PortSIP VoIP SDK Manual for Windows

Version 11.2.2
3/23/2015

Table of Contents

Welcome to the PortSIP VoIP SDK

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

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

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

Changes in this release

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

Getting Started

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

Contents

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

Web Site

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

Background

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

Frequently Asked Questions

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

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

2. How to compile the sample project?

```
      1. Download the sample projects from PortSIP website.
      2. Extract the .zip file.
      3. Open the project by your IDE:
         C#, VB.NET, VC++: Visual Studio 2008 or higher.
         Delphi: Delphi XE4 or higher.
      4. Compile the sample project directly, the trial version SDK allows 2-3 minutes
conversation.
```

3. How to create a new project base on PortSIP VoIP SDK

```
   C#/VB.NET:
      1) Download and extract the sample project for C#/VB.NET.
      2) Create a new "Windows application" project.
      3) Copy the PortSIP_sdk.dll to project output directories: bin and bin.
      4) Copy the "PortSIP" folder to project folder and add into Solution.
      5) Inherit the interface "SIPCallbackEvents" to process the callback events.
      6) Right click the project, choose "Properties", click "Build" tab, check the "Allow
unsafe code" checkbox.
```

```
      More details please read the Sample project source code.
```

```
   Delphi:
      1) Download and extract the sample project.
      2) Create a new "VCL Forms Application" project.
      3) Copy the PortSIP_sdk.dll to project output directories.
      4) Copy the "PortSIPLib" folder to project folder and add into this new project.
```

```
      More details please read the Sample project source code.
```

```
   VC++:
      1) Download and extract the sample project.
      2) Create a new "MFC Application" project.
      3) Copy the PortSIP_sdk.dll to project output directories.
      4) Copy the "include/PortSIPLib" folder to project folder and add the ".hxx" files into
project.
      5) Copy the "lib" folder to project folder and link "PortSIP_sdk.lib" into project.
```

```
      More details please read the Sample project source code.
```

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

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

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

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

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

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

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

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

5. Does the SDK is thread safe?

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

6. Does the SDK support native 64 bits?

```
Yes, the SDK support both 32bit and 64 bits.
```

Support

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

Module Index

Modules

Here is a list of all modules:

Namespace Index

Packages

Here are the packages with brief descriptions (if available):

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

Detailed Description

SDK functions

Initialize and register functions

Functions

- Int32 PortSIP.PortSIPLib.initialize (TRANSPORT_TYPE transportType, PORTSIP_LOG_LEVEL logLevel, String logFilePath, Int32 maxCallLines, String sipAgent, Int32 audioDeviceLayer, Int32 videoDeviceLayer)
  Initialize the SDK.
- void PortSIP.PortSIPLib.unInitialize ()
  Un-initialize the SDK and release resources.
- Int32 PortSIP.PortSIPLib.setUser (String userName, String displayName, String authName, String password, String localIp, Int32 localSipPort, String userDomain, String sipServer, Int32 sipServerPort, String stunServer, Int32 stunServerPort, String outboundServer, Int32 outboundServerPort)
  Set user account info.
- Int32 PortSIP.PortSIPLib.registerServer (Int32 expires, Int32 retryTimes)
  Register to SIP proxy server(login to server)
- Int32 PortSIP.PortSIPLib.unRegisterServer ()
  Un-register from the SIP proxy server.
- Int32 PortSIP.PortSIPLib.setLicenseKey (String key)

Set the license key, must called before setUser function.

Detailed Description

Initialize and register functions

Function Documentation

Int32 PortSIP.PortSIPLib.initialize (TRANSPORT_TYPE *transportType*, PORTSIP_LOG_LEVEL *logLevel*, String *logFilePath*, Int32 *maxCallLines*, String *sipAgent*, Int32 *audioDeviceLayer*, Int32 *videoDeviceLayer*)

Initialize the SDK.

Parameters:

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

Returns:

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

Int32 PortSIP.PortSIPLib.setUser (String *userName*, String *displayName*, String *authName*, String *password*, String *localIp*, Int32 *localSipPort*, String *userDomain*, String *sipServer*, Int32 *sipServerPort*, String *stunServer*, Int32 *stunServerPort*, String *outboundServer*, Int32 *outboundServerPort*)

Set user account info.

