Enable/disable the video stream callback.
- (int) - [startRecord:recordFilePath:recordFileName:appendTimeStamp:channels:fileFormat:audioRecordMode:videoRecordMode:](#)
Start recording the call.
- (int) - [stopRecord:](#)
Stop recording.

- (int) - [startPlayingFileToRemote:fileUrl:loop:playAudio:](#)
Play a file to remote party.
- (int) - [stopPlayingFileToRemote:](#)
Stop playing file to remote party.
- (int) - [startPlayingFileLocally:loop:playVideoWindow:](#)
Play a file to locally.
- (int) - [stopPlayingFileLocally](#)
Stop playing file to locally.
- (void) - [audioPlayLoopbackTest:](#)
Used for the loop back testing against audio device.
- (int) - [createAudioConference](#)
Create an audio conference.
- (int) - [createVideoConference:videoWidth:videoHeight:layout:](#)
Create a video conference.
- (void) - [destroyConference](#)
Destroy the existent conference.
- (int) - [setConferenceVideoWindow:](#)
Set the window for a conference that is used to display the received remote video image.
- (int) - [joinToConference:](#)
Join a session into existent conference. If the call is in hold, please un-hold first.
- (int) - [removeFromConference:](#)
Remove a session from an existent conference.
- (int) - [setAudioRtcpBandwidth:BitsRR:BitsRS:KBitsAS:](#)
Set the audio RTCP bandwidth parameters as the RFC3556.
- (int) - [setVideoRtcpBandwidth:BitsRR:BitsRS:KBitsAS:](#)
Set the video RTCP bandwidth parameters as the RFC3556.
- (int) - [enableAudioQos:](#)
Set the DSCP (differentiated services code point) value of QoS (Quality of Service) for audio channel.
- (int) - [enableVideoQos:](#)
Set the DSCP (differentiated services code point) value of QoS (Quality of Service) for video channel.

- (int) - [setVideoMTU:](#)
Set the MTU size for video RTP packet.
- (int) - [getStatistics:](#)
Obtain the statistics of channel. the event onStatistics will be triggered.
- (void) - [enableVAD:](#)
Enable/disable Voice Activity Detection (VAD).
- (void) - [enableCNG:](#)
Enable/disable Comfort Noise Generator (CNG).
- (int) - [sendOptions:sdp:](#)
Send OPTIONS message.
- (int) - [sendInfo:mimeType:subMimeType:infoContents:](#)
Send an INFO message to remote side in dialog.
- (long) - [sendMessage:mimeType:subMimeType:message:messageLength:](#)
Send a MESSAGE message to remote side in dialog.
- (long) - [sendOutOfDialogMessage:mimeType:subMimeType:isSMS:message:messageLength:](#)
Send an out of dialog MESSAGE message to remote side.
- (int) - [setPresenceMode:](#)
Indicate that SDK uses the P2P mode for presence or presence agent mode.
- (int) - [setDefaultSubscriptionTime:](#)
Set the default expiration time to be used when creating a subscription.
- (int) - [setDefaultPublicationTime:](#)
Set the default expiration time to be used when creating a publication.
- (long) - [presenceSubscribe:subject:](#)
Send a SUBSCRIBE message for subscribing the contact's presence status.
- (int) - [presenceTerminateSubscribe:](#)
Terminate the given presence subscription.
- (int) - [presenceAcceptSubscribe:](#)
Accept the presence SUBSCRIBE request which is received from contact.
- (int) - [presenceRejectSubscribe:](#)
Reject a presence SUBSCRIBE request which is received from contact.
- (int) - [setPresenceStatus:statusText:](#)
Send a NOTIFY message to contact to notify that presence status is online/offline/changed.

