PortSIP VoIP SDK Manual for Mac

Version 11.2.2
4/4/2015

Table of Contents

Welcome to the PortSIP VoIP SDK 3
Module Index 6
Hierarchical Index 7
Class Index 8
Module Documentation 9
SDK functions 9
Initialize and register functions 9
NIC and local IP functions 12
Audio and video codecs functions 12
Additional setting functions 15
Access SIP message header functions 22
Audio and video functions 24
Call functions 28
Refer functions 32
Send audio and video stream functions 34
RTP packets, Audio stream and video stream callback functions 36
Record functions 38
Play audio and video file to remoe functions 39
Conference functions 41
RTP and RTCP QOS functions 43
RTP statistics functions 44
Audio effect functions 47
Send OPTIONS/INFO/MESSAGE functions 48
Presence functions 50
Device Manage functions. 52
SDK Callback events 56
Register events 56
Call events 57
Refer events 60
Signaling events 61
MWI events 62
DTMF events 63
INFO/OPTIONS message events 64
Presence events 64
MESSAGE message events 65
Play audio and video file finished events 67
RTP callback events 68
Audio and video stream callback events 69
Class Documentation 71
<PortSIPEventDelegate> 71
PortSIPSDK 73
PortSIPVideoRenderView 80
Index 82

Welcome to the PortSIP VoIP SDK

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

The award-winning PortSIP VoIP SDK is a powerful and highly versatile set of tools to dramatically accelerate SIP application development. It includes a suite of stacks, SDKs, Sample projects. Each one enables developers to combine all the necessary components to create an ideal development environment for every application's specific needs.

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

Changes in this release

This release is a major upgrade, see Release Notes for more information.

Getting Started

You can download the PortSIP VoIP SDK Sample projects at our Website, the samples include for VC++, C#, VB.NET, Delphi XE, XCode(for iOS and Mac OS), Eclipse(Java, for Android), the sample project source code is provided(not include SDK source code). The sample projects demonstrate how to create a SIP application base on our SDK, powerful, easy and quick.

Contents

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

SDK User Manual

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

Web Site

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

Background

Read the Overview to help you understand what PortSIP is about and to help in educating your organization about PortSIP.

Support

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

Machine Requirements

Development using the PortSIP VoIP/IMS SDK for Mac requires an Intel-based Macintosh running Snow Leopard (OS X 10.8 or higher), Xcode 5.0 and above

Frequently Asked Questions

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

```
 All sample projects of the PortSIP VoIP SDK can be download to test at:
                  http://www.PortSIP.com/downloads.html
                  http://www.PortSIP.com/voipsdk.html.
```

2. How to compile the sample project?

```
      1. Download the sample projects from PortSIP website.
      2. Extract the .zip file.
      3. Open the project by your xcode:
      4. Compile the sample project directly, the trial version SDK allows 2-3 minutes
conversation.
```

3. Create a new project base on PortSIP VoIP SDK

```
  1). Download the Sample project and evaluation SDK and extract it to a directory
   2). Run the Xcode and create a new OS X Cocoa Applicatiion Project
  3). Drag and drop PortSIPSDK.framework from Finder to XCode->Frameworks.
  4). Copying Frameworks Files While Building a Product:
      Click Build Phases at the top of the project editor
      Choose Editor > Add Build Phase > Add Copy Files Build Phase
      Destination specify Frameworks,Click the Add button (+) to select PortSIPSDK.framework
to copy and click Add
  5). Add the code in .h file to import the SDK, example:
      #import <PortSIPSDK/PortSIPSDK.h>
  6). Inherit the interface PortSIPEventDelegate to process the callback events.
  7). Initialize sdk, Example:
      mPortSIPSDK = [[PortSIPSDK alloc] init];
      mPortSIPSDK.delegate = self;
  8). More details please read the Sample project source code.
```

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

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

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

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

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

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

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

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

5. Does the SDK is thread safe?

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

6. Does the SDK support native 64 bits?

```
Yes, the SDK support 64 bits.
```

Module Index

Modules

Here is a list of all modules:

SDK functions 9
- Initialize and register functions 9
- NIC and local IP functions 12
- Audio and video codecs functions....12
- Additional setting functions 15
- Access SIP message header functions....22
- Audio and video functions 24
- Call functions 28
- Refer functions....32
- Send audio and video stream functions....34
- RTP packets, Audio stream and video stream callback functions....36
- Record functions 38
- Play audio and video file to remoe functions....39
- Conference functions 41
- RTP and RTCP QOS functions....43
- RTP statistics functions....44
- Audio effect functions....47
- Send OPTIONS/INFO/MESSAGE functions 48
- Presence functions....50
- Device Manage functions.....52

SDK Callback events 56
- Register events 56
- Call events....57
- Refer events....60
- Signaling events 61
- MWI events....62
- DTMF events 63
- INFO/OPTIONS message events....64
- Presence events 64
- MESSAGE message events 65
- Play audio and video file finished events....67
- RTP callback events....68
- Audio and video stream callback events 69

Hierarchical Index

Class Hierarchy

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

<NSObject>
- <PortSIPEventDelegate> 71
- PortSIPSDK 73

NSView
- PortSIPVideoRenderView 80

Class Index

Class List

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

<PortSIPEventDelegate> (PortSIP SDK Callback events Delegate) 71

PortSIPSDK (PortSIP VoIP SDK functions class) 73

PortSIPVideoRenderView (PortSIP VoIP SDK Video Render View class) 80

Module Documentation

SDK functions

Modules

- Initialize and register functions
- NIC and local IP functions
- Audio and video codecs functions
- Additional setting functions
- Access SIP message header functions
- Audio and video functions
- Call functions
- Refer functions
- Send audio and video stream functions
- RTP packets, Audio stream and video stream callback functions
- Record functions
- Play audio and video file to remoe functions
- Conference functions
- RTP and RTCP QOS functions
- RTP statistics functions
- Audio effect functions
- Send OPTIONS/INFO/MESSAGE functions
- Presence functions
- Device Manage functions.

Detailed Description

SDK functions

Initialize and register functions

Functions

- (int) - PortSIPSDK::initialize:loglevel:logPath:maxLine:agent:audioDeviceLayer:videoDeviceLayer:
  Initialize the SDK.
- (void) - PortSIPSDK::unInitialize
  Un-initialize the SDK and release resources.
- (int) - PortSIPSDK::setUser:displayName:authName:password:localIP:localSIPPort:userDomain:SIPServer:SIPServerPort:STUNServer:STUNServerPort:outboundServer:outboundServerPort:
  Set user account info.
- (int) - PortSIPSDK::registerServer:retryTimes:
  Register to SIP proxy server(login to server)
- (int) - PortSIPSDK::unRegisterServer
  Un-register from the SIP proxy server.
- (int) - PortSIPSDK::setLicenseKey:
  Set the license key, must called before setUser function.

Detailed Description

Initialize and register functions

Function Documentation

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

Initialize the SDK.

Parameters:

transport	Transport for SIP signaling.TRANSPORT_PERS is the PortSIP private transport for anti the SIP blocking, it must using with the PERS.
logLevel	Set the application log level, the SDK generate the "PortSIP_Log_datatime.log" file if the log enabled.
logFilePath	The log file path, the path(folder) MUST is exists.
maxCallLines	In theory support unlimited lines just depends on the device capability, for SIP client recommend less than 1 - 100;
sipAgent	The User-Agent header to insert in SIP messages.
audioDeviceLayer	0 = Use OS default device 1 = Set to 1 to use the virtual audio device if the no sound device installed.
videoDeviceLayer	0 = Use OS default device 1 = Set to 1 to use the virtual video device if no camera installed.

Returns:

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

- (int) setUser: (NSString *) *userName* displayName: (NSString *) *displayName* authName: (NSString *) *authName* password: (NSString *) *password* localIP: (NSString *) *localIP* localSIPPort: (int) *localSIPPort* 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:

userName	Account(User name) of the SIP, usually provided by an IP-Telephony service provider.
displayName	The display name of user, you can set it as your like, such as "James Kend". It's optional.
authName	Authorization user name (usually equals the username).
password	The password of user, it's optional.
localIP	The local computer IP address to bind (for example: 192.168.1.108), it will be

	using for send and receive SIP message and RTP packet. If pass this IP as the IPv6 format then the SDK using IPv6. If you want the SDK choose correct network interface(IP) automatically, please pass the "0.0.0.0"(for IPv4) or "::"(for IPv6).
localSIPPort	The SIP message transport listener port(for example: 5060).
userDomain	User domain; this parameter is optional that allow pass a empty string if you are not use domain.
sipServer	SIP proxy server IP or domain(for example: xx.xxx.xx.x or sip.xxx.com).
sipServerPort	Port of the SIP proxy server, (for example: 5060).
stunServer	Stun server, use for NAT traversal, it's optional and can be pass empty string to disable STUN.
stunServerPort	STUN server port,it will be ignored if the outboundServer is empty.
outboundServer	Outbound proxy server(for example: sip.domain.com), it's optional that allow pass a empty string if not use outbound server.
outboundServerPort	Outbound proxy server port, it will be ignored if the outboundServer is empty.

Returns:

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

- (int) registerServer: (int) *expires* retryTimes: (int) *retryTimes*

Register to SIP proxy server(login to server)

Parameters:

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

Returns:

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

- (int) unRegisterServer

Un-register from the SIP proxy server.

Returns:

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

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

Set the license key, must called before setUser function.

Parameters:

key	The SDK license key, please purchase from PortSIP

Returns:

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

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, the return value is NIC numbers >= 0. If the function fails, the return value is a specific error code.

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

Get the local IP address by Network Interface Card index.

Parameters:

index	The IP address index, for example, the PC has two NICs, we want to obtain the second NIC IP, then set this parameter 1. The first NIC IP index is 0.

Returns:

The buffer that to receives the IP.

Audio and video codecs functions

Functions

- (int) - PortSIPSDK::addAudioCodec:

Enable an audio codec, it will be appears in SDP.

- (int) - PortSIPSDK::addVideoCodec:
  Enable a video codec, it will be appears in SDP.
- (BOOL) - PortSIPSDK::isAudioCodecEmpty
  Detect enabled audio codecs is empty or not.
- (BOOL) - PortSIPSDK::isVideoCodecEmpty
  Detect 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 be appears in SDP.

Parameters:

codecType	Audio codec type.

Returns:

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

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

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

Parameters:

codecType	Video codec type.

Returns:

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

- (BOOL) isAudioCodecEmpty

Detect enabled audio codecs is empty or not.

Returns:

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

- (BOOL) isVideoCodecEmpty

Detect enabled video codecs is empty or not.

Returns:

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

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

Set the RTP payload type for dynamic audio codec.

Parameters:

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

Returns:

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

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

Set the RTP payload type for dynamic Video codec.

Parameters:

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

Returns:

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

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

Set the codec parameter for audio codec.

Parameters:

codecType	Audio codec type, defined in the PortSIPTypes file.
parameter	The parameter in string format.