Parameters:

userName	Account(User name) of the SIP, usually provided by an IP-Telephony service provider.
displayName	The display name of user, you can set it as your like, such as "James Kend".

	It's optional.
authName	Authorization user name (usually equals the username).
password	The password of user, it's optional.
localIp	The local computer IP address to bind (for example: 192.168.1.108), it will be using for send and receive SIP message and RTP packet. If pass this IP as the IPv6 format then the SDK using IPv6.
localSipPort	The SIP message transport listener port(for example: 5060).
userDomain	User domain; this parameter is optional that allow pass a empty string if you are not use domain.
sipServer	SIP proxy server IP or domain(for example: xx.xxx.xx.x or sip.xxx.com).
sipServerPort	Port of the SIP proxy server, (for example: 5060).
stunServer	Stun server, use for NAT traversal, it's optional and can be pass empty string to disable STUN.
stunServerPort	STUN server port,it will be ignored if the outboundServer is empty.
outboundServer	Outbound proxy server(for example: sip.domain.com), it's optional that allow pass a empty string if not use outbound server.
outboundServerPort	Outbound proxy server port, it will be ignored if the outboundServer is empty.

Returns:

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

Int32 PortSIP.PortSIPLib.registerServer (Int32 *expires*, Int32 *retryTimes*)

Register to SIP proxy server(login to server)

Parameters:

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

Returns:

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

Int32 PortSIP.PortSIPLib.unRegisterServer ()

Un-register from the SIP proxy server.

Returns:

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

Int32 PortSIP.PortSIPLib.setLicenseKey (String *key*)

Set the license key, must called before setUser function.

Parameters:

key	The SDK license key, please purchase from PortSIP

Returns:

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

NIC and local IP functions

Functions

- Int32 PortSIP.PortSIPLib.getNICNums ()
  Get the Network Interface Card numbers.
- Int32 PortSIP.PortSIPLib.getLocalIpAddress (Int32 index, StringBuilder ip, Int32 ipSize)
  Get the local IP address by Network Interface Card index.

Detailed Description

Function Documentation

Int32 PortSIP.PortSIPLib.getNICNums ()

Get the Network Interface Card numbers.

Returns:

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

Int32 PortSIP.PortSIPLib.getLocalIpAddress (Int32 *index*, StringBuilder *ip*, Int32 *ipSize*)

Get the local IP address by Network Interface Card index.

Parameters:

index	The IP address index, for example, the PC has two NICs, we want to obtain the second NIC IP, then set this parameter 1. The first NIC IP index is 0.
ip	The buffer that to receives the IP.
ipSize	The IP buffer size, don't let it less than 32 bytes.

Returns:

If the function succeeds, the return value is0. If the function fails, the return value is a specific error code.

Audio and video codecs functions

Functions

- Int32 PortSIP.PortSIPLib.addAudioCodec (AUDIOCODEC_TYPE codecType)
  Enable an audio codec, it will be appears in SDP.
- Int32 PortSIP.PortSIPLib.addVideoCodec (VIDEOCODEC_TYPE codecType)
  Enable a video codec, it will be appears in SDP.
- Boolean PortSIP.PortSIPLib.isAudioCodecEmpty ()
  Detect enabled audio codecs is empty or not.
- Boolean PortSIP.PortSIPLib.isVideoCodecEmpty ()
  Detect enabled video codecs is empty or not.
- Int32 PortSIP.PortSIPLib.setAudioCodecPayloadType (AUDIOCODEC_TYPE codecType, Int32 payloadType)
  Set the RTP payload type for dynamic audio codec.
- Int32 PortSIP.PortSIPLib.setVideoCodecPayloadType (VIDEOCODEC_TYPE codecType, Int32 payloadType)
  Set the RTP payload type for dynamic Video codec.
- void PortSIP.PortSIPLib.clearAudioCodec ()
  Remove all enabled audio codecs.
- void PortSIP.PortSIPLib.clearVideoCodec ()
  Remove all enabled video codecs.
- Int32 PortSIP.PortSIPLib.setAudioCodecParameter (AUDIOCODEC_TYPE codecType, String parameter)
  Set the codec parameter for audio codec.
- Int32 PortSIP.PortSIPLib.setVideoCodecParameter (VIDEOCODEC_TYPE codecType, String parameter)
  Set the codec parameter for video codec.