- (long) - [sendSubscription:eventName:](#)
Send a SUBSCRIBE message to remote side.
- (int) - [terminateSubscription:](#)
Terminate the given subscription.
- (BOOL) - [startKeepAwake](#)
Keep VoIP awake in the background. @discussion If you want your application to be able to receive the incoming call while it's running in background, you should call this function in applicationDidEnterBackground.
- (BOOL) - [stopKeepAwake](#)
Keep VoIP awake in the background. @discussion Call this function in applicationWillEnterForeground once your application comes back to foreground.
- (BOOL) - [startAudio:](#)
Start Audio Device. @discussion Call it as AVAudioSessionInterruptionTypeEnded.
- (BOOL) - [stopAudio:](#)
Stop Audio Device. @discussion Call it as AVAudioSessionInterruptionTypeBegan.
- (int) - [enableCallKit:](#)
Enable/disable CallKit(Native Integration).

Properties

- id< [PortSIPEventDelegate](#) > delegate

Detailed Description

PortSIP VoIP SDK functions class.

Author

Copyright (c) 2006-2021 PortSIP Solutions,Inc. All rights reserved.

Version

19

See also

<http://www.PortSIP.com>

PortSIP SDK functions class description.

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

- PortSIPSDK.h

PortSIPVideoRenderView Class Reference

PortSIP VoIP SDK Video Render View class.

```
#import <PortSIPVideoRenderView.h>
```

Inherits UIView.

Instance Methods

- (void) - [initWithVideoRender](#)
Initialize the Video Render view. Render should be initialized before using.
- (void) - [releaseVideoRender](#)
Release the Video Render.
- (void *) - [getVideoRenderView](#)
Don't use this. Just call by SDK.
- (void) - [updateVideoRenderFrame:](#)
Change the Video Render size.

Detailed Description

PortSIP VoIP SDK Video Render View class.

Author

Copyright (c) 2006-2021 PortSIP Solutions, Inc. All rights reserved.

Version

18

See also

<http://www.PortSIP.com>

PortSIP VoIP SDK Video Render View class description.

Method Documentation

- (void) updateVideoRenderFrame: (CGRect) *frameRect*

Change the Video Render size.

Remarks

Example:

```
CGRect rect = videoRenderView.frame;
rect.size.width += 20;
rect.size.height += 20;

videoRenderView.frame = rect;
[videoRenderView setNeedsDisplay:YES];
```



```
CGRect renderRect = [videoRenderView bounds];  
[videoRenderView updateVideoRenderFrame:renderRect];
```