Returns:

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

Remarks:

Example:

```
[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:

codecType	Video codec type, defined in the PortSIPTypes file.
parameter	The parameter in string format.

Returns:

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

Remarks:

Example:

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

Additional setting functions

Functions

- (int) - PortSIPSDK::setDisplayName:
  Set user display name.
- (int) - PortSIPSDK::getVersion:minorVersion:
  Get the current version number of the SDK.
- (int) - PortSIPSDK::enableReliableProvisional:
  Enable/disable PRACK.
- (int) - PortSIPSDK::enable3GppTags:
  Enable/disable the 3Gpp tags, include "ims.icsi.mmtel" and "g.3gpp.smsip".
- (void) - PortSIPSDK::enableCallbackSendingSignaling:
  Enable/disable callback the sending SIP messages.
- (int) - PortSIPSDK::setSrtpPolicy:

Set the SRTP policy.

- (int) - PortSIPSDK::setRtpPortRange:maximumRtpAudioPort:minimumRtpVideoPort:maximumRtpVideoPort:
  Set the RTP ports range for audio and video streaming.
- (int) - PortSIPSDK::setRtcpPortRange:maximumRtcpAudioPort:minimumRtcpVideoPort:maximumRtcpVideoPort:
  Set the RTCP ports range for audio and video streaming.
- (int) - PortSIPSDK::enableCallForward:forwardTo:
  Enable call forward.
- (int) - PortSIPSDK::disableCallForward
  Disable the call forward, the SDK is not forward any incoming call after this function is called.
- (int) - PortSIPSDK::enableSessionTimer:refreshMode:
  Allows to periodically refresh Session Initiation Protocol (SIP) sessions by sending repeated INVITE requests.
- (int) - PortSIPSDK::disableSessionTimer
  Disable the session timer.
- (void) - PortSIPSDK::setDoNotDisturb:
  Enable the "Do not disturb" to enable/disable.
- (int) - PortSIPSDK::detectMwi
  Use to obtain the MWI status.
- (int) - PortSIPSDK::enableCheckMwi:
  Allows enable/disable the check MWI(Message Waiting Indication).
- (int) - PortSIPSDK::setRtpKeepAlive:keepAlivePayloadType:deltaTransmitTimeMS:
  Enable or disable send RTP keep-alive packet during the call is established.
- (int) - PortSIPSDK::setKeepAliveTime:
  Enable or disable send SIP keep-alive packet.
- (int) - PortSIPSDK::setAudioSamples:maxPtime:
  Set the audio capture sample.
- (int) - PortSIPSDK::addSupportedMimeType:mimeType:subMimeType:
  Set the SDK receive the SIP message that include special mime type.

Detailed Description

Function Documentation

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

Set user display name.

Parameters:

displayName	The display name.

Returns:

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

- (int) getVersion: (int *) *majorVersion* minorVersion: (int *) *minorVersion*

Get the current version number of the SDK.

Parameters:

majorVersion	Return the major version number.
minorVersion	Return the minor version number.

Returns:

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

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

Enable/disable PRACK.

Parameters:

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

Returns:

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

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

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

Parameters:

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

Returns:

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

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

Enable/disable callback the sending SIP messages.

Parameters:

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

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

Set the SRTP policy.

Parameters:

srtpPolicy	The SRTP policy.

Returns:

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

- (int) setRtpPortRange: (int) *minimumRtpAudioPort* maximumRtpAudioPort: (int) *maximumRtpAudioPort* minimumRtpVideoPort: (int) *minimumRtpVideoPort* maximumRtpVideoPort: (int) *maximumRtpVideoPort*

Set the RTP ports range for audio and video streaming.

Parameters:

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

Returns:

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

Remarks:

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

- (int) setRtcpPortRange: (int) *minimumRtcpAudioPort* maximumRtcpAudioPort: (int) *maximumRtcpAudioPort* minimumRtcpVideoPort: (int) *minimumRtcpVideoPort* maximumRtcpVideoPort: (int) *maximumRtcpVideoPort*

Set the RTCP ports range for audio and video streaming.

Parameters:

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

Returns:

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

Remarks:

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

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

Enable call forward.

Parameters:

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

Returns:

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

- (int) disableCallForward

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

Returns:

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

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

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

Parameters:

timerSeconds	The value of the refresh interval in seconds. Minimum requires 90 seconds.
refreshMode	Allow set the session refresh by UAC or UAS: SESSION_REFERESH_UAC or SESSION_REFERESH_UAS;

Returns:

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

Remarks:

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

- (int) disableSessionTimer

Disable the session timer.

Returns:

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

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

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

Parameters:

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

- (int) detectMwi

Use to obtain the MWI status.

Returns:

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

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

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

Parameters:

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

Returns:

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

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

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

Parameters:

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

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

Returns:

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

- (int) setKeepAliveTime: (int) *keepAliveTime*

Enable or disable send SIP keep-alive packet.

Parameters:

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

Returns:

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

- (int) setAudioSamples: (int) *ptime* maxPtime: (int) *maxPtime*

Set the audio capture sample.

Parameters:

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

Returns:

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

Remarks:

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

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

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

Parameters:

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

Returns:

If the function succeeds, the return value is 0. If the function fails, the return value is a specific error code.   Default, the PortSIP VoIP SDK support these media types(mime types) that in the below incoming SIP messages:

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

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

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

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

```
[myVoIPSdk addSupportedMimeType:@"NOTIFY" mimeType:@"application"
subMimeType:@"media_control+xml"];
```

About the mime type details, please visit this website:
http://www.iana.org/assignments/media-types/

Access SIP message header functions

Functions

- (NSString *) - PortSIPSDK::getExtensionHeaderValue:headerName:
  Access the SIP header of SIP message.
- (int) - PortSIPSDK::addExtensionHeader:headerValue:
  Add the extension header(custom header) into every outgoing SIP message.
- (int) - PortSIPSDK::clearAddExtensionHeaders
  Clear the added extension headers(custom headers)
- (int) - PortSIPSDK::modifyHeaderValue:headerValue:
  Modify the special SIP header value for every outgoing SIP message.
- (int) - PortSIPSDK::clearModifyHeaders
  Clear the modify headers value, no longer modify every outgoing SIP message header values.

Detailed Description

Function Documentation

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

Access the SIP header of SIP message.

Parameters:

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

Returns:

If the function succeeds, the return value is headerValue. If the function fails, the return value is nil.

Remarks:

When got a SIP message in the onReceivedSignaling callback event, and want to get SIP message header value, use getExtensionHeaderValue to do it:

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

- (int) addExtensionHeader: (NSString *) *headerName* headerValue: (NSString *) *headerValue*

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

Parameters:

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

Returns:

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

- (int) clearAddExtensionHeaders

Clear the added extension headers(custom headers)

Returns:

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

Remarks:

```
Example, we have added two custom headers into every outgoing SIP message and want remove them.
[myVoIPSdk addExtensionHeader:@"Blling" headerValue:@"usd100.00"];
[myVoIPSdk addExtensionHeader:@"ServiceId" headerValue:@"8873456"];
[myVoIPSdk clearAddextensionHeaders];
```

- (int) modifyHeaderValue: (NSString *) *headerName* headerValue: (NSString *) *headerValue*

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

Parameters:

headerName	The SIP header name which will be modify it's value.

headerValue	The heaver value want to modify.

Returns:

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

- (int) clearModifyHeaders

Clear the modify headers value, no longer modify every outgoing SIP message header values.

Returns:

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

Remarks:

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