Detailed Description

Function Documentation

Int32 PortSIP.PortSIPLib.addAudioCodec (AUDIOCODEC_TYPE *codecType*)

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

Parameters:

codecType	Audio codec type.

Returns:

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

Int32 PortSIP.PortSIPLib.addVideoCodec (VIDEOCODEC_TYPE *codecType*)

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

Parameters:

codecType	Video codec type.

Returns:

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

Boolean PortSIP.PortSIPLib.isAudioCodecEmpty ()

Detect enabled audio codecs is empty or not.

Returns:

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

Boolean PortSIP.PortSIPLib.isVideoCodecEmpty ()

Detect enabled video codecs is empty or not.

Returns:

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

Int32 PortSIP.PortSIPLib.setAudioCodecPayloadType (AUDIOCODEC_TYPE *codecType*, Int32 *payloadType*)

Set the RTP payload type for dynamic audio codec.

Parameters:

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

Returns:

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

Int32 PortSIP.PortSIPLib.setVideoCodecPayloadType (VIDEOCODEC_TYPE *codecType*, Int32 *payloadType*)

Set the RTP payload type for dynamic Video codec.

Parameters:

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

Returns:

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

Int32 PortSIP.PortSIPLib.setAudioCodecParameter (AUDIOCODEC_TYPE *codecType*, String *parameter*)

Set the codec parameter for audio codec.

Parameters:

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

Returns:

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

Remarks:

Example:

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

Int32 PortSIP.PortSIPLib.setVideoCodecParameter (VIDEOCODEC_TYPE *codecType*, String *parameter*)

Set the codec parameter for video codec.

Parameters:

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

Returns:

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

Remarks:

Example:

```
setVideoCodecParameter(VIDEO_CODEC_H264, "profile-level-id=420033;
packetization-mode=0");
```

Additional setting functions

Functions

- Int32 PortSIP.PortSIPLib.setDisplayName (String name)
  Set user display name.
- Int32 PortSIP.PortSIPLib.getVersion (out Int32 majorVersion, out Int32 minorVersion)
  Get the current version number of the SDK.
- Int32 PortSIP.PortSIPLib.enableReliableProvisional (Boolean enable)

Enable/disable PRACK.

- Int32 PortSIP.PortSIPLib.enable3GppTags (Boolean enable)
  Enable/disable the 3Gpp tags, include "ims.icsi.mmtel" and "g.3gpp.smsip".
- void PortSIP.PortSIPLib.enableCallbackSendingSignaling (Boolean enable)
  Enable/disable callback the sending SIP messages.
- Int32 PortSIP.PortSIPLib.setSrtpPolicy (SRTP_POLICY srtpPolicy)
  Set the SRTP policy.
- Int32 PortSIP.PortSIPLib.setRtpPortRange (Int32 minimumRtpAudioPort, Int32 maximumRtpAudioPort, Int32 minimumRtpVideoPort, Int32 maximumRtpVideoPort)
  Set the RTP ports range for audio and video streaming.
- Int32 PortSIP.PortSIPLib.setRtcpPortRange (Int32 minimumRtcpAudioPort, Int32 maximumRtcpAudioPort, Int32 minimumRtcpVideoPort, Int32 maximumRtcpVideoPort)
  Set the RTCP ports range for audio and video streaming.
- Int32 PortSIP.PortSIPLib.enableCallForward (Boolean forBusyOnly, String forwardTo)
  Enable call forward.
- Int32 PortSIP.PortSIPLib.disableCallForward ()
  Disable the call forward, the SDK is not forward any incoming call after this function is called.
- Int32 PortSIP.PortSIPLib.enableSessionTimer (Int32 timerSeconds, SESSION_REFRESH_MODE refreshMode)
  Allows to periodically refresh Session Initiation Protocol (SIP) sessions by sending repeated INVITE requests.
- Int32 PortSIP.PortSIPLib.disableSessionTimer ()
  Disable the session timer.
- void PortSIP.PortSIPLib.setDoNotDisturb (Boolean state)
  Enable the "Do not disturb" to enable/disable.
- Int32 PortSIP.PortSIPLib.detectMwi ()
  Use to obtain the MWI status.
- Int32 PortSIP.PortSIPLib.enableCheckMwi (Boolean state)
  Allows enable/disable the check MWI(Message Waiting Indication).
- Int32 PortSIP.PortSIPLib.setRtpKeepAlive (Boolean state, Int32 keepAlivePayloadType, Int32 deltaTransmitTimeMS)
  Enable or disable send RTP keep-alive packet during the call is established.
- Int32 PortSIP.PortSIPLib.setKeepAliveTime (Int32 keepAliveTime)
  Enable or disable send SIP keep-alive packet.
- Int32 PortSIP.PortSIPLib.setAudioSamples (Int32 ptime, Int32 maxPtime)
  Set the audio capture sample.
- Int32 PortSIP.PortSIPLib.addSupportedMimeType (String methodName, String mimeType, String subMimeType)
  Set the SDK receive the SIP message that include special mime type.

