PortSIP VoIP SDK Manual for Mac

Version 11.3
Wed Dec 28 2016

Table of Contents

Welcome to PortSIP VoIP SDK

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

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

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

Getting Started

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

Contents

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

SDK User Manual

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

Website

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

Support

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

Installation Prerequisites

Development using the PortSIP VoIP/IMS SDK for iOS requires an Intel-based Macintosh running Snow Leopard (OS X 10.8 or higher), Xcode 5.0 or 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 found and downloaded at:
                http://www.portsip.com/downloads
```

http://www.portsip.com/portsip-voip-sdk.

2. How can I compile the sample project?

```
      1. Download the sample project from PortSIP website.
      2. Extract the .zip file.
      3. Open the project with your Xcode:
      4. Compile the sample project directly. The trial version SDK allows a 2-3 minutes
conversation.
```

3. How can I 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. Copy 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
     Specify destination frameworks. Click the Add button (+) to select PortSIPSDK.framework
to be copied and click Add.
  5. Add the code in .h file to import the SDK. For example:
     #import <PortSIPSDK/PortSIPSDK.h>
  6. Inherit the interface PortSIPEventDelegate to process the callback events.
  7. Initialize SDK. For example:
     mPortSIPSDK = [[PortSIPSDK alloc] init];
     mPortSIPSDK.delegate = self;
  8. For more details, please read the Sample project source code.
```

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

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

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

3. Enter a user name and password on A. For example, user name is 111, password is aaa (you
can enter anything for   the password as 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 is already in use, the
P2PSample will prompt "Initialize failure". In case of this, please click the "Uninitialize"
button and change the local port, and click the "Initialize" button to proceed again.

5. The log box will show "Initialized" if the SDK initialization succeeded.

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

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

5. Is the SDK thread safe?

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

6. Does the SDK support native 64 bits?

Yes, the SDK support 64 bits.

Module Index

Modules

Here is a list of all modules:

Hierarchical Index

Class Hierarchy

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

Class Index

Class List

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

Module Documentation

SDK functions

Modules

- Initialize and register functions
- NIC and local IP functions
- Audio and video codecs functions
- Additional settings functions
- Access SIP message header functions
- Audio and video functions
- Call functions
- Refer functions
- Send audio and video stream functions
- RTP packets, Audio stream and video stream callback functions
- Record functions
- Play audio and video file to remote 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.
- (int) - PortSIPSDK::setInstanceId:
  Set the instance Id, the outbound instanceId((RFC5626)) used in contact headers.
- (void) - PortSIPSDK::unInitialize
  Un-initialize the SDK and release resources.
- (int) - PortSIPSDK::setUser:displayName:authName:password: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. It must be 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 SIP blocking. It must be used with PERS.
logLevel	Set the application log level. The SDK will generate "PortSIP_Log_datatime.log" file if the log enabled.
logFilePath	The log file path. The path (folder) MUST be existent.
maxCallLines	Theoretically unlimited lines are supported depending on the device capability. For SIP client recommended value ranges 1 - 100.
sipAgent	The User-Agent header to be inserted 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, it will return value 0. If the function fails, it will return a specific error code.

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

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

Parameters:

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

Returns:

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

- (int) 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. It's 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 equal to the username).
password	The password of user. It's optional.
localIP	The local computer IP address to be bound (for example: 192.168.1.108). It will be used for sending and receiving SIP messages and RTP packets. If this IP is passed in IPv6 format, the SDK will be using IPv6. If you want the SDK to choose correct network interface (IP) automatically, please pass the "0.0.0.0"(for IPv4) or "::" (for IPv6).
localSIPPort	The SIP message transport listener port (for example: 5060).
userDomain	User domain; this parameter is optional, which allows to pass an empty string if you are not using 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, used for NAT traversal. It's optional and can be used to 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 allows to pass an empty string if outbound server is not used.
outboundServerPort	Outbound proxy server port. It will be ignored if the outboundServer is empty.

Returns:

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

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

Register to SIP proxy server (login to server)

Parameters:

expires	Registration refreshment interval in seconds. Maximum of 3600 allowed. It will be inserted into SIP REGISTER message headers.
retryTimes	The retrial times if failed to refresh the registration. It could be set to less than or equal to 0, and the retry will be disabled and onRegisterFailure callback triggered for retrial failure.

Returns:

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

- (int) unRegisterServer

Un-register from the SIP proxy server.

Returns:

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

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

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

Parameters:

key	The SDK license key. Please purchase from PortSIP.

Returns:

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

NIC and local IP functions

Functions

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

Detailed Description

Function Documentation

- (int) getNICNums

Get the Network Interface Card numbers.

Returns:

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

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

Get the local IP address by Network Interface Card index.

Parameters:

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

Returns:

The buffer that to receives the IP.

Audio and video codecs functions

Functions

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

Detailed Description

Function Documentation

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

Enable an audio codec. It will appear in SDP.

Parameters:

codecType	Audio codec type.

Returns:

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

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

Enable a video codec. It will appear in SDP.

Parameters:

codecType	Video codec type.

Returns:

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

- (BOOL) isAudioCodecEmpty

Detect if enabled audio codecs is empty or not.

Returns:

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

- (BOOL) isVideoCodecEmpty

Detect if enabled video codecs is empty or not.

Returns:

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

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

Set the RTP payload type for dynamic audio codec.

Parameters:

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

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

Set the RTP payload type for dynamic Video codec.

Parameters:

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

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

Set the codec parameter for audio codec.

Parameters:

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

Returns:

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

Remarks:

Example:

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

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

Set the codec parameter for video codec.