```
[myVoIPSdk modifyHeaderValue:@"Expires" headerValue:@"1000");
[myVoIPSdk modifyHeaderValue:@"User-Agent", headerValue:@"MyTest Softphone 1.0");
[myVoIPSdk cleaModifyHeaders];
```

Audio and video functions

Functions

- (int) - PortSIPSDK::setVideoDeviceId:
  Set the video device that will use for video call.
- (int) - PortSIPSDK::setVideoResolution:
  Set the video capture resolution.
- (int) - PortSIPSDK::setVideoBitrate:
  Set the video bit rate.
- (int) - PortSIPSDK::setVideoFrameRate:
  Set the video frame rate.
- (int) - PortSIPSDK::sendVideo:sendState:
  Send the video to remote side.
- (int) - PortSIPSDK::setVideoOrientation:
  Changing the orientation of the video.
- (void) - PortSIPSDK::setLocalVideoWindow:
  Set the the window that using to display the local video image.
- (int) - PortSIPSDK::setRemoteVideoWindow:remoteVideoWindow:
  Set the window for a session that using to display the received remote video image.
- (int) - PortSIPSDK::displayLocalVideo:
  Start/stop to display the local video image.
- (int) - PortSIPSDK::setVideoNackStatus:
  Enable/disable the NACK feature(rfc6642) which help to improve the video quatliy.
- (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::setAudioDeviceId:outputDeviceId:
  Set the audio device that will use for audio call.
- (void) - PortSIPSDK::getDynamicVolumeLevel:microphoneVolume:
  Obtain the dynamic microphone volume level from current call.

Detailed Description

Function Documentation

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

Set the video device that will use for video call.

Parameters:

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

Returns:

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

- (int) setVideoResolution: (VIDEO_RESOLUTION) *resolution*

Set the video capture resolution.

Parameters:

resolution	Video resolution, defined in PortSIPType file. Note: Some cameras don't support SVGA and XVGA, 720P, please read your camera manual.

Returns:

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

- (int) setVideoBitrate: (int) *bitrateKbps*

Set the video bit rate.

Parameters:

bitrateKbps	The video bit rate in KBPS.

Returns:

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

- (int) setVideoFrameRate: (int) *frameRate*

Set the video frame rate.

Parameters:

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

Returns:

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

Remarks:

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

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

Send the video to remote side.

Parameters:

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

Returns:

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

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

Changing the orientation of the video.

Parameters:

rotation	The video rotation that you want to set(0,90,180,270).

Returns:

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

- (void) setLocalVideoWindow: (PortSIPVideoRenderView *) *localVideoWindow*

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

Parameters:

localVideoWindow	The PortSIPVideoRenderView to display local video image from camera.

- (int) setRemoteVideoWindow: (long) *sessionId* remoteVideoWindow: (PortSIPVideoRenderView *) *remoteVideoWindow*

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

Parameters:

sessionId	The session ID of the call.
remoteVideoWindow	The PortSIPVideoRenderView to display received remote video image.

Returns:

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

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

Start/stop to display the local video image.

Parameters:

state	state Set to true to display local video iamge.

Returns:

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

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

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

Parameters:

state	state Set to true to enable.

Returns:

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

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

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

Parameters:

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

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

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

Parameters:

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

- (int) setAudioDeviceId: (int) *inputDeviceId* outputDeviceId: (int) *outputDeviceId*

Set the audio device that will use for audio call.

Parameters:

inputDeviceId	Device ID(index) for audio record.(Microphone).
outputDeviceId	Device ID(index) for audio playback(Speaker).

Returns:

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

- (void) getDynamicVolumeLevel: (int *) *speakerVolume* microphoneVolume: (int *) *microphoneVolume*

Obtain the dynamic microphone volume level from current call.

Parameters:

speakerVolume	Return the dynamic speaker volume by this parameter, the range is 0 - 9.
microphoneVolume	Return the dynamic microphone volume by this parameter, the range is 0 - 9.

Remarks:

Usually set a timer to call this function to refresh the volume level indicator.

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:
  To 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 call to another one when received the incoming call.
- (int) - PortSIPSDK::sendDtmf:dtmfMethod:code:dtmfDration:playDtmfTone:
  Send DTMF tone.

Detailed Description

Function Documentation

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

Make a call.

Parameters:

callee	The callee, it can be name only or full SIP URI, for example: user001 or sip:user001@sip.iptel.org or sip:user002@sip.yourdomain.com:5068
sendSdp	If set to false then the outgoing call doesn't include the SDP in INVITE message.
videoCall	If set the true and at least one video codec was added, then the outgoing call include the video codec into SDP.

Returns:

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

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

rejectCall Reject the incoming call.

Parameters:

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

Returns:

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

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

hangUp Hang up the call.

Parameters:

sessionId	Session ID of the call.

Returns:

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

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

answerCall Answer the incoming call.

Parameters:

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

Returns:

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

- (int) updateCall: (long) *sessionId* enableAudio: (BOOL) *enableAudio* enableVideo: (BOOL) *enableVideo*

Use the re-INVITE to update the established call.

Parameters:

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

Returns:

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

Remarks:

Example usage:

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

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

Example 2: Remove video stream from currently conversation.

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

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

To place a call on hold.

Parameters:

sessionId	The session ID of call.

Returns:

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

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

Take off hold.

Parameters:

sessionId	The session ID of call.

Returns:

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

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

Mute the specified session audio or video.

Parameters:

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

Returns:

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

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

Forward call to another one when received the incoming call.

Parameters:

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

Returns:

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

- (int) sendDtmf: (long) *sessionId* dtmfMethod: (DTMF_METHOD) *dtmfMethod* code: (int) *code* dtmfDration: (int) *dtmfDuration* playDtmfTone: (BOOL) *playDtmfTone*

Send DTMF tone.

Parameters:

sessionId	The session ID of the call.
dtmfMethod	Support send DTMF tone with two methods: DTMF_RFC2833 and DTMF_INFO. The DTMF_RFC2833 is recommend.
code	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:

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

Returns:

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

Refer functions

Functions

- (int) - PortSIPSDK::refer:referTo:
  Refer the currently call to another one.

- (int) - PortSIPSDK::attendedRefer:replaceSessionId:referTo:
  Make an attended refer.
- (long) - PortSIPSDK::acceptRefer:referSignaling:
  Accept the REFER request, a new call will be make if called this function, usuall 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 currently call to another one.

Parameters:

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

Returns:

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

Remarks:

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

You can download the demo AVI at: "http://www.portsip.com/downloads/video/blindtransfer.rar", use the Windows Media Player to play the AVI file after extracted, it will shows how to do the transfer.

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

Make an attended refer.

Parameters:

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

Returns:

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

Remarks:

Please read the sample project source code to got more details. Or download the demo AVI at:"http://www.portsip.com/downloads/video/blindtransfer.rar"
use the Windows Media Player to play the AVI file after extracted, it will shows how to do the transfer.

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

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

Parameters:

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

Returns:

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

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

Reject the REFER request.

Parameters:

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

Returns:

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

Send audio and video stream functions

Functions

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

Detailed Description

Function Documentation

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

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

Parameters:

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

Returns:

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

Remarks:

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

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

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

Parameters:

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

Returns:

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

Remarks:

Usually we should use it like below:

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

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

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

Parameters:

sessionId	The session ID of call.

state	Set to true to enable the send stream, false to disable.

Returns:

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

- (int) sendVideoStreamToRemote: (long) *sessionId* data: (NSData *) *data* width: (int) *width* height: (int) *height*

Send the video stream to remote.

Parameters:

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

Returns:

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

Remarks:

Send the video stream in i420 from another source to instead of video device capture(camera).
Before called 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::setRtpCallback:
  Set the RTP callbacks to allow access the sending 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) setRtpCallback: (BOOL) *enable*

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

Parameters:

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

Returns:

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

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

Enable/disable the audio stream callback.

Parameters:

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

Type	Description
AUDIOSTREAM_LOCAL_MIX	Callback the audio stream from microphone for all channels.
AUDIOSTREAM_LOCAL_PER_CHANNEL	Callback the audio stream from microphone for one channel base on the given sessionId.
AUDIOSTREAM_REMOTE_MIX	Callback the received audio stream that mixed including all channels.
AUDIOSTREAM_REMOTE_PER_CHANNEL	Callback the received audio stream for one channel base on the given sessionId.

Returns:

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

Remarks:

the onAudioRawCallback event will be triggered if the callback is enabled.

- (int) enableVideoStreamCallback: (long) *sessionId* callbackMode: (VIDEOSTREAM_CALLBACK_MODE) *callbackMode*

Enable/disable the video stream callback.

Parameters:

sessionId	The session ID of call.
callbackMode	The video stream callback mode.

Mode	Description
VIDEOSTREAM_NONE	Disable video stream callback.
VIDEOSTREAM_LOCAL	Local video stream callback.
VIDEOSTREAM_REMOTE	Remote video stream callback.

VIDEOSTREAM_BOTH	Both of local and remote video stream callback.

Returns:

If the function succeeds, the return value is 0. If the function fails, the return value is 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:audioFileFormat:audioRecordMode:aviFileCodecType:videoRecordMode:
  Start record the call.
- (int) - PortSIPSDK::stopRecord:
  Stop record.

Detailed Description

Function Documentation

- (int) startRecord: (long) *sessionId* recordFilePath: (NSString *) *recordFilePath* recordFileName: (NSString *) *recordFileName* appendTimeStamp: (BOOL) *appendTimeStamp* audioFileFormat: (AUDIO_FILE_FORMAT) *audioFileFormat* audioRecordMode: (RECORD_MODE) *audioRecordMode* aviFileCodecType: (VIDEOCODEC_TYPE) *aviFileCodecType* videoRecordMode: (RECORD_MODE) *videoRecordMode*

Start record the call.

Parameters:

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

	received and send.

Returns:

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

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

Stop record.

Parameters:

sessionId	The session ID of call conversation.

Returns:

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

Play audio and video file to remoe functions

Functions

- (int) - PortSIPSDK::playVideoFileToRemote:aviFile:loop:playAudio:
  Play an AVI file to remote party.
- (int) - PortSIPSDK::stopPlayVideoFileToRemote:
  Stop play video file to remote side.
- (int) - PortSIPSDK::playAudioFileToRemote:filename:fileSamplesPerSec:loop:
  Play an wave file to remote party.
- (int) - PortSIPSDK::stopPlayAudioFileToRemote:
  Stop play wave file to remote side.
- (int) - PortSIPSDK::playAudioFileToRemoteAsBackground:filename:fileSamplesPerSec:
  Play an wave file to remote party as conversation background sound.
- (int) - PortSIPSDK::stopPlayAudioFileToRemoteAsBackground:
  Stop play an wave file to remote party as conversation background sound.

Detailed Description

Function Documentation

- (int) playVideoFileToRemote: (long) *sessionId* aviFile: (NSString *) *aviFile* loop: (BOOL) *loop* playAudio: (BOOL) *playAudio*

Play an AVI file to remote party.

Parameters:

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

Returns:

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

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

Stop play video file to remote side.

Parameters:

sessionId	Session ID of the call.

Returns:

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

- (int) playAudioFileToRemote: (long) *sessionId* filename: (NSString *) *filename* fileSamplesPerSec: (int) *fileSamplesPerSec* loop: (BOOL) *loop*

Play an wave file to remote party.

Parameters:

sessionId	Session ID of the call.
filename	The file full path name, such as "/test.wav".
fileSamplesPerSec	The wave file sample in seconds, should be 8000 or 16000 or 32000.
loop	Set to false to stop play audio file when it is end. Set to true to play it as repeat.

Returns:

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

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

Stop play wave file to remote side.

Parameters:

sessionId	Session ID of the call.

Returns:

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

- (int) playAudioFileToRemoteAsBackground: (long) *sessionId* filename: (NSString *) *filename* fileSamplesPerSec: (int) *fileSamplesPerSec*

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

Parameters:

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

Returns:

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

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

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

Parameters:

sessionId	Session ID of the call.

Returns:

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

Conference functions

Functions

- (int) - PortSIPSDK::createConference:videoResolution:displayLocalVideo:
  Create a conference. It's failures if the exists conference isn't destroy yet.
- (void) - PortSIPSDK::destroyConference
  Destroy the exist conference.
- (int) - PortSIPSDK::setConferenceVideoWindow:
  Set the window for a conference that using to display the received remote video image.
- (int) - PortSIPSDK::joinToConference:
  Join a session into exist conference, if the call is in hold, please un-hold first.
- (int) - PortSIPSDK::removeFromConference:
  Remove a session from an exist conference.

Detailed Description

Function Documentation

- (int) createConference: (PortSIPVideoRenderView *) *conferenceVideoWindow* videoResolution: (VIDEO_RESOLUTION) *videoResolution* displayLocalVideo: (BOOL) *displayLocalVideoInConference*

Create a conference. It's failures if the exists conference isn't destroy yet.

Parameters:

conferenceVideoWindow	The PortSIPVideoRenderView which using to display the conference video.
videoResolution	The conference video resolution.
displayLocalVideoInConference	Display the local video on video window or not.

Returns:

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

- (int) setConferenceVideoWindow: (PortSIPVideoRenderView *) *conferenceVideoWindow*

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

Parameters:

conferenceVideoWindow	The PortSIPVideoRenderView which using to display the conference video.

Returns:

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

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

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

Parameters:

sessionId	Session ID of the call.

Returns:

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

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

Remove a session from an exist conference.

Parameters:

sessionId	Session ID of the call.

Returns:

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

RTP and RTCP QOS functions

Functions

- (int) - 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::setAudioQos:DSCPValue:priority:
  Set the DSCP(differentiated services code point) value of QoS(Quality of Service) for audio channel.
- (int) - PortSIPSDK::setVideoQos:DSCPValue:
  Set the DSCP(differentiated services code point) value of QoS(Quality of Service) for video channel.

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:

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

Returns:

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

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

Set the video RTCP bandwidth parameters as the RFC3556.

Parameters:

sessionId	The session ID of call conversation.

BitsRR	The bits for the RR parameter.
BitsRS	The bits for the RS parameter.
KBitsAS	The Kbits for the AS parameter.

Returns:

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

- (int) setAudioQos: (BOOL) *enable* DSCPValue: (int) *DSCPValue* priority: (int) *priority*

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

Parameters:

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

Returns:

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

- (int) setVideoQos: (BOOL) *enable* DSCPValue: (int) *DSCPValue*

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

Parameters:

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

Returns:

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

RTP statistics functions

Functions

- (int) - PortSIPSDK::getNetworkStatistics:currentBufferSize:preferredBufferSize:currentPacketLossRate:currentDiscardRate:currentExpandRate:currentPreemptiveRate:currentAccelerateRate:
  Get the "in-call" statistics. The statistics are reset after the query.
- (int) - PortSIPSDK::getAudioRtpStatistics:averageJitterMs:maxJitterMs:discardedPackets:
  Obtain the RTP statisics of audio channel.

- (int) - PortSIPSDK::getAudioRtcpStatistics:bytesSent:packetsSent:bytesReceived:packetsReceived:sendFractionLost:sendCumulativeLost:recvFractionLost:recvCumulativeLost:
  Obtain the RTCP statisics of audio channel.
- (int) - PortSIPSDK::getVideoRtpStatistics:bytesSent:packetsSent:bytesReceived:packetsReceived:
  Obtain the RTP statisics of video.

Detailed Description

Function Documentation

- (int) getNetworkStatistics: (long) *sessionId* currentBufferSize: (int *) *currentBufferSize* preferredBufferSize: (int *) *preferredBufferSize* currentPacketLossRate: (int *) *currentPacketLossRate* currentDiscardRate: (int *) *currentDiscardRate* currentExpandRate: (int *) *currentExpandRate* currentPreemptiveRate: (int *) *currentPreemptiveRate* currentAccelerateRate: (int *) *currentAccelerateRate*

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

Parameters:

sessionId	The session ID of call conversation.
currentBufferSize	Preferred (optimal) buffer size in ms.
preferredBufferSize	Preferred (optimal) buffer size in ms.
currentPacketLossRate	Loss rate (network + late) in percent.
currentDiscardRate	Fraction of synthesized speech inserted through pre-emptive expansion .
currentExpandRate	Fraction of synthesized speech inserted through pre-emptive expansion .
currentPreemptiveRate	Fraction of synthesized speech inserted through pre-emptive expansion.
currentAccelerateRate	Fraction of data removed through acceleration .

Returns:

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

- (int) getAudioRtpStatistics: (long) *sessionId* averageJitterMs: (int *) *averageJitterMs* maxJitterMs: (int *) *maxJitterMs* discardedPackets: (int *) *discardedPackets*

Obtain the RTP statisics of audio channel.

Parameters:

sessionId	The session ID of call conversation.

averageJitterMs	Short-time average jitter (in milliseconds).
maxJitterMs	Maximum short-time jitter (in milliseconds).
discardedPackets	The number of discarded packets on a channel during the call.

Returns:

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

- (int) getAudioRtcpStatistics: (long) *sessionId* bytesSent: (int *) *bytesSent* packetsSent: (int *) *packetsSent* bytesReceived: (int *) *bytesReceived* packetsReceived: (int *) *packetsReceived* sendFractionLost: (int *) *sendFractionLost* sendCumulativeLost: (int *) *sendCumulativeLost* recvFractionLost: (int *) *recvFractionLost* recvCumulativeLost: (int *) *recvCumulativeLost*

Obtain the RTCP statisics of audio channel.

Parameters:

sessionId	The session ID of call conversation.
bytesSent	The number of sent bytes.
packetsSent	The number of sent packets.
bytesReceived	The number of received bytes.
packetsReceived	The number of received packets.
sendFractionLost	Fraction of sent lost in percent.
sendCumulativeLost	The number of sent cumulative lost packet.
recvFractionLost	Fraction of received lost in percent.
recvCumulativeLost	The number of received cumulative lost packets.

Returns:

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

- (int) getVideoRtpStatistics: (long) *sessionId* bytesSent: (int *) *bytesSent* packetsSent: (int *) *packetsSent* bytesReceived: (int *) *bytesReceived* packetsReceived: (int *) *packetsReceived*

Obtain the RTP statisics of video.

Parameters:

sessionId	The session ID of call conversation.
bytesSent	The number of sent bytes.
packetsSent	The number of sent packets.
bytesReceived	The number of received bytes.
packetsReceived	The number of received packets.

Returns:

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

Audio effect functions

Functions

- (void) - PortSIPSDK::enableVAD:
  Enable/disable Voice Activity Detection(VAD).
- (void) - PortSIPSDK::enableAEC:
  Enable/disable AEC (Acoustic Echo Cancellation).
- (void) - PortSIPSDK::enableCNG:
  Enable/disable Comfort Noise Generator(CNG).
- (void) - PortSIPSDK::enableAGC:
  Enable/disable Automatic Gain Control(AGC).
- (void) - PortSIPSDK::enableANS:
  Enable/disable Audio Noise Suppression(ANS).

Detailed Description

Function Documentation

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

Enable/disable Voice Activity Detection(VAD).

Parameters:

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

- (void) enableAEC: (EC_MODES) *state*

Enable/disable AEC (Acoustic Echo Cancellation).

Parameters:

state	AEC type, default is EC_NONE.

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

Enable/disable Comfort Noise Generator(CNG).

Parameters:

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

- (void) enableAGC: (AGC_MODES) *state*

Enable/disable Automatic Gain Control(AGC).

Parameters:

state	AGC type, default is AGC_NONE.

- (void) enableANS: (NS_MODES) *state*

Enable/disable Audio Noise Suppression(ANS).

Parameters:

state	NS type, default is NS_NONE.

Send OPTIONS/INFO/MESSAGE functions

Functions

- (int) - PortSIPSDK::sendOptions:sdp:
  Send OPTIONS message.
- (int) - PortSIPSDK::sendInfo:mimeType:subMimeType:infoContents:
  Send a 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:message:messageLength:
  Send a out of dialog MESSAGE message to remote side.

Detailed Description

Function Documentation

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

Send OPTIONS message.

Parameters:

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

Returns:

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

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

Send a INFO message to remote side in dialog.

Parameters:

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

Returns:

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

- (long) 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:

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

Returns:

If the function succeeds, the return value is a message ID allows track the message send state in onSendMessageSuccess and onSendMessageFailure. If the function fails, the return value is a specific error code less than 0.

Remarks:

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