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

- PortSIPVideoRenderView.h

Index

- <PortSIPEventDelegate>, 83
- acceptRefer:referSignaling:
 - Refer functions, 59
- Access SIP message header functions, 44
 - addSipMessageHeader:methodName:msgType:headerName:headerValue:, 45
 - clearAddedSipMessageHeaders, 45
 - clearModifiedSipMessageHeaders, 46
 - getSipMessageHeaderValue:headerName:, 44
 - modifySipMessageHeader:methodName:msgType:headerName:headerValue:, 46
 - removeAddedSipMessageHeader:, 45
 - removeModifiedSipMessageHeader:, 46
- addAudioCodec:
 - Audio and video codecs functions, 34
- Additional settings functions, 37
 - addSupportedMimeType:mimeType:subMimeType:, 43
 - disableCallForward, 41
 - disableSessionTimer, 41
 - enable3GppTags:, 39
 - enableAutoCheckMwi:, 42
 - enableCallbackSignaling:enableReceived:, 40
 - enableCallForward:forwardTo:, 40
 - enableEarlyMedia:, 38
 - enablePriorityIPv6Domain:, 39
 - enableRport:, 38
 - enableSessionTimer:refreshMode:, 41
 - getVersion, 38
 - setAudioSamples:maxPtime:, 42
 - setDoNotDisturb:, 42
 - setKeepAliveTime:, 42
 - setReliableProvisional:, 39
 - setRtpKeepAlive:keepAlivePayloadType:deltaTransmitTimeMS:, 42
 - setRtpPortRange:maximumRtpPort:, 40
 - setSrtpPolicy:, 40
 - setUriUserEncoding:enable:, 39
- addSipMessageHeader:methodName:msgType:headerName:headerValue:
 - Access SIP message header functions, 45
- addSupportedMimeType:mimeType:subMimeType:
 - Additional settings functions, 43
- addVideoCodec:
 - Audio and video codecs functions, 35
- answerCall:videoCall:
 - Call functions, 53
- attendedRefer:replaceSessionId:referTo:
 - Refer functions, 58
- attendedRefer2:replaceSessionId:replaceMethod:target:referTo:
 - Refer functions, 58
- Audio and video codecs functions, 34
 - addAudioCodec:, 34
 - addVideoCodec:, 35
 - isAudioCodecEmpty, 35
 - isVideoCodecEmpty, 35
 - setAudioCodecParameter:parameter:, 36
 - setAudioCodecPayloadType:payloadType:, 35
 - setVideoCodecParameter:parameter:, 36
 - setVideoCodecPayloadType:payloadType:, 35
- Audio and video functions, 47
 - displayLocalVideo:mirror:localVideoWindow:, 50
 - muteMicrophone:, 50
 - muteSpeaker:, 51
 - sendVideo:sendState:, 49
 - setAudioBitrate:codecType:bitrateKbps:, 48
 - setChannelInputVolumeScaling:scaling:, 51
 - setChannelOutputVolumeScaling:scaling:, 51
 - setLoudspeakerStatus:, 51
 - setRemoteScreenWindow:remoteScreenWindow:, 50
 - setRemoteVideoWindow:remoteVideoWindow:, 49
 - setVideoBitrate:bitrateKbps:, 49
 - setVideoDeviceId:, 48
 - setVideoFrameRate:frameRate:, 49
 - setVideoNackStatus:, 50
 - setVideoOrientation:, 48
 - setVideoResolution:height:, 48
- Audio and video stream callback events, 26
 - onAudioRawCallback:audioCallbackMode:data:dataLength:samplingFreqHz:, 26
 - onVideoRawCallback:videoCallbackMode:width:height:data:dataLength:, 26
- Audio Controller, 82
 - enableCallKit:, 82
 - startAudio:, 82
 - stopAudio:, 82
- Audio effect functions, 73
 - enableCNG:, 73
 - enableVAD:, 73
- audioPlayLoopbackTest:
 - Play audio and video files to remote party, 67
- Call events, 10
 - onDialogStateUpdated:BLFDDialogState:BLFDDialogId:BLFDDialogDirection:, 13
 - onInviteAnswered:callerDisplayName:caller:calleeDisplayName:callee:audioCodecs:videoCodecs:existsAudio:existsVideo:sipMessage:, 11
 - onInviteBeginingForward:, 12