Detailed Description

Function Documentation

Int32 PortSIP.PortSIPLib.setDisplayName (String *name*)

Set user display name.

Parameters:

name	The display name.

Returns:

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

Int32 PortSIP.PortSIPLib.getVersion (out Int32 *majorVersion*, out Int32 *minorVersion*)

Get the current version number of the SDK.

Parameters:

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

Returns:

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

Int32 PortSIP.PortSIPLib.enableReliableProvisional (Boolean *enable*)

Enable/disable PRACK.

Parameters:

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

Returns:

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

Int32 PortSIP.PortSIPLib.enable3GppTags (Boolean *enable*)

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

Parameters:

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

Returns:

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

void PortSIP.PortSIPLib.enableCallbackSendingSignaling (Boolean *enable*)

Enable/disable callback the sending SIP messages.

Parameters:

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

Int32 PortSIP.PortSIPLib.setSrtpPolicy (SRTP_POLICY *srtpPolicy*)

Set the SRTP policy.

Parameters:

srtpPolicy	The SRTP policy.

Returns:

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

Int32 PortSIP.PortSIPLib.setRtpPortRange (Int32 *minimumRtpAudioPort*, Int32 *maximumRtpAudioPort*, Int32 *minimumRtpVideoPort*, Int32 *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, the return value is 0. If the function fails, the return value is a specific error code.

Remarks:

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

Int32 PortSIP.PortSIPLib.setRtcpPortRange (Int32 *minimumRtcpAudioPort*, Int32 *maximumRtcpAudioPort*, Int32 *minimumRtcpVideoPort*, Int32 *maximumRtcpVideoPort*)

Set the RTCP ports range for audio and video streaming.

Parameters:

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

Returns:

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

Remarks:

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

Int32 PortSIP.PortSIPLib.enableCallForward (Boolean *forBusyOnly*, String *forwardTo*)

Enable call forward.

Parameters:

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

Returns:

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

Int32 PortSIP.PortSIPLib.disableCallForward ()

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

Returns:

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

Int32 PortSIP.PortSIPLib.enableSessionTimer (Int32 *timerSeconds*, SESSION_REFRESH_MODE *refreshMode*)

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

Parameters:

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

Returns:

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

Remarks:

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

Int32 PortSIP.PortSIPLib.disableSessionTimer ()

Disable the session timer.

Returns:

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

void PortSIP.PortSIPLib.setDoNotDisturb (Boolean *state*)

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

Parameters:

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

Int32 PortSIP.PortSIPLib.detectMwi ()

Use to obtain the MWI status.

Returns:

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

Int32 PortSIP.PortSIPLib.enableCheckMwi (Boolean *state*)

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

Parameters:

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

Returns:

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

Int32 PortSIP.PortSIPLib.setRtpKeepAlive (Boolean *state*, Int32 *keepAlivePayloadType*, Int32 *deltaTransmitTimeMS*)

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

Parameters:

state	Set to true allow send the keep-alive packet during the conversation.
keepAlivePayloadType	The payload type of the keep-alive RTP packet, usually set to 126.
deltaTransmitTimeMS	The keep-alive RTP packet send interval, in millisecond, usually recommend 15000 - 300000.