```
[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* message: (NSData *) *message* messageLength: (int) *messageLength*

Send a out of dialog MESSAGE message to remote side.

Parameters:

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

Returns:

If the function succeeds, the return value is a message ID allows track the message send state in onSendOutOfMessageSuccess and onSendOutOfMessageFailure. If the function fails, the return value is a specific error code less than 0.

Remarks:

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

```
[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" message:data
messageLength:dataLen];
```

Presence functions

Functions

- (int) - PortSIPSDK::presenceSubscribeContact:subject:
  Send a SUBSCRIBE message for presence to a contact.
- (int) - PortSIPSDK::presenceAcceptSubscribe:
  Accept the presence SUBSCRIBE request which received from contact.
- (int) - PortSIPSDK::presenceRejectSubscribe:
  Reject a presence SUBSCRIBE request which received from contact.
- (int) - PortSIPSDK::presenceOnline:statusText:
  Send a NOTIFY message to contact to notify that presence status is online/changed.
- (int) - PortSIPSDK::presenceOffline:
  Send a NOTIFY message to contact to notify that presence status is offline.

Detailed Description

Function Documentation

- (int) presenceSubscribeContact: (NSString *) *contact* subject: (NSString *) *subject*

Send a SUBSCRIBE message for presence to a contact.

Parameters:

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

Returns:

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

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

Accept the presence SUBSCRIBE request which received from contact.

Parameters:

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

Returns:

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

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

Reject a presence SUBSCRIBE request which received from contact.

Parameters:

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

Returns:

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

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

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

Parameters:

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

Returns:

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

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

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

Parameters:

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

Returns:

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

Device Manage functions.

Functions

- (int) - PortSIPSDK::getNumOfVideoCaptureDevices
  Gets the number of available capture devices.
- (int) - PortSIPSDK::getVideoCaptureDeviceName:uniqueId:deviceName:
  Gets the name of a specific video capture device given by an index.
- (int) - PortSIPSDK::getNumOfRecordingDevices
  Gets the number of audio devices available for audio recording.
- (int) - PortSIPSDK::getNumOfPlayoutDevices
  Gets the number of audio devices available for audio playout.
- (NSString *) - PortSIPSDK::getRecordingDeviceName:
  Gets the name of a specific recording device given by an index.
- (NSString *) - PortSIPSDK::getPlayoutDeviceName:
  Gets the name of a specific playout device given by an index.
- (int) - PortSIPSDK::setSpeakerVolume:
  Set the speaker volume level,.
- (int) - PortSIPSDK::getSpeakerVolume
  Gets the speaker volume level.
- (int) - PortSIPSDK::setSystemOutputMute:
  Mutes the speaker device completely in the OS.
- (BOOL) - PortSIPSDK::getSystemOutputMute
  Retrieves the output device mute state in the operating system.
- (int) - PortSIPSDK::setMicVolume:
  Sets the microphone volume level.
- (int) - PortSIPSDK::getMicVolume
  Retrieves the current microphone volume.
- (int) - PortSIPSDK::setSystemInputMute:
  Mute the microphone input device completely in the OS.
- (BOOL) - PortSIPSDK::getSystemInputMute
  Gets the mute state of the input device in the operating system.
- (void) - PortSIPSDK::audioPlayLoopbackTest:
  Use to do the audio device loop back test.

Detailed Description

Function Documentation

- (int) getNumOfVideoCaptureDevices

Gets the number of available capture devices.

Returns:

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

- (int) getVideoCaptureDeviceName: (int) *index* uniqueId: (NSString **) *uniqueIdUTF8* deviceName: (NSString **) *deviceNameUTF8*

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

Parameters:

index	Device index (0, 1, 2, ..., N-1), where N is given by getNumOfVideoCaptureDevices (). Also -1 is a valid value and will return the name of the default capture device.
uniqueIdUTF8	Unique identifier of the capture device.
deviceNameUTF8	A character buffer to which the device name will be copied as a null-terminated string in UTF8 format.

Returns:

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

- (int) getNumOfRecordingDevices

Gets the number of audio devices available for audio recording.

Returns:

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

- (int) getNumOfPlayoutDevices

Gets the number of audio devices available for audio playout.

Returns:

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

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

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

Parameters:

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

Returns:

A NSString to which the device name will be copied as a null-terminated string in UTF8 format.

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

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

Parameters:

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

Returns:

A NSString to which the device name will be copied as a null-terminated string in UTF8 format.

- (int) setSpeakerVolume: (int) *volume*

Set the speaker volume level,.

Parameters:

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

Returns:

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

- (int) getSpeakerVolume

Gets the speaker volume level.

Returns:

If the function succeeds, the return value is speaker volume, valid range is 0 - 255. If the function fails, the return value is a specific error code.

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

Mutes the speaker device completely in the OS.

Parameters:

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

Returns:

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

- (BOOL) getSystemOutputMute

Retrieves the output device mute state in the operating system.

Returns:

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

- (int) setMicVolume: (int) *volume*

Sets the microphone volume level.

Parameters:

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

Returns:

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

- (int) getMicVolume

Retrieves the current microphone volume.

Returns:

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

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

Mute the microphone input device completely in the OS.

Parameters:

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

Returns:

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

- (BOOL) getSystemInputMute

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

Returns:

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

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

Use to do the audio device loop back test.

Parameters:

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

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:
- (void) - <PortSIPEventDelegate>::onRegisterFailure:statusCode:

Detailed Description

Register events

Function Documentation

- (void) onRegisterSuccess: (char *) *statusText* statusCode: (int) *statusCode*

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

Parameters:

statusText	The status text.
statusCode	The status code.

- (void) onRegisterFailure: (char *) *statusText* statusCode: (int) *statusCode*

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

Parameters:

statusText	The status text.
statusCode	The status code.

Call events

Functions

- (void) - <PortSIPEventDelegate>::onInviteIncoming:callerDisplayName:caller:calleeDisplayName:callee:audioCodecs:videoCodecs:existsAudio:existsVideo:
- (void) - <PortSIPEventDelegate>::onInviteTrying:
- (void) - <PortSIPEventDelegate>::onInviteSessionProgress:audioCodecs:videoCodecs:existsEarlyMedia:existsAudio:existsVideo:
- (void) - <PortSIPEventDelegate>::onInviteRinging:statusText:statusCode:
- (void) - <PortSIPEventDelegate>::onInviteAnswered:callerDisplayName:caller:calleeDisplayName:callee:audioCodecs:videoCodecs:existsAudio:existsVideo:
- (void) - <PortSIPEventDelegate>::onInviteFailure:reason:code:
- (void) - <PortSIPEventDelegate>::onInviteUpdated:audioCodecs:videoCodecs:existsAudio:existsVideo:
- (void) - <PortSIPEventDelegate>::onInviteConnected:
- (void) - <PortSIPEventDelegate>::onInviteBeginingForward:
- (void) - <PortSIPEventDelegate>::onInviteClosed:
- (void) - <PortSIPEventDelegate>::onRemoteHold:
- (void) - <PortSIPEventDelegate>::onRemoteUnHold:audioCodecs:videoCodecs:existsAudio:existsVideo:

Detailed Description

Function Documentation

- (void) onInviteIncoming: (long) *sessionId* callerDisplayName: (char *) *callerDisplayName* caller: (char *) *caller* calleeDisplayName: (char *) *calleeDisplayName* callee: (char *) *callee*

audioCodecs: (char *) *audioCodecs* videoCodecs: (char *) *videoCodecs* existsAudio: (BOOL) *existsAudio* existsVideo: (BOOL) *existsVideo*

When the call is coming, this event was triggered.