Parameters:

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

Returns:

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

Remarks:

Example:

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

Additional settings 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, including "ims.icsi.mmtel" and "g.3gpp.smsip".
- (void) - PortSIPSDK::enableCallbackSendingSignaling:
  Enable/disable callback the sent 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 forwarding.
- (int) - PortSIPSDK::disableCallForward
  Disable the call forwarding. SDK will not forward any incoming calls after this function is called.
- (int) - PortSIPSDK::enableSessionTimer:refreshMode:
  Allows to periodically refresh Session Initiation Protocol (SIP) sessions by sending INVITE requests repeatedly.
- (int) - PortSIPSDK::disableSessionTimer
  Disable the session timer.
- (void) - PortSIPSDK::setDoNotDisturb:
  Enable the "Do not disturb" to enable/disable.
- (int) - PortSIPSDK::detectMwi
  Used to obtain the MWI status.
- (int) - PortSIPSDK::enableCheckMwi:
  Allow to enable/disable the check MWI (Message Waiting Indication).
- (int) - PortSIPSDK::setRtpKeepAlive:keepAlivePayloadType:deltaTransmitTimeMS:
  Enable or disable to send RTP keep-alive packet when the call is established.
- (int) - PortSIPSDK::setKeepAliveTime:
  Enable or disable to send SIP keep-alive packet.

- (int) - PortSIPSDK::setAudioSamples:maxPtime:
  Set the audio capturing sample.
- (int) - PortSIPSDK::addSupportedMimeType:mimeType:subMimeType:
  Set the SDK to receive the SIP message that includes 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, it will return value 0. If the function fails, it will return 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, it will return value 0. If the function fails, it will return a specific error code.

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

Enable/disable PRACK.

Parameters:

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

Returns:

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

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

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

Parameters:

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

Returns:

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

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

Enable/disable callback the sent SIP messages.

Parameters:

enable	Set as true to enable callback the sent SIP messages, or false to disable. Once enabled, the "onSendingSignaling" event will be triggered when the SDK sends a SIP message.

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

Set the SRTP policy.

Parameters:

srtpPolicy	The SRTP policy.

Returns:

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

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

Set the RTP ports range for audio and video streaming.

Parameters:

minimumRtpAudio Port	The minimum RTP port for audio stream.
maximumRtpAudio Port	The maximum RTP port for audio stream.
minimumRtpVideo Port	The minimum RTP port for video stream.
maximumRtpVideo Port	The maximum RTP port for video stream.

Returns:

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

Remarks:

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

- (int) 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, it will return value 0. If the function fails, it will return a specific error code.

Remarks:

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

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

Enable call forwarding.

Parameters:

forBusyOnly	If this parameter is set as true, the SDK will forward all incoming calls when currently it's busy. If it's set as false, the SDK forward all incoming calls.
forwardTo	The target of call forwarding. It must be like sip:xxxx@sip.portsip.com.

Returns:

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

- (int) disableCallForward

Disable the call forwarding. SDK will not forward any incoming calls after this function is called.

Returns:

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

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

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

Parameters:

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

Returns:

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

Remarks:

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

- (int) disableSessionTimer

Disable the session timer.

Returns:

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

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

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

Parameters:

state	If it's set to true, the SDK will reject all incoming calls anyway.

- (int) detectMwi

Used to obtain the MWI status.

Returns:

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

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

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

Parameters:

state	If it's set as true, MWI will be checked automatically once successfully registered to a SIP proxy server.

Returns:

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

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

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

Parameters:

state	Set to true to allow sending the keep-alive packet during the conversation.
keepAlivePayloadType	The payload type of the keep-alive RTP packet. It's usually set to 126.
deltaTransmitTimeMS	The keep-alive RTP packet sending interval, in milliseconds. Recommended to set between 15000 and 300000.

Returns:

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

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

Enable or disable to send SIP keep-alive packet.

Parameters:

keepAliveTime	This is the SIP keep-alive time interval in seconds. Set it to 0 to disable the SIP keep alive. Recommended to set to 30 or 50.

Returns:

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

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

Set the audio capturing sample.

Parameters:

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

Returns:

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

Remarks:

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

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

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

Parameters:

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

Returns:

If the function succeeds, it will return value 0. If the function fails, it will return specific error code. By default, the PortSIP VoIP SDK support these media types (mime types) included in the below incoming SIP messages:

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

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

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

If user wishes to receive the NOTIFY message with "application/media_control+xml", please:

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

For details about the mime type, please visit: 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 modified headers value, and do not modify every outgoing SIP message header values any longer.

Detailed Description

Function Documentation

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

Access the SIP header of SIP message.

Parameters:

sipMessage	The SIP message.
headerName	The header that needs access to the SIP message.

Returns:

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

Remarks:

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

```
NSString* headerValue = [myVoIPSdk 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 appear in every outgoing SIP message.
headerValue	The custom header value.

Returns:

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

- (int) clearAddExtensionHeaders

Clear the added extension headers (custom headers)

Returns:

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

Remarks:

```
For example, we have added two custom headers into every outgoing SIP message and wish
to 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 of which the value will be modified.
headerValue	The heaver value to be modified.

Returns:

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

- (int) clearModifyHeaders

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

Returns:

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

Remarks:

For example, if we have modified two headers value for every outgoing SIP message and wish to 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 devices that will be used for video call.
- (int) - PortSIPSDK::setVideoResolution:height:
  Set the video capturing resolution.
- (int) - PortSIPSDK::setVideoBitrate:
  Set the video bitrate.
- (int) - PortSIPSDK::setVideoFrameRate:

Set the video frame rate.