- onInviteClosed:sipMessage:, 13
- onInviteConnected:, 12
- onInviteFailure:callerDisplayName:caller: calleeDisplayName:callee:reason:code:sipMessage:, 12
- onInviteIncoming:callerDisplayName:caller:calleeDisplayName:callee:audioCodecs:videoCodecs:existsAudio:existsVideo:sipMessage:, 10
- onInviteRinging:statusText:statusCode:sipMessage:, 11
- onInviteSessionProgress:audioCodecs:videoCodecs:existsEarlyMedia:existsAudio:existsVideo:sipMessage:, 11
- onInviteTrying:, 11
- onInviteUpdated:audioCodecs:videoCodecs:screenCodecs:existsAudio:existsVideo:existsScreen:sipMessage:, 12
- onRemoteHold:, 13
- onRemoteUnHold:audioCodecs:videoCodecs:existsAudio:existsVideo:, 13
- Call functions, 52
 - answerCall:videoCall:, 53
 - call:sendSdp:videoCall:, 53
 - forwardCall:forwardTo:, 55
 - hangUp:, 53
 - hold:, 54
 - muteSession:muteIncomingAudio:muteOutgoingAudio:muteIncomingVideo:muteOutgoingVideo:, 55
 - pickupBLFCall:videoCall:, 55
 - rejectCall:code:, 53
 - sendDtmf:dtmfMethod:code:dtmfDuration:playDtmfTone:, 56
 - unHold:, 54
 - updateCall:enableAudio:enableVideo:, 54
- call:sendSdp:videoCall:
 - Call functions, 53
- clearAddedSipMessageHeaders
 - Access SIP message header functions, 45
- clearModifiedSipMessageHeaders
 - Access SIP message header functions, 46
- Conference functions, 68
 - createAudioConference, 68
 - createVideoConference:videoWidth:videoHeight:layout:, 68
 - joinToConference:, 69
 - removeFromConference:, 69
 - setConferenceVideoWindow:, 69
- createAudioConference
 - Conference functions, 68
- createVideoConference:videoWidth:videoHeight:layout:
 - Conference functions, 68
- disableCallForward
 - Additional settings functions, 41
- disableSessionTimer
 - Additional settings functions, 41
- displayLocalVideo:mirror:localVideoWindow:
 - Audio and video functions, 50
- DTMF events, 18
 - onRecvDtmfTone:tone:, 18
- enable3GppTags:
 - Additional settings functions, 39
- enableAudioQos:
 - RTP and RTCP QOS functions, 71
- enableAudioStreamCallback:enable:callbackMode:
 - RTP packets, audio stream and video stream callback functions, 62
- enableAutoCheckMwi:
 - Additional settings functions, 42
- enableCallbackSignaling:enableReceived:
 - Additional settings functions, 40
- enableCallForward:forwardTo:
 - Additional settings functions, 40
- enableCallKit:
 - Audio Controller, 82
- enableCNG:
 - Audio effect functions, 73
- enableEarlyMedia:
 - Additional settings functions, 38
- enablePriorityIPv6Domain:
 - Additional settings functions, 39
- enableRport:
 - Additional settings functions, 38
- enableRtpCallback:mediaType:mode:
 - RTP packets, audio stream and video stream callback functions, 62
- enableSendPcmStreamToRemote:state:streamSamplesPerSec:
 - Send audio and video stream functions, 60
- enableSendVideoStreamToRemote:state:
 - Send audio and video stream functions, 61
- enableSessionTimer:refreshMode:
 - Additional settings functions, 41
- enableVAD:
 - Audio effect functions, 73
- enableVideoQos:
 - RTP and RTCP QOS functions, 71
- enableVideoStreamCallback:callbackMode:
 - RTP packets, audio stream and video stream callback functions, 63
- forwardCall:forwardTo:
 - Call functions, 55
- getLocalIpAddress:
 - NIC and local IP functions, 33
- getNICNums
 - NIC and local IP functions, 33
- getSipMessageHeaderValue:headerName:
 - Access SIP message header functions, 44
- getStatistics:
 - Media statistics functions, 72
- getVersion
 - Additional settings functions, 38
- hangUp:
 - Call functions, 53
- hold:

- Call functions, 54
- INFO/OPTIONS message events, 19
 - onRecvInfo:, 19
 - onRecvNotifyOfSubscription:notifyMessage:messageData:messageDataLength:, 19
 - onRecvOptions:, 19
- Initialize and register functions, 29
 - initialize:localIP:localSIPPort:loglevel:logPath:maxLine:agent:audioDeviceLayer:videoDeviceLayer:TLSCertificatesRootPath:TLSCipherList:verifyTLSCertificate: dnsServers:, 29
 - refreshRegistration:, 32
 - registerServer:retryTimes:, 31
 - setDisplayNames:, 31
 - setInstanceId:, 30
 - setLicenseKey:, 32
 - setUser:displayName:authName:password:userDomain:SIPServer:SIPServerPort:STUNServer:STUNServerPort:outboundServer:outboundServerPort:, 31
 - unRegisterServer:, 32
- initialize:localIP:localSIPPort:loglevel:logPath:maxLine:agent:audioDeviceLayer:videoDeviceLayer:TLSCertificatesRootPath:TLSCipherList:verifyTLSCertificate: dnsServers:
 - Initialize and register functions, 29
- isAudioCodecEmpty
 - Audio and video codecs functions, 35
- isVideoCodecEmpty
 - Audio and video codecs functions, 35
- joinToConference:
 - Conference functions, 69
- Keep awake functions, 81
 - startKeepAwake, 81
 - stopKeepAwake, 81
- Media statistics functions, 72
 - getStatistics:, 72
- MESSAGE message events, 21
 - onRecvMessage:mimeType:subMimeType:messageData:messageDataLength:, 21
 - onRecvOutOfDialogMessage:from:toDisplayName:to:mimeType:subMimeType:messageData:messageDataLength:sipMessage:, 21
 - onSendMessageFailure:messageId:reason:code:sipMessage:, 22
 - onSendMessageSuccess:messageId:sipMessage:, 22
 - onSendOutOfDialogMessageFailure:from:displayName:from:toDisplayName:to:reason:code:sipMessage:, 22
 - onSendOutOfDialogMessageSuccess:from:displayName:from:toDisplayName:to:sipMessage:, 22
 - onSubscriptionFailure:statusCode:, 23
 - onSubscriptionTerminated:, 23
- modifySipMessageHeader:methodName:msgType:headerName:headerValue:
 - Access SIP message header functions, 46
- muteMicrophone:
 - Audio and video functions, 50
- muteSession:muteIncomingAudio:muteOutgoingAudio:muteIncomingVideo:muteOutgoingVideo:
 - Call functions, 55
- muteSpeaker:
 - Audio and video functions, 51
- MWI events, 17
 - onWaitingFaxMessage:urgentNewMessageCount:urgentOldMessageCount:newMessageCount:oldMessageCount:, 17
 - onWaitingVoiceMessage:urgentNewMessageCount:urgentOldMessageCount:newMessageCount:oldMessageCount:, 17
- NIC and local IP functions, 33
 - getLocalIpAddress:, 33
 - getNICNums, 33
- onACTVTransferFailure:reason:code:
 - Refer events, 15
- onACTVTransferSuccess:
 - Refer events, 15
- onAudioRawCallback:audioCallbackMode:data:dataLength:samplingFreqHz:
 - Audio and video stream callback events, 26
- onDialogStateUpdated:BLFDialogState:BLFDialogId:BLFDialogDirection:
 - Call events, 13
- onInviteAnswered:callerDisplayName:caller:alleeDisplayName:allee:audioCodecs:videoCodecs:existsAudio:existsVideo:sipMessage:
 - Call events, 11
- onInviteBeginningForward:
 - Call events, 12
- onInviteClosed:sipMessage:
 - Call events, 13
- onInviteConnected:
 - Call events, 12
- onInviteFailure:callerDisplayName:caller:alleeDisplayName:allee:reason:code:sipMessage:
 - Call events, 12
- onInviteIncoming:callerDisplayName:caller:alleeDisplayName:allee:audioCodecs:videoCodecs:existsAudio:existsVideo:sipMessage:
 - Call events, 10
- onInviteRinging:statusText:statusCode:sipMessage:
 - Call events, 11
- onInviteSessionProgress:audioCodecs:videoCodecs:existsEarlyMedia:existsAudio:existsVideo:sipMessage:
 - Call events, 11
- onInviteTrying:
 - Call events, 11
- onInviteUpdated:audioCodecs:videoCodecs:screenCodecs:existsAudio:existsVideo:existsScreen:sipMessage:
 - Call events, 12