Parameters:

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

- (void) onInviteTrying: (long) *sessionId*`[optional]`

If the outgoing call was processing, this event triggered.

Parameters:

sessionId	The session ID of the call.

- (void) onInviteSessionProgress: (long) *sessionId* audioCodecs: (char *) *audioCodecs* videoCodecs: (char *) *videoCodecs* existsEarlyMedia: (BOOL) *existsEarlyMedia* existsAudio: (BOOL) *existsAudio* existsVideo: (BOOL) *existsVideo*`[optional]`

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

Parameters:

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

- (void) onInviteRinging: (long) *sessionId* statusText: (char *) *statusText* statusCode: (int) *statusCode*`[optional]`

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

Parameters:

sessionId	The session ID of the call.
statusText	The status text.
statusCode	The status code.

- (void) onInviteAnswered: (long) *sessionId* callerDisplayName: (char *) *callerDisplayName* caller: (char *) *caller* calleeDisplayName: (char *) *calleeDisplayName* callee: (char *) *callee* audioCodecs: (char *) *audioCodecs* videoCodecs: (char *) *videoCodecs* existsAudio: (BOOL) *existsAudio* existsVideo: (BOOL) *existsVideo*`[optional]`

If the remote party was answered the call, this event triggered.

Parameters:

sessionId	The session ID of the call.

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

- (void) onInviteFailure: (long) *sessionId* reason: (char *) *reason* code: (int) *code*`[optional]`

If the outgoing call is fails, this event triggered.

Parameters:

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

- (void) onInviteUpdated: (long) *sessionId* audioCodecs: (char *) *audioCodecs* videoCodecs: (char *) *videoCodecs* existsAudio: (BOOL) *existsAudio* existsVideo: (BOOL) *existsVideo*`[optional]`

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

Parameters:

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

- (void) onInviteConnected: (long) *sessionId*`[optional]`

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

Parameters:

sessionId	The session ID of the call.

- (void) onInviteBeginingForward: (char *) *forwardTo*`[optional]`

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

Parameters:

forwardTo	The forward target SIP URI.

- (void) onInviteClosed: (long) *sessionId*`[optional]`

This event is triggered once remote side close the call.

Parameters:

sessionId	The session ID of the call.

- (void) onRemoteHold: (long) *sessionId*`[optional]`

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

Parameters:

sessionId	The session ID of the call.

- (void) onRemoteUnHold: (long) *sessionId* audioCodecs: (char *) *audioCodecs* videoCodecs: (char *) *videoCodecs* existsAudio: (BOOL) *existsAudio* existsVideo: (BOOL) *existsVideo*`[optional]`

If the remote side was un-hold the call, this event triggered

Parameters:

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

Refer events

Functions

- (void) - <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: (char *) *to* from: (char *) *from* referSipMessage: (char *) *referSipMessage*

This event will be triggered once received a REFER message.

Parameters:

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

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

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

Parameters:

sessionId	The session ID of the call.

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

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

Parameters:

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

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

When the refer call is processing, this event trigged.

Parameters:

sessionId	The session ID of the call.

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

When the refer call is ringing, this event trigged.

Parameters:

sessionId	The session ID of the call.

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

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

Parameters:

sessionId	The session ID of the call.

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

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

Parameters:

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

Signaling events

Functions

- (void) - <PortSIPEventDelegate>::onReceivedSignaling:message:
- (void) - <PortSIPEventDelegate>::onSendingSignaling:message:

Detailed Description

Function Documentation

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

This event will be triggered when received a SIP message.

Parameters:

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

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

This event will be triggered when sent a SIP message.

Parameters:

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

MWI events

Functions

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

Detailed Description

Function Documentation

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

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

Parameters:

messageAccount	Voice message account
urgentNewMessag	Urgent new message count.

eCount	
urgentOldMessage Count	Urgent old message count.
newMessageCount	New message count.
oldMessageCount	Old message count.

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

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

Parameters:

messageAccount	Fax message account
urgentNewMessag eCount	Urgent new message count.
urgentOldMessage Count	Urgent old message count.
newMessageCount	New message count.
oldMessageCount	Old message count.

DTMF events

Functions

- (void) - <PortSIPEventDelegate>::onRecvDtmfTone:tone:

Detailed Description

Function Documentation

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

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

Parameters:

sessionId	The session ID of the call.
tone	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:

Detailed Description

Function Documentation

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

This event will be triggered when received the OPTIONS message.

Parameters:

optionsMessage	The received whole OPTIONS message in text format.

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

This event will be triggered when received the INFO message.

Parameters:

infoMessage	The received whole INFO message in text format.

Presence events

Functions

- (void) - <PortSIPEventDelegate>::onPresenceRecvSubscribe:fromDisplayName:from:subject:
- (void) - <PortSIPEventDelegate>::onPresenceOnline:from:stateText:
- (void) - <PortSIPEventDelegate>::onPresenceOffline:from:

Detailed Description

Function Documentation

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

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

Parameters:

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

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

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

Parameters:

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

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

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

Parameters:

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

MESSAGE message events

Functions

- (void) - <PortSIPEventDelegate>::onRecvMessage:mimeType:subMimeType:messageData:messageDataLength:
- (void) - <PortSIPEventDelegate>::onRecvOutOfDialogMessage:from:toDisplayName:to:mimeType:subMimeType:messageData:messageDataLength:
- (void) - <PortSIPEventDelegate>::onSendMessageSuccess:messageId:
- (void) - <PortSIPEventDelegate>::onSendMessageFailure:messageId:reason:code:
- (void) - <PortSIPEventDelegate>::onSendOutOfDialogMessageSuccess:fromDisplayName:from:toDisplayName:to:
- (void) - <PortSIPEventDelegate>::onSendOutOfDialogMessageFailure:fromDisplayName:from:toDisplayName:to:reason:code:

Detailed Description

Function Documentation

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

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

Parameters:

sessionId	The session ID of the call.
mimeType	The message mime type.
subMimeType	The message sub mime type.
messageData	The received message body, it's can be text or binary data.
messageDataLength	The length of "messageData".

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

This event will be triggered when received a MESSAGE message out of dialog, for example: pager message.

Parameters:

fromDisplayName	The display name of sender.
from	The message sender.
toDisplayName	The display name of receiver.
to	The receiver.
mimeType	The message mime type.
subMimeType	The message sub mime type.
messageData	The received message body, it's can be text or binary data.
messageDataLength	The length of "messageData".

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

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

Parameters:

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

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

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

Parameters:

sessionId	The session ID of the call.

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

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

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

Parameters:

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

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

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

Parameters:

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

Play audio and video file finished events

Functions

- (void) - <PortSIPEventDelegate>::onPlayAudioFileFinished:fileName:
- (void) - <PortSIPEventDelegate>::onPlayVideoFileFinished:

Detailed Description

Function Documentation

- (void) onPlayAudioFileFinished: (long) *sessionId* fileName: (char *) *fileName*

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

Parameters:

sessionId	The session ID of the call.
fileName	The play file name.

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

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

Parameters:

sessionId	The session ID of the call.

RTP callback events

Functions

- (void) - <PortSIPEventDelegate>::onReceivedRTPPacket:isAudio:RTPPacket:packetSize:
- (void) - <PortSIPEventDelegate>::onSendingRTPPacket:isAudio:RTPPacket:packetSize:

Detailed Description

Function Documentation

- (void) onReceivedRTPPacket: (long) *sessionId* isAudio: (BOOL) *isAudio* RTPPacket: (unsigned char *) *RTPPacket* packetSize: (int) *packetSize*

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

Parameters:

sessionId	The session ID of the call.
isAudio	If the received RTP packet is of audio, this parameter is true, otherwise false.
RTPPacket	The memory of whole RTP packet.
packetSize	The size of received RTP Packet.

Note:

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

- (void) onSendingRTPPacket: (long) *sessionId* isAudio: (BOOL) *isAudio* RTPPacket: (unsigned char *) *RTPPacket* packetSize: (int) *packetSize*

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

Parameters:

sessionId	The session ID of the call.
isAudio	If the received RTP packet is of audio, this parameter is true, otherwise false.
RTPPacket	The memory of whole RTP packet.

packetSize	The size of received RTP Packet.

Note:

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

Audio and video stream callback events

Functions

- (void) - <PortSIPEventDelegate>::onAudioRawCallback:audioCallbackMode:data:dataLength:samplingFreqHz:
- (void) - <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 received the audio packets if called enableAudioStreamCallback function.

Parameters:

sessionId	The session ID of the call.
audioCallbackMode	TThe type which pasdded in enableAudioStreamCallback function.
data	The memory of audio stream, it's PCM format.
dataLength	The data size.
samplingFreqHz	The audio stream sample in HZ, for example, it's 8000 or 16000.

Note:

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

- (void) 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:

sessionId	The session ID of the call.

videoCallbackMode	The type which pasdded in enableVideoStreamCallback function.
width	The width of video image.
height	The height of video image.
data	The memory of video stream, it's YUV420 format, YV12.
dataLength	The data size.

Note:

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

Class Documentation

<PortSIPEventDelegate> Protocol Reference

PortSIP SDK Callback events Delegate.

`#import <PortSIPEventDelegate.h>`

Inherits <NSObject>.

Instance Methods

- (void) - onRegisterSuccess:statusCode:
- (void) - onRegisterFailure:statusCode:
- (void) - onInviteIncoming:callerDisplayName:caller:calleeDisplayName:callee:audioCodecs:videoCodecs:existsAudio:existsVideo:
- (void) - onInviteTrying:
- (void) - onInviteSessionProgress:audioCodecs:videoCodecs:existsEarlyMedia:existsAudio:existsVideo:
- (void) - onInviteRinging:statusText:statusCode:
- (void) - onInviteAnswered:callerDisplayName:caller:calleeDisplayName:callee:audioCodecs:videoCodecs:existsAudio:existsVideo:
- (void) - onInviteFailure:reason:code:
- (void) - onInviteUpdated:audioCodecs:videoCodecs:existsAudio:existsVideo:
- (void) - onInviteConnected:
- (void) - onInviteBeginingForward:
- (void) - onInviteClosed:
- (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) - 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:
- (void) - onSendMessageSuccess:messageId:
- (void) - onSendMessageFailure:messageId:reason:code:
- (void) - onSendOutOfDialogMessageSuccess:fromDisplayName:from:toDisplayName:to:
- (void) - onSendOutOfDialogMessageFailure:fromDisplayName:from:toDisplayName:to:reason:code:
- (void) - onPlayAudioFileFinished:fileName:
- (void) - onPlayVideoFileFinished:
- (void) - onReceivedRTPPacket:isAudio:RTPPacket:packetSize:
- (void) - onSendingRTPPacket:isAudio:RTPPacket:packetSize:
- (void) - onAudioRawCallback:audioCallbackMode:data:dataLength:samplingFreqHz:
- (void) - onVideoRawCallback:videoCallbackMode:width:height:data:dataLength:

Detailed Description

PortSIP SDK Callback events Delegate.