- (int) - PortSIPSDK::sendVideo:sendState:
  Send the video to remote side.
- (int) - PortSIPSDK::setVideoOrientation:
  Change the orientation of the video.
- (void) - PortSIPSDK::setLocalVideoWindow:
  Set the window that is used to display the local video image.
- (int) - PortSIPSDK::setRemoteVideoWindow:remoteVideoWindow:
  Set the window for a session that is used 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 (RFC4585) that helps to improve the video quality.
- (void) - PortSIPSDK::muteMicrophone:
  Mute the device microphone. It's unavailable for Android and iOS.
- (void) - PortSIPSDK::muteSpeaker:
  Mute the device speaker. It's unavailable for Android and iOS.
- (int) - PortSIPSDK::setAudioDeviceId:outputDeviceId:
  Set the audio device that will be used for audio call.
- (void) - PortSIPSDK::getDynamicVolumeLevel:microphoneVolume:
  Obtain the dynamic microphone volume level from current call.
- (int) - PortSIPSDK::setChannelOutputVolumeScaling:scaling:

Detailed Description

Function Documentation

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

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

Parameters:

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

Returns:

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

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

Set the video capturing resolution.

Parameters:

width	Video width.
height	Video height.

Returns:

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

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

Set the video bitrate.

Parameters:

bitrateKbps	The video bitrate in KBPS.

Returns:

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

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

Set the video frame rate.

Parameters:

frameRate	The frame rate value, with its minimum value 5, and maximum value 30. The bigger value renders you better video quality but required more bandwidth.

Returns:

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

Remarks:

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

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

Send the video to remote side.

Parameters:

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

Returns:

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

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

Change the orientation of the video.

Parameters:

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

Returns:

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

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

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

Parameters:

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

Returns:

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

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

Start/stop to display the local video image.

Parameters:

state	Set to true to display local video iamge.

Returns:

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

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

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

Parameters:

state	Set to true to enable.

Returns:

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

- (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. It also could be set to false to un-mute.

- (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 microphone is muted. It also could be set to false to un-mute.

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

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

Parameters:

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

Returns:

If the function succeeds, it will return value 0. If the function fails, it will return 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 value ranges 0 - 9.
microphoneVolume	Return the dynamic microphone volume by this parameter. The value ranges 0 - 9.

Remarks:

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

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

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

Parameters:

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

Returns:

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

Call functions

Functions

- (long) - PortSIPSDK::call:sendSdp:videoCall:
  Make a call.
- (int) - PortSIPSDK::rejectCall:code:
  rejectCall Reject the incoming call.
- (int) - PortSIPSDK::hangUp:
  hangUp Hang up the call.
- (int) - PortSIPSDK::answerCall:videoCall:
  answerCall Answer the incoming call.
- (int) - PortSIPSDK::updateCall:enableAudio:enableVideo:
  Use the re-INVITE to update the established call.
- (int) - PortSIPSDK::hold:
  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 receiving 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, sip:user001@sip.iptel.org or sip:user002@sip.yourdomain.com:5068
sendSdp	If it's set to false, the outgoing call will not include the SDP in INVITE message.
videoCall	If it's set to true, and at least one video codec added, the outgoing call will include the video codec into SDP.

Returns:

If the function succeeds, it will return the session ID of the call greater than 0. If the function fails, it will return a specific error code. Note: the function success just means the outgoing call is being processed. 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, it will return value 0. If the function fails, it will return a specific error code.

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

hangUp Hang up the call.

Parameters:

sessionId	Session ID of the call.

Returns:

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

- (int) 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 it to true to answer the video call. If it's set to false, the answer call will not include video codec answer anyway.

Returns:

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

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

Use the re-INVITE to update the established call.

Parameters:

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

Returns:

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

Remarks:

Example usage:

Example 1: A called B with the audio only, and B answered A, then an audio conversation between A and B established. Now A wants to see B visually, A could use these functions.

```
[myVoIPSdk clearVideoCodec];
[myVoIPSdk addVideoCodec:VIDEOCODEC H264];
[myVoIPSdk updateCall:sessionId enableAudio:true enableVideo:true];
```

Example 2: Remove video stream from current conversation.

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

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

To place a call on hold.

Parameters:

sessionId	The session ID of call.

Returns:

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

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

Take off hold.

Parameters:

sessionId	The session ID of call.

Returns:

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

- (int) 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 stream, and remote side audio can't be heard.
muteOutgoingAudio	Set it to true to mute outgoing audio stream, and the remote side can't hear the audio.
muteIncomingVideo	Set it to true to mute incoming video stream, and remote side video will be invisible.
muteOutgoingVideo	Set it to true to mute outgoing video stream, and remote side can't see the video.

Returns:

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

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

Forward call to another one when receiving the incoming call.

Parameters:

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

Returns:

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

- (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 sending DTMF tone with two methods: DTMF_RFC2833 and DTMF_INFO. The DTMF_RFC2833 is recommended.
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. Recommended value 160.
playDtmfTone	Set to true to enable the SDK play local DTMF tone sound when sending DTMF.

Returns:

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

Refer functions

Functions

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

- (int) - PortSIPSDK::attendedRefer:replaceSessionId:referTo:
  Make an attended refer.
- (long) - PortSIPSDK::acceptRefer:referSignaling:
  Accept the REFER request. A new call will be made if called this function. It's usually called after onReceivedRefer callback event.
- (int) - PortSIPSDK::rejectRefer:
  Reject the REFER request.

Detailed Description

Function Documentation

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

Refer the current call to another one.

Parameters:

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

Returns:

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

Remarks:

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

You can watch the video on YouTube at https://www.youtube.com/watch?v=_2w9EGgr3FY and it will demonstrate 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 refer call.
referTo	Target of the refer. It can be either "sip:number@sipserver.com" or "number".