- onPlayFileFinished:fileName:
 - Play audio and video file finished events, 24
- onPresenceOffline:from:
 - Presence events, 20
- onPresenceOnline:from:stateText:
 - Presence events, 20
- onPresenceRecvSubscribe:fromDisplayName:from:subject:
 - Presence events, 20
- onReceivedRefer:referId:to:from:referSipMessage:
 - Refer events, 14
- onReceivedSignaling:message:
 - Signaling events, 16
- onRecvDtmfTone:tone:
 - DTMF events, 18
- onRecvInfo:
 - INFO/OPTIONS message events, 19
- onRecvMessage:mimeType:subMimeType:messageData:messageDataLength:
 - MESSAGE message events, 21
- onRecvNotifyOfSubscription:notifyMessage:messageData:messageDataLength:
 - INFO/OPTIONS message events, 19
- onRecvOptions:
 - INFO/OPTIONS message events, 19
- onRecvOutOfDialogMessage:from:toDisplayName:to:mimeType:subMimeType:messageData:messageDataLength:sipMessage:
 - MESSAGE message events, 21
- onReferAccepted:
 - Refer events, 14
- onReferRejected:reason:code:
 - Refer events, 14
- onRegisterFailure:statusCode:sipMessage:
 - Register events, 9
- onRegisterSuccess:statusCode:sipMessage:
 - Register events, 9
- onRemoteHold:
 - Call events, 13
- onRemoteUnHold:audioCodecs:videoCodecs:existsAudio:existsVideo:
 - Call events, 13
- onRTPPacketCallback:mediaType:direction:RTPPacket:packetSize:
 - RTP callback events, 25
- onSendingSignaling:message:
 - Signaling events, 16
- onSendMessageFailure:messageId:reason:code:sipMessage:
 - MESSAGE message events, 22
- onSendMessageSuccess:messageId:sipMessage:
 - MESSAGE message events, 22
- onSendOutOfDialogMessageFailure:fromDisplayName:from:toDisplayName:to:reason:code:sipMessage:
 - MESSAGE message events, 22
- onSendOutOfDialogMessageSuccess:fromDisplayName:from:toDisplayName:to:sipMessage:
 - MESSAGE message events, 22
- onStatistics:stat:
 - Play audio and video file finished events, 24
- onSubscriptionFailure:statusCode:
 - MESSAGE message events, 23
- onSubscriptionTerminated:
 - MESSAGE message events, 23
- onTransferRinging:
 - Refer events, 14
- onTransferTrying:
 - Refer events, 14
- onVideoRawCallback:videoCallbackMode:width:height:data:dataLength:
 - Audio and video stream callback events, 26
- onWaitingFaxMessage:urgentNewMessageCount:urgentOldMessageCount:newMessageCount:oldMessageCount:
 - MWI events, 17
- onWaitingVoiceMessage:urgentNewMessageCount:urgentOldMessageCount:newMessageCount:oldMessageCount:
 - MWI events, 17
- outOfDialogRefer:replaceMethod:target:referTo:
 - Refer functions, 58
- pickupBLFCall:videoCall:
 - Call functions, 55
- Play audio and video file finished events, 24
 - onPlayFileFinished:fileName:, 24
 - onStatistics:stat:, 24
- Play audio and video files to remote party, 66
 - audioPlayLoopbackTest:, 67
 - startPlayingFileLocally:loop:playVideoWindow:, 67
 - startPlayingFileToRemote:fileUrl:loop:playAudio:, 66
 - stopPlayingFileLocally, 67
 - stopPlayingFileToRemote:, 66
- PortSIPSDK, 85
- PortSIPVideoRenderView, 94
 - updateVideoRenderFrame:, 94
- Presence events, 20
 - onPresenceOffline:from:, 20
 - onPresenceOnline:from:stateText:, 20
 - onPresenceRecvSubscribe:fromDisplayName:from:subject:, 20
- Presence functions, 77
 - presenceAcceptSubscribe:, 79
 - presenceRejectSubscribe:, 79
 - presenceSubscribe:subject:, 78
 - presenceTerminateSubscribe:, 78
 - sendSubscription:eventName:, 80
 - setDefaultPublicationTime:, 78
 - setDefaultSubscriptionTime:, 78
 - setPresenceMode:, 77