Author:

Copyright (c) 2006-2014 PortSIP Solutions,Inc. All rights reserved.

Version:

11.2

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:loglevel:logPath:maxLine:agent:audioDeviceLayer:videoDeviceLayer:
  Initialize the SDK.
- (void) - unInitialize
  Un-initialize the SDK and release resources.
- (int) - setUser:displayName:authName:password:localIP:localSIPPort:userDomain:SIPServer:SIPServerPort:STUNServer:STUNServerPort:outboundServer:outboundServerPort:
  Set user account info.
- (int) - registerServer:retryTimes:
  Register to SIP proxy server(login to server)
- (int) - unRegisterServer
  Un-register from the SIP proxy server.
- (int) - setLicenseKey:
  Set the license key, must 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 be appears in SDP.
- (int) - addVideoCodec:
  Enable a video codec, it will be appears in SDP.
- (BOOL) - isAudioCodecEmpty
  Detect enabled audio codecs is empty or not.
- (BOOL) - isVideoCodecEmpty
  Detect 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.
- (int) - setDisplayName:

Set user display name.

- (int) - getVersion:minorVersion:
  Get the current version number of the SDK.
- (int) - enableReliableProvisional:
  Enable/disable PRACK.
- (int) - enable3GppTags:
  Enable/disable the 3Gpp tags, include "ims.icsi.mmtel" and "g.3gpp.smsip".
- (void) - enableCallbackSendingSignaling:
  Enable/disable callback the sending SIP messages.
- (int) - setSrtpPolicy:
  Set the SRTP policy.
- (int) - setRtpPortRange:maximumRtpAudioPort:minimumRtpVideoPort:maximumRtpVideoPort:
  Set the RTP ports range for audio and video streaming.
- (int) - setRtcpPortRange:maximumRtcpAudioPort:minimumRtcpVideoPort:maximumRtcpVideoPort:
  Set the RTCP ports range for audio and video streaming.
- (int) - enableCallForward:forwardTo:
  Enable call forward.
- (int) - disableCallForward
  Disable the call forward, the SDK is not forward any incoming call after this function is called.
- (int) - enableSessionTimer:refreshMode:
  Allows to periodically refresh Session Initiation Protocol (SIP) sessions by sending repeated INVITE requests.
- (int) - disableSessionTimer
  Disable the session timer.
- (void) - setDoNotDisturb:
  Enable the "Do not disturb" to enable/disable.
- (int) - detectMwi
  Use to obtain the MWI status.
- (int) - enableCheckMwi:
  Allows enable/disable the check MWI(Message Waiting Indication).
- (int) - setRtpKeepAlive:keepAlivePayloadType:deltaTransmitTimeMS:
  Enable or disable send RTP keep-alive packet during the call is established.
- (int) - setKeepAliveTime:
  Enable or disable send SIP keep-alive packet.
- (int) - setAudioSamples:maxPtime:
  Set the audio capture sample.
- (int) - addSupportedMimeType:mimeType:subMimeType:
  Set the SDK receive the SIP message that include special mime type.
- (NSString *) - getExtensionHeaderValue:headerName:
  Access the SIP header of SIP message.
- (int) - addExtensionHeader:headerValue:
  Add the extension header(custom header) into every outgoing SIP message.
- (int) - clearAddExtensionHeaders
  Clear the added extension headers(custom headers)
- (int) - modifyHeaderValue:headerValue:
  Modify the special SIP header value for every outgoing SIP message.
- (int) - clearModifyHeaders

Clear the modify headers value, no longer modify every outgoing SIP message header values.

- (int) - setVideoDeviceId:
  Set the video device that will use for video call.
- (int) - setVideoResolution:
  Set the video capture resolution.
- (int) - setVideoBitrate:
  Set the video bit rate.
- (int) - setVideoFrameRate:
  Set the video frame rate.
- (int) - sendVideo:sendState:
  Send the video to remote side.
- (int) - setVideoOrientation:
  Changing the orientation of the video.
- (void) - setLocalVideoWindow:
  Set the the window that using to display the local video image.
- (int) - setRemoteVideoWindow:remoteVideoWindow:
  Set the window for a session that using to display the received remote video image.
- (int) - displayLocalVideo:
  Start/stop to display the local video image.
- (int) - setVideoNackStatus:
  Enable/disable the NACK feature(rfc6642) which help to improve the video quatliy.
- (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) - setAudioDeviceId:outputDeviceId:
  Set the audio device that will use for audio call.
- (void) - getDynamicVolumeLevel:microphoneVolume:
  Obtain the dynamic microphone volume level from current call.
- (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:
  To 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 call to another one when received the incoming call.

- (int) - sendDtmf:dtmfMethod:code:dtmfDration:playDtmfTone:
  Send DTMF tone.
- (int) - referTo:
  Refer the currently call to another one.

- (int) - attendedRefer:replaceSessionId:referTo:
  Make an attended refer.
- (long) - acceptRefer:referSignaling:
  Accept the REFER request, a new call will be make if called this function, usuall called after onReceivedRefer callback event.
- (int) - rejectRefer:
  Reject the REFER request.
- (int) - enableSendPcmStreamToRemote:state:streamSamplesPerSec:
  Enable the SDK send PCM stream data to remote side from another source to instread of microphone.
- (int) - sendPcmStreamToRemote:data:
  Send the audio stream in PCM format from another source to instead of audio device capture(microphone).
- (int) - enableSendVideoStreamToRemote:state:
  Enable the SDK send video stream data to remote side from another source to instead of camera.
- (int) - sendVideoStreamToRemote:data:width:height:
  Send the video stream to remote.
- (int) - setRtpCallback:
  Set the RTP callbacks to allow access the sending 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:audioFileFormat:audioRecordMode:aviFileCodecType:videoRecordMode:
  Start record the call.
- (int) - stopRecord:
  Stop record.
- (int) - playVideoFileToRemote:aviFile:loop:playAudio:
  Play an AVI file to remote party.
- (int) - stopPlayVideoFileToRemote:
  Stop play video file to remote side.
- (int) - playAudioFileToRemote:filename:fileSamplesPerSec:loop:
  Play an wave file to remote party.
- (int) - stopPlayAudioFileToRemote:
  Stop play wave file to remote side.
- (int) - playAudioFileToRemoteAsBackground:filename:fileSamplesPerSec:
  Play an wave file to remote party as conversation background sound.
- (int) - stopPlayAudioFileToRemoteAsBackground:
  Stop play an wave file to remote party as conversation background sound.
- (int) - createConference:videoResolution:displayLocalVideo:

Create a conference. It's failures if the exists conference isn't destroy yet.

- (void) - destroyConference
  Destroy the exist conference.
- (int) - setConferenceVideoWindow:
  Set the window for a conference that using to display the received remote video image.
- (int) - joinToConference:
  Join a session into exist conference, if the call is in hold, please un-hold first.
- (int) - removeFromConference:
  Remove a session from an exist 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) - setAudioQos:DSCPValue:priority:
  Set the DSCP(differentiated services code point) value of QoS(Quality of Service) for audio channel.
- (int) - setVideoQos:DSCPValue:
  Set the DSCP(differentiated services code point) value of QoS(Quality of Service) for video channel.
- (int) - getNetworkStatistics:currentBufferSize:preferredBufferSize:currentPacketLossRate:currentDiscardRate:currentExpandRate:currentPreemptiveRate:currentAccelerateRate:
  Get the "in-call" statistics. The statistics are reset after the query.
- (int) - getAudioRtpStatistics:averageJitterMs:maxJitterMs:discardedPackets:
  Obtain the RTP statisics of audio channel.
- (int) - getAudioRtcpStatistics:bytesSent:packetsSent:bytesReceived:packetsReceived:sendFractionLost:sendCumulativeLost:recvFractionLost:recvCumulativeLost:
  Obtain the RTCP statisics of audio channel.
- (int) - getVideoRtpStatistics:bytesSent:packetsSent:bytesReceived:packetsReceived:
  Obtain the RTP statisics of video.
- (void) - enableVAD:
  Enable/disable Voice Activity Detection(VAD).
- (void) - enableAEC:
  Enable/disable AEC (Acoustic Echo Cancellation).
- (void) - enableCNG:
  Enable/disable Comfort Noise Generator(CNG).
- (void) - enableAGC:
  Enable/disable Automatic Gain Control(AGC).
- (void) - enableANS:
  Enable/disable Audio Noise Suppression(ANS).
- (int) - sendOptions:sdp:
  Send OPTIONS message.
- (int) - sendInfo:mimeType:subMimeType:infoContents:
  Send a 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:message:messageLength:
  Send a out of dialog MESSAGE message to remote side.

- (int) - presenceSubscribeContact:subject:
  Send a SUBSCRIBE message for presence to a contact.
- (int) - presenceAcceptSubscribe:
  Accept the presence SUBSCRIBE request which received from contact.
- (int) - presenceRejectSubscribe:
  Reject a presence SUBSCRIBE request which received from contact.
- (int) - presenceOnline:statusText:
  Send a NOTIFY message to contact to notify that presence status is online/changed.
- (int) - presenceOffline:
  Send a NOTIFY message to contact to notify that presence status is offline.
- (int) - getNumOfVideoCaptureDevices
  Gets the number of available capture devices.
- (int) - getVideoCaptureDeviceName:uniqueId:deviceName:
  Gets the name of a specific video capture device given by an index.
- (int) - getNumOfRecordingDevices
  Gets the number of audio devices available for audio recording.
- (int) - getNumOfPlayoutDevices
  Gets the number of audio devices available for audio playout.
- (NSString *) - getRecordingDeviceName:
  Gets the name of a specific recording device given by an index.
- (NSString *) - getPlayoutDeviceName:
  Gets the name of a specific playout device given by an index.
- (int) - setSpeakerVolume:
  Set the speaker volume level,.
- (int) - getSpeakerVolume
  Gets the speaker volume level.
- (int) - setSystemOutputMute:
  Mutes the speaker device completely in the OS.
- (BOOL) - getSystemOutputMute
  Retrieves the output device mute state in the operating system.
- (int) - setMicVolume:
  Sets the microphone volume level.
- (int) - getMicVolume
  Retrieves the current microphone volume.
- (int) - setSystemInputMute:
  Mute the microphone input device completely in the OS.
- (BOOL) - getSystemInputMute
  Gets the mute state of the input device in the operating system.
- (void) - audioPlayLoopbackTest:
  Use to do the audio device loop back test.

Properties

- id< PortSIPEventDelegate > **delegate**

Detailed Description

PortSIP VoIP SDK functions class.

Author:

Version:

11.2.2

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 NSView.

Instance Methods

- (void) - initVideoRender
  Initialize the Video Render view. shoud Initialize render before use.
- (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:

Version:

11.2.2

See also:

http://www.PortSIP.com

PortSIP VoIP SDK Video Render View class description.

Method Documentation

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

change the Video Render size.