Returns:

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

Remarks:

Please read the sample project source code for more details, or you can watch the video on YouTube at https://www.youtube.com/watch?v=_2w9EGgr3FY, which will demonstrate the transfer.

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

Accept the REFER request. A new call will be made if called this function. It's usually called after onReceivedRefer callback event.

Parameters:

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

Returns:

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

Send audio and video stream functions

Functions

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

Detailed Description

Function Documentation

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

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

Parameters:

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

Returns:

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

Remarks:

This function MUST be called first if we 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 instead of audio device capturing (microphone).

Parameters:

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

Returns:

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

Parameters:

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

Returns:

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

- (int) 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. It must be in i420 format.
width	The video image width.
height	The video image height.

Returns:

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

Remarks:

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

Before calling this function, enableSendVideoStreamToRemote function MUST be called first. 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 to the sent and received RTP packets.
- (int) - PortSIPSDK::enableAudioStreamCallback:enable:callbackMode:
  Enable/disable the audio stream callback.
- (int) - PortSIPSDK::enableVideoStreamCallback:callbackMode:
  Enable/disable the video stream callback.
- (int) - PortSIPSDK::enableVideoDecoderCallback:
  Enable/disable the video Decoder callback.

Detailed Description

Function Documentation

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

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

Parameters:

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

Returns:

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

- (int) enableAudioStreamCallback: (long) *sessionId* enable: (BOOL) *enable* callbackMode: (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, or 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 based on the given sessionId.
AUDIOSTREAM_REMOTE_MIX	Callback the received audio stream that is mixed to include all channels.
AUDIOSTREAM_REMOTE_PER_CHANNEL	Callback the received audio stream for one channel based on the given sessionId.

Returns:

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

Remarks:

onAudioRawCallback event will be triggered if the callback is enabled.

- (int) enableVideoStreamCallback: (long) *sessionId* callbackMode: (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, it will return value 0. If the function fails, it will return a specific error code.

Remarks:

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

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

Enable/disable the video Decoder callback.

Parameters:

enable	Set to true to enable video Decoder callback, or false to stop the callback.

Returns:

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

Remarks:

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

Record functions

Functions

- (int) - PortSIPSDK::startRecord:recordFilePath:recordFileName:appendTimeStamp:audioFileFormat:audioRecordMode:aviFileCodecType:videoRecordMode:
  Start recording 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 recording the call.

Parameters:

sessionId	The session ID of call conversation.
recordFilePath	The filepath to which the recording will be saved. It must be existent.
recordFileName	The file name of recording. For example: audiorecord.wav or videorecord.avi.
appendTimeStamp	Set to true to append the timestamp to the file name of the recording.
audioFileFormat	The file format for the audio recording.
audioRecordMode	The audio recording mode.
aviFileCodecType	The codec which is used for compressing the video data to save into video recording file.
videoRecordMode	Allow to set video recording mode. Support to record received and/or sent

	video.

Returns:

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

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

Stop record.

Parameters:

sessionId	The session ID of call conversation.

Returns:

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

Play audio and video file to remote functions

Functions

- (int) - PortSIPSDK::playVideoFileToRemote:aviFile:loop:playAudio:
  Play an AVI file to remote party.
- (int) - PortSIPSDK::stopPlayVideoFileToRemote:
  Stop playing video file to remote side.
- (int) - PortSIPSDK::playAudioFileToRemote:filename:fileSamplesPerSec:loop:
  Play a wave file to remote party.
- (int) - PortSIPSDK::stopPlayAudioFileToRemote:
  Stop playing 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 playing a 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 full filepath, such as "/test.avi".
loop	Set to false to stop playing video file when it ends, or true to play it repeatedly.
playAudio	If it's set to true, audio and video will be played together; or false to play video only.

Returns:

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

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

Stop playing video file to remote side.

Parameters:

sessionId	Session ID of the call.

Returns:

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

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

Play a wave file to remote party.

Parameters:

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

Returns:

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

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

Stop playing wave file to remote side.

Parameters:

sessionId	Session ID of the call.

Returns:

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

- (int) 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 full filepath, such as "/test.wav".
fileSamplesPerSec	The wave file sample in seconds. It should be 8000, 16000 or 32000.

Returns:

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

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

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

Parameters:

sessionId	Session ID of the call.

Returns:

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

Conference functions

Functions

- (int) - PortSIPSDK::createConference:videoWidth:videoHeight:displayLocalVideo:
  Create a conference. It will be failed if the existent conference is not ended yet.
- (void) - PortSIPSDK::destroyConference
  Destroy the existent conference.
- (int) - PortSIPSDK::setConferenceVideoWindow:
  Set the window for a conference that is used to display the received remote video image.
- (int) - PortSIPSDK::joinToConference:
  Join a session into existent conference. If the call is in hold, please un-hold first.
- (int) - PortSIPSDK::removeFromConference:
  Remove a session from an existent conference.

Detailed Description

Function Documentation

- (int) createConference: (PortSIPVideoRenderView *) *conferenceVideoWindow* videoWidth: (int) *videoWidth* videoHeight: (int) *videoHeight* displayLocalVideo: (BOOL) *displayLocalVideoInConference*

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

Parameters:

conferenceVideoWindow	The PortSIPVideoRenderView used for displaying the conference video.
videoWidth	The conference video width.
videoHeight	The conference video height.
displayLocalVideoInConference	Display the local video on video window.

Returns:

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

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

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

Parameters:

conferenceVideoWindow	The PortSIPVideoRenderView used to display the conference video.