- setPresenceStatus:statusText:, 79
 - terminateSubscription:, 80
- presenceAcceptSubscribe:
 - Presence functions, 79
- presenceRejectSubscribe:
 - Presence functions, 79
- presenceSubscribe:subject:
 - Presence functions, 78
- presenceTerminateSubscribe:
 - Presence functions, 78
- Record functions, 64
 - startRecord:recordFilePath:recordFileName:appendTimeStamp:channels:fileFormat:audioRecordMode:videoRecordMode:, 64
 - stopRecord:, 64
- Refer events, 14
 - onACTVTransferFailure:reason:code:, 15
 - onACTVTransferSuccess:, 15
 - onReceivedRefer:referId:to:from:referSipMessage:, 14
 - onReferAccepted:, 14
 - onReferRejected:reason:code:, 14
 - onTransferRinging:, 14
 - onTransferTrying:, 14
- Refer functions, 57
 - acceptRefer:referSignaling:, 59
 - attendedRefer:replaceSessionId:referTo:, 58
 - attendedRefer2:replaceSessionId:replaceMethod:target:referTo:, 58
 - outOfDialogRefer:replaceMethod:target:referTo:, 58
 - refer:referTo:, 57
 - rejectRefer:, 59
- refer:referTo:
 - Refer functions, 57
- refreshRegistration:
 - Initialize and register functions, 32
- Register events, 9
 - onRegisterFailure:statusCode:sipMessage:, 9
 - onRegisterSuccess:statusCode:sipMessage:, 9
- registerServer:retryTimes:
 - Initialize and register functions, 31
- rejectCall:code:
 - Call functions, 53
- rejectRefer:
 - Refer functions, 59
- removeAddedSipMessageHeader:
 - Access SIP message header functions, 45
- removeFromConference:
 - Conference functions, 69
- removeModifiedSipMessageHeader:
 - Access SIP message header functions, 46
- RTP and RTCP QOS functions, 70
 - enableAudioQos:, 71
 - enableVideoQos:, 71
- setAudioRtcpBandwidth:BitsRR:BitsRS:KBitsAS:, 70
- setVideoMTU:, 71
- setVideoRtcpBandwidth:BitsRR:BitsRS:KBitsAS:, 70
- RTP callback events, 25
 - onRTTPPacketCallback:mediaType:direction:RTTPPacket:packetSize:, 25
- RTP packets, audio stream and video stream
 - callback functions, 62
 - enableAudioStreamCallback:enable:callbackMode:, 62
 - enableRtpCallback:mediaType:mode:, 62
 - enableVideoStreamCallback:callbackMode:, 63
- SDK Callback events, 8
- SDK functions, 28
- Send audio and video stream functions, 60
 - enableSendPcmStreamToRemote:state:streamSamplesPerSec:, 60
 - enableSendVideoStreamToRemote:state:, 61
 - sendPcmStreamToRemote:data:, 60
 - sendVideoStreamToRemote:data:width:height:, 61
- Send OPTIONS/INFO/MESSAGE functions, 74
 - sendInfo:mimeType:subMimeType:infoContents:, 74
 - sendMessage:mimeType:subMimeType:message:messageLength:, 75
 - sendOptions:sdp:, 74
 - sendOutOfDialogMessage:mimeType:subMimeType:isSMS:message:messageLength:, 75
- sendDtmf:dtmfMethod:code:dtmfDuration:playDtmfTone:
 - Call functions, 56
- sendInfo:mimeType:subMimeType:infoContents:
 - Send OPTIONS/INFO/MESSAGE functions, 74
- sendMessage:mimeType:subMimeType:message:messageLength:
 - Send OPTIONS/INFO/MESSAGE functions, 75
- sendOptions:sdp:
 - Send OPTIONS/INFO/MESSAGE functions, 74
- sendOutOfDialogMessage:mimeType:subMimeType:isSMS:message:messageLength:
 - Send OPTIONS/INFO/MESSAGE functions, 75
- sendPcmStreamToRemote:data:
 - Send audio and video stream functions, 60
- sendSubscription:eventName:
 - Presence functions, 80
- sendVideo:sendState:
 - Audio and video functions, 49