Remarks:

Example:

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

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

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

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

- PortSIPVideoRenderView.h

Index

<PortSIPEventDelegate>, 70
acceptRefer:referSignaling:
    Refer functions, 33
Access SIP message header functions, 21
    addExtensionHeader:headerValue:, 22
    clearAddExtensionHeaders, 22
    clearModifyHeaders, 23
    getExtensionHeaderValue:headerName:, 21
    modifyHeaderValue:headerValue:, 22
addAudioCodec:
    Audio and video codecs functions, 12
addExtensionHeader:headerValue:
    Access SIP message header functions, 22
Additional setting functions, 14
    addSupportedMimeType:mimeType:subMimeType:, 20
    detectMwi, 19
    disableCallForward, 18
    disableSessionTimer, 19
    enable3GppTags:, 16
    enableCallbackSendingSignaling:, 16
    enableCallForward:forwardTo:, 18
    enableCheckMwi:, 19
    enableReliableProvisional:, 16
    enableSessionTimer:refreshMode:, 18
    getVersion:minorVersion:, 16
    setAudioSamples:maxPtime:, 20
    setDisplayName:, 15
    setDoNotDisturb:, 19
    setKeepAliveTime:, 20
    setRtcpPortRange:maximumRtcpAudioPort:minimumRtcpVideoPort:maximumRtcpVideoPort:, 17
    setRtpKeepAlive:keepAlivePayloadType:deltaTransmitTimeMS:, 19
    setRtpPortRange:maximumRtpAudioPort:minimumRtpVideoPort:maximumRtpVideoPort:, 17
    setSrtpPolicy:, 17
addSupportedMimeType:mimeType:subMimeType:
    Additional setting functions, 20
addVideoCodec:
    Audio and video codecs functions, 12
answerCall:videoCall:
    Call functions, 29
attendedRefer:replaceSessionId:referTo:
    Refer functions, 32
Audio and video codecs functions, 11
    addAudioCodec:, 12
    addVideoCodec:, 12
    isAudioCodecEmpty, 13
    isVideoCodecEmpty, 13
    setAudioCodecParameter:parameter:, 13
    setAudioCodecPayloadType:payloadType:, 13
    setVideoCodecParameter:parameter:, 14
    setVideoCodecPayloadType:payloadType:, 13
Audio and video functions, 23
    displayLocalVideo:, 26
    getDynamicVolumeLevel:microphoneVolume:, 27
    muteMicrophone:, 26
    muteSpeaker:, 26
    sendVideo:sendState:, 25
    setAudioDeviceId:outputDeviceId:, 27
    setLocalVideoWindow:, 25
    setRemoteVideoWindow:remoteVideoWindow:, 25
    setVideoBitrate:, 24
    setVideoDeviceId:, 24
    setVideoFrameRate:, 25
    setVideoNackStatus:, 26
    setVideoOrientation:, 25
    setVideoResolution:, 24
Audio and video stream callback events, 68
    onAudioRawCallback:audioCallbackMode:data:dataLength:samplingFreqHz:, 68
    onVideoRawCallback:videoCallbackMode:width:height:data:dataLength:, 68
Audio effect functions, 46
    enableAEC:, 46
    enableAGC:, 47
    enableANS:, 47
    enableCNG:, 46
    enableVAD:, 46
audioPlayLoopbackTest:
    Device Manage functions., 55
Call events, 56
    onInviteAnswered:callerDisplayName:caller:calleeDisplayName:callee:audioCodecs:videoCodecs:existsAudio:existsVideo:, 57
    onInviteBeginingForward:, 58
    onInviteClosed:, 58
    onInviteConnected:, 58
    onInviteFailure:reason:code:, 58
    onInviteIncoming:callerDisplayName:caller:calleeDisplayName:callee:audioCodecs:videoCodecs:existsAudio:existsVideo:, 56
    onInviteRinging:statusText:statusCode:, 57
    onInviteSessionProgress:audioCodecs:videoCodecs:existsEarlyMedia:existsAudio:existsVideo:, 57

onInviteTrying:, 57
onInviteUpdated:audioCodecs:videoCodecs:existsAudio:existsVideo:, 58
onRemoteHold:, 59
onRemoteUnHold:audioCodecs:videoCodecs:existsAudio:existsVideo:, 59
Call functions, 27
answerCall:videoCall:, 29
call:sendSdp:videoCall:, 28
forwardCall:forwardTo:, 30
hangUp:, 28
hold:, 29
muteSession:muteIncomingAudio:muteOutgoingAudio:muteIncomingVideo:muteOutgoingVideo:, 30
rejectCall:code:, 28
sendDtmf:dtmfMethod:code:dtmfDration:playDtmfTone:, 31
unHold:, 30
updateCall:enableAudio:enableVideo:, 29
call:sendSdp:videoCall:
Call functions, 28
clearAddExtensionHeaders
Access SIP message header functions, 22
clearModifyHeaders
Access SIP message header functions, 23
Conference functions, 40
createConference:videoResolution:displayLocalVideo:, 41
joinToConference:, 41
removeFromConference:, 41
setConferenceVideoWindow:, 41
createConference:videoResolution:displayLocalVideo:
Conference functions, 41
detectMwi
Additional setting functions, 19
Device Manage functions., 51
audioPlayLoopbackTest:, 55
getMicVolume, 54
getNumOfPlayoutDevices, 52
getNumOfRecordingDevices, 52
getNumOfVideoCaptureDevices, 52
getPlayoutDeviceName:, 53
getRecordingDeviceName:, 53
getSpeakerVolume, 53
getSystemInputMute, 54
getSystemOutputMute, 54
getVideoCaptureDeviceName:uniqueId:deviceName:, 52
setMicVolume:, 54
setSpeakerVolume:, 53
setSystemInputMute:, 54
setSystemOutputMute:, 53
disableCallForward
Additional setting functions, 18
disableSessionTimer
Additional setting functions, 19
displayLocalVideo:
Audio and video functions, 26
DTMF events, 62
onRecvDtmfTone:tone:, 62
enable3GppTags:
Additional setting functions, 16
enableAEC:
Audio effect functions, 46
enableAGC:
Audio effect functions, 47
enableANS:
Audio effect functions, 47
enableAudioStreamCallback:enable:callbackMode:
RTP packets, Audio stream and video stream callback functions, 36
enableCallbackSendingSignaling:
Additional setting functions, 16
enableCallForward:forwardTo:
Additional setting functions, 18
enableCheckMwi:
Additional setting functions, 19
enableCNG:
Audio effect functions, 46
enableReliableProvisional:
Additional setting functions, 16
enableSendPcmStreamToRemote:state:streamSamplesPerSec:
Send audio and video stream functions, 34
enableSendVideoStreamToRemote:state:
Send audio and video stream functions, 34
enableSessionTimer:refreshMode:
Additional setting functions, 18
enableVAD:
Audio effect functions, 46
enableVideoStreamCallback:callbackMode:
RTP packets, Audio stream and video stream callback functions, 36
forwardCall:forwardTo:
Call functions, 30
getAudioRtcpStatistics:bytesSent:packetsSent:bytesReceived:packetsReceived:sendFractionLost:sendCumulativeLost:recvFractionLost:recvCumulativeLost:
RTP statistics functions, 45
getAudioRtpStatistics:averageJitterMs:maxJitterMs:discardedPackets:
RTP statistics functions, 44
getDynamicVolumeLevel:microphoneVolume:
Audio and video functions, 27
getExtensionHeaderValue:headerName:
Access SIP message header functions, 21
getLocalIpAddress:
NIC and local IP functions, 11

getMicVolume
Device Manage functions., 54
getNetworkStatistics:currentBufferSize:preferredBufferSize:currentPacketLossRate:currentDiscardRate:currentExpandRate:currentPreemptiveRate:currentAccelerateRate:
RTP statistics functions, 44
getNICNums
NIC and local IP functions, 11
getNumOfPlayoutDevices
Device Manage functions., 52
getNumOfRecordingDevices
Device Manage functions., 52
getNumOfVideoCaptureDevices
Device Manage functions., 52
getPlayoutDeviceName:
Device Manage functions., 53
getRecordingDeviceName:
Device Manage functions., 53
getSpeakerVolume
Device Manage functions., 53
getSystemInputMute
Device Manage functions., 54
getSystemOutputMute
Device Manage functions., 54
getVersion:minorVersion:
Additional setting functions, 16
getVideoCaptureDeviceName:uniqueId:deviceName:
Device Manage functions., 52
getVideoRtpStatistics:bytesSent:packetsSent:bytesReceived:packetsReceived:
RTP statistics functions, 45
hangUp:
Call functions, 28
hold:
Call functions, 29
INFO/OPTIONS message events, 63
onRecvInfo:, 63
onRecvOptions:, 63
Initialize and register functions, 8
initialize:loglevel:logPath:maxLine:agent:audioDeviceLayer:videoDeviceLayer:, 9
registerServer:retryTimes:, 10
setLicenseKey:, 10
setUser:displayName:authName:password:localIP:localSIPPort:userDomain:SIPServer:SIPServerPort:STUNServer:STUNServerPort:outboundServer:outboundServerPort:, 9
unRegisterServer, 10
initialize:loglevel:logPath:maxLine:agent:audioDeviceLayer:videoDeviceLayer:
Initialize and register functions, 9
isAudioCodecEmpty
Audio and video codecs functions, 13
isVideoCodecEmpty
Audio and video codecs functions, 13
joinToConference:
Conference functions, 41
MESSAGE message events, 64
onRecvMessage:mimeType:subMimeType:messageData:messageDataLength:, 65
onRecvOutOfDialogMessage:from:toDisplayName:to:mimeType:subMimeType:messageData:messageDataLength:, 65
onSendMessageFailure:messageId:reason:code:, 65
onSendMessageSuccess:messageId:, 65
onSendOutOfDialogMessageFailure:fromDisplayName:from:toDisplayName:to:reason:code:, 66
onSendOutOfDialogMessageSuccess:fromDisplayName:from:toDisplayName:to:, 66
modifyHeaderValue:headerValue:
Access SIP message header functions, 22
muteMicrophone:
Audio and video functions, 26
muteSession:muteIncomingAudio:muteOutgoingAudio:muteIncomingVideo:muteOutgoingVideo:
Call functions, 30
muteSpeaker:
Audio and video functions, 26
MWI events, 61
onWaitingFaxMessage:urgentNewMessageCount:urgentOldMessageCount:newMessageCount:oldMessageCount:, 62
onWaitingVoiceMessage:urgentNewMessageCount:urgentOldMessageCount:newMessageCount:oldMessageCount:, 61
NIC and local IP functions, 11
getLocalIpAddress:, 11
getNICNums, 11
onACTVTransferFailure:reason:code:
Refer events, 60
onACTVTransferSuccess:
Refer events, 60
onAudioRawCallback:audioCallbackMode:data:dataLength:samplingFreqHz:
Audio and video stream callback events, 68
onInviteAnswered:callerDisplayName:caller:calleeDisplayName:callee:audioCodecs:videoCodecs:existsAudio:existsVideo:
Call events, 57
onInviteBeginingForward:
Call events, 58
onInviteClosed:
Call events, 58
onInviteConnected:
Call events, 58
onInviteFailure:reason:code:
Call events, 58