Returns:

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

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

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

Parameters:

sessionId	Session ID of the call.

Returns:

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

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

Remove a session from an existent conference.

Parameters:

sessionId	Session ID of the call.

Returns:

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

RTP and RTCP QOS functions

Functions

- (int) - 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.
- (int) - PortSIPSDK::setVideoMTU:
  Set the MTU size for video RTP packet.

Detailed Description

Function Documentation

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

Set the audio RTCP bandwidth parameters as the RFC3556.

Parameters:

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

Returns:

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

- (int) 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, it will return value 0. If the function fails, it will return 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 value ranges 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 indiciates the priority, while value -1 leaves the priority settings unchanged.

Returns:

If the function succeeds, it will return value 0. If the function fails, it will return 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, or false to disable.
DSCPValue	The six-bit DSCP value. Valid value ranges 0-63. As defined in RFC 2472, the DSCP value is the high-order 6 bits of the IP version 4 (IPv4) TOS field and the IP version 6 (IPv6) Traffic Class field.

Returns:

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

- (int) setVideoMTU: (int) ***mtu***

Set the MTU size for video RTP packet.

Parameters:

mtu	Set MTU value. Allow value could range 512-65507. Other value will be modified to the default: 1450.

Returns:

If the function succeeds, it will return value 0. If the function fails, it will return 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 statistics of audio channel.
- (int) - PortSIPSDK::getAudioRtcpStatistics:bytesSent:packetsSent:bytesReceived:packetsReceived:sendFractionLost:sendCumulativeLost:recvFractionLost:recvCumulativeLost:
  Obtain the RTCP statistics of audio channel.
- (int) - PortSIPSDK::getVideoRtpStatistics:bytesSent:packetsSent:bytesReceived:packetsReceived:
  Obtain the RTP statistics 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 percentage.
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, it will return value 0. If the function fails, it will return a specific error code.

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

Obtain the RTP statistics 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, it will return value 0. If the function fails, it will return 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 statistics 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 percentage.
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, it will return value 0. If the function fails, it will return a specific error code.

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

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

Audio effect functions

Functions

- (void) - PortSIPSDK::enableVAD:
  Enable/disable Voice Activity Detection (VAD).
- (void) - PortSIPSDK::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, or false to disable.

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

Enable/disable AEC (Acoustic Echo Cancellation).

Parameters:

state	AEC type. It's defaulted as EC_NONE.

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

Enable/disable Comfort Noise Generator (CNG).

Parameters:

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

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

Enable/disable Automatic Gain Control (AGC).

Parameters:

state	AGC type. It's defaulted as AGC_NONE.

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

Enable/disable Audio Noise Suppression (ANS).

Parameters:

state	NS type. It's defaulted as 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 an 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 recipient of OPTIONS message.
sdp	The SDP of OPTIONS message. It's optional if user doesn't want to send the SDP with OPTIONS message.

Returns:

If the function succeeds, it will return value 0. If the function fails, it will return 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 sent with INFO message.

Returns:

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

- (long) sendMessage: (long) *sessionId* mimeType: (NSString *) *mimeType* subMimeType: (NSString *) *subMimeType* message: (NSData *) *message* messageLength: (int) *messageLength*

Send a MESSAGE message to remote side in dialog.

Parameters:

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. Binary data allowed.
messageLength	The message size.

Returns:

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

Remarks:

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

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

Example 2: Send a binary message.

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

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

Send an out of dialog MESSAGE message to remote side.

Parameters:

to	The message recipient, such as sip:receiver@portsip.com.
mimeType	The mime type of MESSAGE message.
subMimeType	The sub mime type of MESSAGE message.
message	The contents sent with MESSAGE message. Binary data allowed.
messageLength	The message size.

Returns:

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

Remarks:

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

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

Example 2: Send a binary message.

```
[myVoIPsdk sendOutOfDialogMessage:@"sip:user1@sip.portsip.com"
mimeType:@"application" subMimeType:@"vnd.3gpp.sms" 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 received from contact.
- (int) - PortSIPSDK::presenceRejectSubscribe:

Reject a presence SUBSCRIBE request that is 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 be like sip:contact001@sip.portsip.com.
subject	This subject text will be inserted into the SUBSCRIBE message. For example: "Hello, I'm Jason". The subject maybe in UTF-8 format, and you should use UTF-8 to decode it.

Returns:

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

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

Accept the presence SUBSCRIBE request received from contact.

Parameters:

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

Returns:

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

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

Reject a presence SUBSCRIBE request that is received from contact.

Parameters:

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

	subscription ID.

Returns:

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

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

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

Parameters:

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

Returns:

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

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

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

Parameters:

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

Returns:

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

Device Manage functions.

Functions

- (int) - PortSIPSDK::getNumOfVideoCaptureDevices
  Gets the number of available capturing devices.
- (int) - PortSIPSDK::getVideoCaptureDeviceName:uniqueId:deviceName:
  Gets the name of a specific video capturing 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:
  Get the name of a specific recording device given by an index.
- (NSString *) - PortSIPSDK::getPlayoutDeviceName:

Get 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.
- (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:
  Used for the loop back test for audio device.

Detailed Description

Function Documentation

- (int) getNumOfVideoCaptureDevices

Gets the number of available capturing devices.

Returns:

The return value is the count of video capturing devices. If fails, it will return a specific error code less than 0.

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

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

Parameters:

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

	null-terminated string in UTF-8 format.

Returns:

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

- (int) getNumOfRecordingDevices

Gets the number of audio devices available for audio recording.

Returns:

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

- (int) getNumOfPlayoutDevices

Gets the number of audio devices available for audio playout.

Returns:

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

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

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

Parameters:

index	Device index (0, 1, 2, ..., N-1), of which 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 UTF-8 format.

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

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

Parameters:

index	Device index (0, 1, 2, ..., N-1), of which 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 of speaker. Valid value ranges 0 - 255.

Returns:

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

- (int) getSpeakerVolume

Gets the speaker volume.

Returns:

If the function succeeds, it will return the value of speaker volume that ranges 0 - 255. If the function fails, it will return a specific error code.

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

Mutes the speaker device completely in the OS.

Parameters:

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

Returns:

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

- (BOOL) getSystemOutputMute

Retrieves the output device mute state in the operating system.

Returns:

If it returns 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. The valid value ranges 0 - 255.

Returns:

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

- (int) getMicVolume

Retrieves the current microphone volume.

Returns:

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

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

Mute the microphone input device completely in the OS.

Parameters:

enable	If it's set to true, the input device is muted. If false it is unmuted.

Returns:

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

- (BOOL) getSystemInputMute

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

Returns:

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

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

Used for the loop back test for audio device.

Parameters:

enable	Set to true to start audio look back test, or false 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 finishes 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 PortSIPEventDelegate) onRegisterSuccess: (char *) *statusText* statusCode: (int) *statusCode*

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

Parameters:

statusText	The status text.
statusCode	The status code.

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

If registration to SIP server fails, 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 PortSIPEventDelegate) 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 will be 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 list of matched audio codecs. It's separated by "#" if there are more than one codecs.
videoCodecs	The list of matched video codecs. It's separated by "#" if there are more than one codecs.
existsAudio	By setting to true, it indicates that this call includes the audio.
existsVideo	By setting to true, it indicates that this call includes the video.

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

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

Parameters:

sessionId	The session ID of the call.

- (void PortSIPEventDelegate) 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 list of matched audio codecs. It's separated by "#" if there are more than one codecs.
videoCodecs	The list of matched video codecs. It's separated by "#" if there are more than one codecs.
existsEarlyMedia	By setting to true, it indicates that this call has early media.
existsAudio	By setting to true, it indicates that this call includes the audio.
existsVideo	By setting to true, it indicates that this call includes the video.

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

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

Parameters:

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

- (void PortSIPEventDelegate) 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 answers the call, this event will be 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 list of matched audio codecs. It's separated by "#" if there are more than one codecs.
videoCodecs	The list of matched video codecs. It's separated by "#" if there are more than one codecs.
existsAudio	By setting to true, it indicates that this call includes the audio.
existsVideo	By setting to true, it indicates that this call includes the video.

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

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

Parameters:

sessionId	The session ID of the call.

reason	The failure reason.
code	The failure code.

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

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

Parameters:

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

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

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

Parameters:

forwardTo	The target SIP URI of the call forwarding.

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

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

Parameters:

sessionId	The session ID of the call.

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

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

Parameters:

sessionId	The session ID of the call.

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

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

Parameters:

sessionId	The session ID of the call.
audioCodecs	The list of matched audio codecs. It's separated by "#" if there are more than

	one codecs.
videoCodecs	The list of matched video codecs. It's separated by "#" if there are more than one codecs.