sendVideoStreamToRemote:data:width:height:
 Send audio and video stream functions, 61
 setAudioBitrate:codecType:bitrateKbps:
 Audio and video functions, 48
 setAudioCodecParameter:parameter:
 Audio and video codecs functions, 36
 setAudioCodecPayloadType:payloadType:
 Audio and video codecs functions, 35
 setAudioRtcpBandwidth:BitsRR:BitsRS:KBitsAS:
 RTP and RTCP QOS functions, 70
 setAudioSamples:maxPtime:
 Additional settings functions, 42
 setChannelInputVolumeScaling:scaling:
 Audio and video functions, 51
 setChannelOutputVolumeScaling:scaling:
 Audio and video functions, 51
 setConferenceVideoWindow:
 Conference functions, 69
 setDefaultPublicationTime:
 Presence functions, 78
 setDefaultSubscriptionTime:
 Presence functions, 78
 setDisplayName:
 Initialize and register functions, 31
 setDoNotDisturb:
 Additional settings functions, 42
 setInstanceId:
 Initialize and register functions, 30
 setKeepAliveTime:
 Additional settings functions, 42
 setLicenseKey:
 Initialize and register functions, 32
 setLoudspeakerStatus:
 Audio and video functions, 51
 setPresenceMode:
 Presence functions, 77
 setPresenceStatus:statusText:
 Presence functions, 79
 setReliableProvisional:
 Additional settings functions, 39
 setRemoteScreenWindow:remoteScreenWindow:
 Audio and video functions, 50
 setRemoteVideoWindow:remoteVideoWindow:
 Audio and video functions, 49
 setRtpKeepAlive:keepAlivePayloadType:deltaTransmitTimeMS:
 Additional settings functions, 42
 setRtpPortRange:maximumRtpPort:
 Additional settings functions, 40
 setSrtpPolicy:
 Additional settings functions, 40
 setUriUserEncoding:enable:
 Additional settings functions, 39
 setUser:displayName:authName:password:userDomain:SIPServer:SIPServerPort:STUNServer:STUNServerPort:outboundServer:outboundServerPort:
 Initialize and register functions, 31
 setVideoBitrate:bitrateKbps:
 Audio and video functions, 49
 setVideoCodecParameter:parameter:
 Audio and video codecs functions, 36
 setVideoCodecPayloadType:payloadType:
 Audio and video codecs functions, 35
 setVideoDeviceId:
 Audio and video functions, 48
 setVideoFrameRate:frameRate:
 Audio and video functions, 49
 setVideoMTU:
 RTP and RTCP QOS functions, 71
 setVideoNackStatus:
 Audio and video functions, 50
 setVideoOrientation:
 Audio and video functions, 48
 setVideoResolution:height:
 Audio and video functions, 48
 setVideoRtcpBandwidth:BitsRR:BitsRS:KBitsAS:
 RTP and RTCP QOS functions, 70
 Signaling events, 16
 onReceivedSignaling:message:, 16
 onSendingSignaling:message:, 16
 startAudio:
 Audio Controller, 82
 startKeepAwake
 Keep awake functions, 81
 startPlayingFileLocally:loop:playVideoWindow:
 Play audio and video files to remote party, 67
 startPlayingFileToRemote:fileUrl:loop:playAudio:
 Play audio and video files to remote party, 66
 startRecord:recordFilePath:recordFileName:appendTimeStamp:channels:fileFormat:audioRecordMode:videoRecordMode:
 Record functions, 64
 stopAudio:
 Audio Controller, 82
 stopKeepAwake
 Keep awake functions, 81
 stopPlayingFileLocally
 Play audio and video files to remote party, 67
 stopPlayingFileToRemote:
 Play audio and video files to remote party, 66
 stopRecord:
 Record functions, 64
 terminateSubscription:
 Presence functions, 80
 unHold:
 Call functions, 54
 unRegisterServer:

Initialize and register functions, 32
updateCall:enableAudio:enableVideo:
Call functions, 54

updateVideoRenderFrame:
PortSIPVideoRenderView, 94