onInviteIncoming:callerDisplayName:caller:calleeDisplayName:callee:audioCodecs:videoCodecs:existsAudio:existsVideo:
    Call events, 56
onInviteRinging:statusText:statusCode:
    Call events, 57
onInviteSessionProgress:audioCodecs:videoCodecs:existsEarlyMedia:existsAudio:existsVideo:
    Call events, 57
onInviteTrying:
    Call events, 57
onInviteUpdated:audioCodecs:videoCodecs:existsAudio:existsVideo:
    Call events, 58
onPlayAudioFileFinished:fileName:
    Play audio and video file finished events, 66
onPlayVideoFileFinished:
    Play audio and video file finished events, 67
onPresenceOffline:from:
    Presence events, 64
onPresenceOnline:from:stateText:
    Presence events, 64
onPresenceRecvSubscribe:fromDisplayName:from:subject:
    Presence events, 64
onReceivedRefer:referId:to:from:referSipMessage:
    Refer events, 59
onReceivedRTPPacket:isAudio:RTPPacket:packetSize:
    RTP callback events, 67
onReceivedSignaling:message:
    Signaling events, 61
onRecvDtmfTone:tone:
    DTMF events, 62
onRecvInfo:
    INFO/OPTIONS message events, 63
onRecvMessage:mimeType:subMimeType:messageData:messageDataLength:
    MESSAGE message events, 65
onRecvOptions:
    INFO/OPTIONS message events, 63
onRecvOutOfDialogMessage:from:toDisplayName:to:mimeType:subMimeType:messageData:messageDataLength:
    MESSAGE message events, 65
onReferAccepted:
    Refer events, 60
onReferRejected:reason:code:
    Refer events, 60
onRegisterFailure:statusCode:
    Register events, 56
onRegisterSuccess:statusCode:
    Register events, 56
onRemoteHold:
    Call events, 59
onRemoteUnHold:audioCodecs:videoCodecs:existsAudio:existsVideo:
    Call events, 59
onSendingRTPPacket:isAudio:RTPPacket:packetSize:
    RTP callback events, 67
onSendingSignaling:message:
    Signaling events, 61
onSendMessageFailure:messageId:reason:code:
    MESSAGE message events, 65
onSendMessageSuccess:messageId:
    MESSAGE message events, 65
onSendOutOfDialogMessageFailure:fromDisplayName:from:toDisplayName:to:reason:code:
    MESSAGE message events, 66
onSendOutOfDialogMessageSuccess:fromDisplayName:from:toDisplayName:to:
    MESSAGE message events, 66
onTransferRinging:
    Refer events, 60
onTransferTrying:
    Refer events, 60
onVideoRawCallback:videoCallbackMode:width:height:data:dataLength:
    Audio and video stream callback events, 68
onWaitingFaxMessage:urgentNewMessageCount:urgentOldMessageCount:newMessageCount:oldMessageCount:
    MWI events, 62
onWaitingVoiceMessage:urgentNewMessageCount:urgentOldMessageCount:newMessageCount:oldMessageCount:
    MWI events, 61
Play audio and video file finished events, 66
    onPlayAudioFileFinished:fileName:, 66
    onPlayVideoFileFinished:, 67
Play audio and video file to remoe functions, 38
    playAudioFileToRemote:filename:fileSamplesPerSec:loop:, 39
    playAudioFileToRemoteAsBackground:filename:fileSamplesPerSec:, 40
    playVideoFileToRemote:aviFile:loop:playAudio:, 38
    stopPlayAudioFileToRemote:, 39
    stopPlayAudioFileToRemoteAsBackground:, 40
    stopPlayVideoFileToRemote:, 39
playAudioFileToRemote:filename:fileSamplesPerSec:loop:
    Play audio and video file to remoe functions, 39
playAudioFileToRemoteAsBackground:filename:fileSamplesPerSec:
    Play audio and video file to remoe functions, 40

playVideoFileToRemote:aviFile:loop:playAudio:
Play audio and video file to remoe functions, 38
PortSIPSDK, 72
PortSIPVideoRenderView, 79
updateVideoRenderFrame:, 79
Presence events, 63
onPresenceOffline:from:, 64
onPresenceOnline:from:stateText:, 64
onPresenceRecvSubscribe:fromDisplayName:from:subject:, 64
Presence functions, 49
presenceAcceptSubscribe:, 50
presenceOffline:, 50
presenceOnline:statusText:, 50
presenceRejectSubscribe:, 50
presenceSubscribeContact:subject:, 49
presenceAcceptSubscribe:
Presence functions, 50
presenceOffline:
Presence functions, 50
presenceOnline:statusText:
Presence functions, 50
presenceRejectSubscribe:
Presence functions, 50
presenceSubscribeContact:subject:
Presence functions, 49
Record functions, 37
startRecord:recordFilePath:recordFileName:appendTimeStamp:audioFileFormat:audioRecordMode:aviFileCodecType:videoRecordMode:, 37
stopRecord:, 38
Refer events, 59
onACTVTransferFailure:reason:code:, 60
onACTVTransferSuccess:, 60
onReceivedRefer:referId:to:from:referSipMessage:, 59
onReferAccepted:, 60
onReferRejected:reason:code:, 60
onTransferRinging:, 60
onTransferTrying:, 60
Refer functions, 31
acceptRefer:referSignaling:, 33
attendedRefer:replaceSessionId:referTo:, 32
refer:referTo:, 32
rejectRefer:, 33
refer:referTo:
Refer functions, 32
Register events, 55
onRegisterFailure:statusCode:, 56
onRegisterSuccess:statusCode:, 56
registerServer:retryTimes:
Initialize and register functions, 10
rejectCall:code:
Call functions, 28
rejectRefer:
Refer functions, 33
removeFromConference:
Conference functions, 41
RTP and RTCP QOS functions, 42
setAudioQos:DSCPValue:priority:, 43
setAudioRtcpBandwidth:BitsRR:BitsRS:KBitsAS:, 42
setVideoQos:DSCPValue:, 43
setVideoRtcpBandwidth:BitsRR:BitsRS:KBitsAS:, 42
RTP callback events, 67
onReceivedRTPPacket:isAudio:RTPPacket:packetSize:, 67
onSendingRTPPacket:isAudio:RTPPacket:packetSize:, 67
RTP packets, Audio stream and video stream callback functions, 35
enableAudioStreamCallback:enable:callbackMode:, 36
enableVideoStreamCallback:callbackMode:, 36
setRtpCallback:, 35
RTP statistics functions, 43
getAudioRtcpStatistics:bytesSent:packetsSent:bytesReceived:packetsReceived:sendFractionLost:sendCumulativeLost:recvFractionLost:recvCumulativeLost:, 45
getAudioRtpStatistics:averageJitterMs:maxJitterMs:discardedPackets:, 44
getNetworkStatistics:currentBufferSize:preferredBufferSize:currentPacketLossRate:currentDiscardRate:currentExpandRate:currentPreemptiveRate:currentAccelerateRate:, 44
getVideoRtpStatistics:bytesSent:packetsSent:bytesReceived:packetsReceived:, 45
SDK Callback events, 55
SDK functions, 8
Send audio and video stream functions, 33
enableSendPcmStreamToRemote:state:streamSamplesPerSec:, 34
enableSendVideoStreamToRemote:state:, 34
sendPcmStreamToRemote:data:, 34
sendVideoStreamToRemote:data:width:height:, 35
Send OPTIONS/INFO/MESSAGE functions, 47
sendInfo:mimeType:subMimeType:infoContents:, 48
sendMessage:mimeType:subMimeType:message:messageLength:, 48
sendOptions:sdp:, 47
sendOutOfDialogMessage:mimeType:subMimeType:message:messageLength:, 48

sendDtmf:dtmfMethod:code:dtmfDration:playDtmfTone:
Call functions, 31
sendInfo:mimeType:subMimeType:infoContents:
Send OPTIONS/INFO/MESSAGE functions, 48
sendMessage:mimeType:subMimeType:message:messageLength:
Send OPTIONS/INFO/MESSAGE functions, 48
sendOptions:sdp:
Send OPTIONS/INFO/MESSAGE functions, 47
sendOutOfDialogMessage:mimeType:subMimeType:message:messageLength:
Send OPTIONS/INFO/MESSAGE functions, 48
sendPcmStreamToRemote:data:
Send audio and video stream functions, 34
sendVideo:sendState:
Audio and video functions, 25
sendVideoStreamToRemote:data:width:height:
Send audio and video stream functions, 35
setAudioCodecParameter:parameter:
Audio and video codecs functions, 13
setAudioCodecPayloadType:payloadType:
Audio and video codecs functions, 13
setAudioDeviceId:outputDeviceId:
Audio and video functions, 27
setAudioQos:DSCPValue:priority:
RTP and RTCP QOS functions, 43
setAudioRtcpBandwidth:BitsRR:BitsRS:KBitsAS:
RTP and RTCP QOS functions, 42
setAudioSamples:maxPtime:
Additional setting functions, 20
setConferenceVideoWindow:
Conference functions, 41
setDisplayName:
Additional setting functions, 15
setDoNotDisturb:
Additional setting functions, 19
setKeepAliveTime:
Additional setting functions, 20
setLicenseKey:
Initialize and register functions, 10
setLocalVideoWindow:
Audio and video functions, 25
setMicVolume:
Device Manage functions., 54
setRemoteVideoWindow:remoteVideoWindow:
Audio and video functions, 25
setRtcpPortRange:maximumRtcpAudioPort:minimumRtcpVideoPort:maximumRtcpVideoPort:
Additional setting functions, 17
setRtpCallback:
RTP packets, Audio stream and video stream callback functions, 35
setRtpKeepAlive:keepAlivePayloadType:deltaTransmitTimeMS:
Additional setting functions, 19
setRtpPortRange:maximumRtpAudioPort:minimumRtpVideoPort:maximumRtpVideoPort:
Additional setting functions, 17
setSpeakerVolume:
Device Manage functions., 53
setSrtpPolicy:
Additional setting functions, 17
setSystemInputMute:
Device Manage functions., 54
setSystemOutputMute:
Device Manage functions., 53
setUser:displayName:authName:password:localIP:localSIPPort:userDomain:SIPServer:SIPServerPort:STUNServer:STUNServerPort:outboundServer:outboundServerPort:
Initialize and register functions, 9
setVideoBitrate:
Audio and video functions, 24
setVideoCodecParameter:parameter:
Audio and video codecs functions, 14
setVideoCodecPayloadType:payloadType:
Audio and video codecs functions, 13
setVideoDeviceId:
Audio and video functions, 24
setVideoFrameRate:
Audio and video functions, 25
setVideoNackStatus:
Audio and video functions, 26
setVideoOrientation:
Audio and video functions, 25
setVideoQos:DSCPValue:
RTP and RTCP QOS functions, 43
setVideoResolution:
Audio and video functions, 24
setVideoRtcpBandwidth:BitsRR:BitsRS:KBitsAS:
RTP and RTCP QOS functions, 42
Signaling events, 60
onReceivedSignaling:message:, 61
onSendingSignaling:message:, 61
startRecord:recordFilePath:recordFileName:appendTimeStamp:audioFileFormat:audioRecordMode:aviFileCodecType:videoRecordMode:
Record functions, 37
stopPlayAudioFileToRemote:
Play audio and video file to remoe functions, 39
stopPlayAudioFileToRemoteAsBackground:
Play audio and video file to remoe functions, 40

stopPlayVideoFileToRemote:
Play audio and video file to remoe functions, 39
stopRecord:
Record functions, 38
unHold:
Call functions, 30
unRegisterServer
Initialize and register functions, 10
updateCall:enableAudio:enableVideo:
Call functions, 29
updateVideoRenderFrame:
PortSIPVideoRenderView, 79