existsAudio	By setting to true, it indicates that this call includes the audio.
existsVideo	By setting to true, it indicates that this call includes the video.

Refer events

Functions

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

Detailed Description

Function Documentation

- (void PortSIPEventDelegate) 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 target of refer.
from	The sender of REFER message.
referSipMessage	The SIP message of "REFER". Pass it to "acceptRefer" function.

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

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

Parameters:

sessionId	The session ID of the call.

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

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

Parameters:

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

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

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

Parameters:

sessionId	The session ID of the call.

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

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

Parameters:

sessionId	The session ID of the call.

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

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

Parameters:

sessionId	The session ID of the call.

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

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

Parameters:

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

Signaling events

Functions

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

Detailed Description

Function Documentation

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

This event will be triggered when receiving a SIP message.

Parameters:

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

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

This event will be triggered when a SIP message sent.

Parameters:

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

MWI events

Functions

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

Detailed Description

Function Documentation

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

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

Parameters:

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

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

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

Parameters:

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

DTMF events

Functions

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

Detailed Description

Function Documentation

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

This event will be triggered when receiving 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 PortSIPEventDelegate) onRecvOptions: (char *) ***optionsMessage***

This event will be triggered when receiving the OPTIONS message.

Parameters:

optionsMessage	The whole OPTIONS message received in text format.

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

This event will be triggered when receiving the INFO message.

Parameters:

infoMessage	The whole INFO message received 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 PortSIPEventDelegate) onPresenceRecvSubscribe: (long) ***subscribeId*** **fromDisplayName: (char *)** ***fromDisplayName*** **from: (char *)** ***from*** **subject: (char *)** ***subject***

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

Parameters:

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

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

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

Parameters:

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

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

When the contact changes to offline=, this event will be triggered.

Parameters:

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

MESSAGE message events

Functions

- (void) - <PortSIPEventDelegate >::onRecvMessage:mimeType:subMimeType:messageData:messageDataLength:
- (void) - <PortSIPEventDelegate >::onRecvOutOfDialogMessage:from:toDisplayName:to:mimeType:subMimeType:messageData:messageDataLength:
- (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 PortSIPEventDelegate) onRecvMessage: (long) *sessionId* mimeType: (char *) *mimeType* subMimeType: (char *) *subMimeType* messageData: (unsigned char *) *messageData* messageDataLength: (int) *messageDataLength*

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

Parameters:

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

- (void PortSIPEventDelegate) 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 can be text or binary data.
messageDataLength	The length of "messageData".

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

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

Parameters:

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

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

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

Parameters:

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

reason	The failure reason.
code	Failure code.

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

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

Parameters:

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

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

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

Parameters:

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

Play audio and video file finishes events

Functions

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

Detailed Description

Function Documentation

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

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

Parameters:

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

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

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

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

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

Parameters:

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

Note:

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

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

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

Parameters:

sessionId	The session ID of the call.
isAudio	If the received RTP packet is of audio, this parameter returns true; otherwise

	false.
RTPPacket	The memory of whole RTP packet.
packetSize	The size of received RTP Packet.

Note:

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

Audio and video stream callback events

Functions

- (void) - <PortSIPEventDelegate >::onAudioRawCallback:audioCallbackMode:data:dataLength:samplingFreqHz:
- (void) - <PortSIPEventDelegate >::onVideoRawCallback:videoCallbackMode:width:height:data:dataLength:
- (void) - <PortSIPEventDelegate >::onVideoDecoderCallback:width:height:framerate:bitrate:

Detailed Description

Function Documentation

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

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

Parameters:

sessionId	The session ID of the call.
audioCallbackMode	The type which is passed in enableAudioStreamCallback function.
data	The memory of audio stream. It's in 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 is time-consuming, you should post a message to another thread and execute SDK API functions or other code in another thread.

- (void PortSIPEventDelegate) 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 receiving the video packets with enableVideoStreamCallback function called.

Parameters:

sessionId	The session ID of the call.
videoCallbackMode	The type which is passed in enableVideoStreamCallback function.
width	The width of video image.
height	The height of video image.
data	The memory of video stream. It's in YUV420 format, YV12.
dataLength	The data size.

Note:

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

- (void PortSIPEventDelegate) onVideoDecoderCallback: (long) *sessionId* width: (int) *width* height: (int) *height* framerate: (int) *framerate* bitrate: (int) *bitrate*

This event will be triggered once per second containing the frame rate and bit rate, for the incoming stream or new incoming Video size is detected. if called enableVideoDecoderCallback function.

Parameters:

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

Note:

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

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:
- (void) - onVideoDecoderCallback:width:height:framerate:bitrate:

Detailed Description

PortSIP SDK Callback events Delegate.

Author:

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.
- (int) - setInstanceId:
  Set the instance Id, the outbound instanceId((RFC5626)) used in contact headers.
- (void) - unInitialize
  Un-initialize the SDK and release resources.
- (int) - setUser:displayName:authName:password: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. It must be called before setUser function.
- (int) - getNICNums
  Get the Network Interface Card numbers.
- (NSString *) - getLocalIpAddress:
  Get the local IP address by Network Interface Card index.
- (int) - addAudioCodec:
  Enable an audio codec. It will appear in SDP.
- (int) - addVideoCodec:
  Enable a video codec. It will appear in SDP.
- (BOOL) - isAudioCodecEmpty
  Detect if enabled audio codecs is empty or not.
- (BOOL) - isVideoCodecEmpty
  Detect if enabled video codecs is empty or not.
- (int) - setAudioCodecPayloadType:payloadType:
  Set the RTP payload type for dynamic audio codec.
- (int) - setVideoCodecPayloadType:payloadType:
  Set the RTP payload type for dynamic Video codec.
- (void) - clearAudioCodec
  Remove all enabled audio codecs.
- (void) - clearVideoCodec
  Remove all enabled video codecs.
- (int) - setAudioCodecParameter:parameter:
  Set the codec parameter for audio codec.
- (int) - setVideoCodecParameter:parameter:

Set the codec parameter for video codec.

- (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, including "ims.icsi.mmtel" and "g.3gpp.smsip".
- (void) - enableCallbackSendingSignaling:
  Enable/disable callback the sent 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 forwarding.
- (int) - disableCallForward
  Disable the call forwarding. SDK will not forward any incoming calls after this function is called.
- (int) - enableSessionTimer:refreshMode:
  Allows to periodically refresh Session Initiation Protocol (SIP) sessions by sending INVITE requests repeatedly.
- (int) - disableSessionTimer
  Disable the session timer.
- (void) - setDoNotDisturb:
  Enable the "Do not disturb" to enable/disable.
- (int) - detectMwi
  Used to obtain the MWI status.
- (int) - enableCheckMwi:
  Allow to enable/disable the check MWI (Message Waiting Indication).
- (int) - setRtpKeepAlive:keepAlivePayloadType:deltaTransmitTimeMS:
  Enable or disable to send RTP keep-alive packet when the call is established.
- (int) - setKeepAliveTime:
  Enable or disable to send SIP keep-alive packet.
- (int) - setAudioSamples:maxPtime:
  Set the audio capturing sample.
- (int) - addSupportedMimeType:mimeType:subMimeType:
  Set the SDK to receive the SIP message that includes 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 modified headers value, and do not modify every outgoing SIP message header values any longer.
- (int) - setVideoDeviceId:
  Set the video devices that will be used for video call.
- (int) - setVideoResolution:height:
  Set the video capturing resolution.
- (int) - setVideoBitrate:
  Set the video bitrate.
- (int) - setVideoFrameRate:
  Set the video frame rate.
- (int) - sendVideo:sendState:
  Send the video to remote side.
- (int) - setVideoOrientation:
  Change the orientation of the video.
- (void) - setLocalVideoWindow:
  Set the window that is used to display the local video image.
- (int) - setRemoteVideoWindow:remoteVideoWindow:
  Set the window for a session that is used to display the received remote video image.
- (int) - displayLocalVideo:
  Start/stop to display the local video image.
- (int) - setVideoNackStatus:
  Enable/disable the NACK feature (RFC4585) that helps to improve the video quality.
- (void) - muteMicrophone:
  Mute the device microphone. It's unavailable for Android and iOS.
- (void) - muteSpeaker:
  Mute the device speaker. It's unavailable for Android and iOS.
- (int) - setAudioDeviceId:outputDeviceId:
  Set the audio device that will be used for audio call.
- (void) - getDynamicVolumeLevel:microphoneVolume:
  Obtain the dynamic microphone volume level from current call.
- (int) - setChannelOutputVolumeScaling:scaling:
- (long) - call:sendSdp:videoCall:
  Make a call.
- (int) - rejectCall:code:
  rejectCall Reject the incoming call.
- (int) - hangUp:
  hangUp Hang up the call.
- (int) - answerCall:videoCall:
  answerCall Answer the incoming call.
- (int) - updateCall:enableAudio:enableVideo:
  Use the re-INVITE to update the established call.
- (int) - hold:
  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 receiving the incoming call.
- (int) - sendDtmf:dtmfMethod:code:dtmfDration:playDtmfTone:
  Send DTMF tone.
- (int) - refer:referTo:
  Refer the current call to another one.
- (int) - attendedRefer:replaceSessionId:referTo:
  Make an attended refer.
- (long) - acceptRefer:referSignaling:
  Accept the REFER request. A new call will be made if called this function. It's usually called after onReceivedRefer callback event.
- (int) - rejectRefer:
  Reject the REFER request.
- (int) - enableSendPcmStreamToRemote:state:streamSamplesPerSec:
  Enable the SDK to send PCM stream data to remote side from another source instead of microphone.
- (int) - sendPcmStreamToRemote:data:
  Send the audio stream in PCM format from another source instead of audio device capturing (microphone).
- (int) - enableSendVideoStreamToRemote:state:
  Enable the SDK to send video stream data to remote side from another source instead of camera.
- (int) - sendVideoStreamToRemote:data:width:height:
  Send the video stream to remote.
- (int) - setRtpCallback:
  Set the RTP callbacks to allow access to the sent and received RTP packets.
- (int) - enableAudioStreamCallback:enable:callbackMode:
  Enable/disable the audio stream callback.
- (int) - enableVideoStreamCallback:callbackMode:
  Enable/disable the video stream callback.
- (int) - enableVideoDecoderCallback:
  Enable/disable the video Decoder callback.
- (int) - startRecord:recordFilePath:recordFileName:appendTimeStamp:audioFileFormat:audioRecordMode:aviFileCodecType:videoRecordMode:
  Start recording the call.
- (int) - stopRecord:
  Stop record.
- (int) - playVideoFileToRemote:aviFile:loop:playAudio:
  Play an AVI file to remote party.
- (int) - stopPlayVideoFileToRemote:
  Stop playing video file to remote side.
- (int) - playAudioFileToRemote:filename:fileSamplesPerSec:loop:
  Play a wave file to remote party.
- (int) - stopPlayAudioFileToRemote:

Stop playing wave file to remote side.

- (int) - playAudioFileToRemoteAsBackground:filename:fileSamplesPerSec:
  Play an wave file to remote party as conversation background sound.
- (int) - stopPlayAudioFileToRemoteAsBackground:
  Stop playing a wave file to remote party as conversation background sound.
- (int) - createConference:videoWidth:videoHeight:displayLocalVideo:
  Create a conference. It will be failed if the existent conference is not ended yet.
- (void) - destroyConference
  Destroy the existent conference.
- (int) - setConferenceVideoWindow:
  Set the window for a conference that is used to display the received remote video image.
- (int) - joinToConference:
  Join a session into existent conference. If the call is in hold, please un-hold first.
- (int) - removeFromConference:
  Remove a session from an existent conference.
- (int) - setAudioRtcpBandwidth:BitsRR:BitsRS:KBitsAS:
  Set the audio RTCP bandwidth parameters as the RFC3556.
- (int) - setVideoRtcpBandwidth:BitsRR:BitsRS:KBitsAS:
  Set the video RTCP bandwidth parameters as the RFC3556.
- (int) - 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) - setVideoMTU:
  Set the MTU size for video RTP packet.
- (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 statistics of audio channel.
- (int) - getAudioRtcpStatistics:bytesSent:packetsSent:bytesReceived:packetsReceived:sendFractionLost:sendCumulativeLost:recvFractionLost:recvCumulativeLost:
  Obtain the RTCP statistics of audio channel.
- (int) - getVideoRtpStatistics:bytesSent:packetsSent:bytesReceived:packetsReceived:
  Obtain the RTP statistics 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 an 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 received from contact.
- (int) - presenceRejectSubscribe:
  Reject a presence SUBSCRIBE request that is 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 capturing devices.
- (int) - getVideoCaptureDeviceName:uniqueId:deviceName:
  Gets the name of a specific video capturing 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:
  Get the name of a specific recording device given by an index.
- (NSString *) - getPlayoutDeviceName:
  Get the name of a specific playout device given by an index.
- (int) - setSpeakerVolume:
  Set the speaker volume level,.
- (int) - getSpeakerVolume
  Gets the speaker volume.
- (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:

Used for the loop back test for audio device.

Properties

- id< PortSIPEventDelegate > **delegate**

Detailed Description

PortSIP VoIP SDK functions class.

Author:

Copyright (c) 2006-2016 PortSIP Solutions,Inc. All rights reserved.

Version:

11.2

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

Detailed Description

PortSIP VoIP SDK Video Render View class.

Author:

Copyright (c) 2006-2015 PortSIP Solutions,Inc. All rights reserved.

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

INDEX