Returns:

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

Int32 PortSIP.PortSIPLib.setKeepAliveTime (Int32 *keepAliveTime*)

Enable or disable send SIP keep-alive packet.

Parameters:

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

Returns:

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

Int32 PortSIP.PortSIPLib.setAudioSamples (Int32 *ptime*, Int32 *maxPtime*)

Set the audio capture sample.

Parameters:

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

Returns:

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

Remarks:

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

Int32 PortSIP.PortSIPLib.addSupportedMimeType (String *methodName*, String *mimeType*, String *subMimeType*)

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

Parameters:

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

Returns:

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

Remarks:

Default, the PortSIP VoIP SDK support these media types(mime types) that in the below incoming SIP messages:

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

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

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

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

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

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

Access SIP message header functions

Functions

- Int32 PortSIP.PortSIPLib.getExtensionHeaderValue (String sipMessage, String headerName, StringBuilder headerValue, Int32 headerValueLength)
  Access the SIP header of SIP message.
- Int32 PortSIP.PortSIPLib.addExtensionHeader (String headerName, String headerValue)
  Add the extension header(custom header) into every outgoing SIP message.
- Int32 PortSIP.PortSIPLib.clearAddExtensionHeaders ()
  Clear the added extension headers(custom headers)
- Int32 PortSIP.PortSIPLib.modifyHeaderValue (String headerName, String headerValue)
  Modify the special SIP header value for every outgoing SIP message.
- Int32 PortSIP.PortSIPLib.clearModifyHeaders ()
  Clear the modify headers value, no longer modify every outgoing SIP message header values.

Detailed Description

Function Documentation

Int32 PortSIP.PortSIPLib.getExtensionHeaderValue (String *sipMessage*, String *headerName*, StringBuilder *headerValue*, Int32 *headerValueLength*)

Access the SIP header of SIP message.

Parameters:

sipMessage	The SIP message.
headerName	Which header want to access of the SIP message.
headerValue	The buffer to receive header value.
headerValueLength	The headerValue buffer size. Usually we recommended set it more than 512 bytes.

Returns:

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

Remarks:

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

```
StringBuilder value = new StringBuilder();
value.Length = 512;
getExtensionHeaderValue(message, name, value);
```

Int32 PortSIP.PortSIPLib.addExtensionHeader (String *headerName*, String *headerValue*)

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

Parameters:

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

Returns:

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

Int32 PortSIP.PortSIPLib.clearAddExtensionHeaders ()

Clear the added extension headers(custom headers)

Returns:

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

Remarks:

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

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

```
clearAddextensionHeaders();
```

Int32 PortSIP.PortSIPLib.modifyHeaderValue (String *headerName*, String *headerValue*)

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

Parameters:

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

Returns:

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

Int32 PortSIP.PortSIPLib.clearModifyHeaders ()

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

Returns:

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

Remarks:

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

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

Audio and video functions

Functions

- Int32 PortSIP.PortSIPLib.setAudioDeviceId (Int32 recordingDeviceId, Int32 playoutDeviceId)
  Set the audio device that will use for audio call.
- Int32 PortSIP.PortSIPLib.setVideoDeviceId (Int32 deviceId)
  Set the video device that will use for video call.
- Int32 PortSIP.PortSIPLib.setVideoResolution (VIDEO_RESOLUTION resolution)
  Set the video capture resolution.
- Int32 PortSIP.PortSIPLib.setVideoBitrate (Int32 bitrateKbps)
  Set the video bit rate.
- Int32 PortSIP.PortSIPLib.setVideoFrameRate (Int32 frameRate)
  Set the video frame rate.
- Int32 PortSIP.PortSIPLib.sendVideo (Int32 sessionId, Boolean sendState)
  Send the video to remote side.
- Int32 PortSIP.PortSIPLib.setVideoOrientation (Int32 rotation)
  Changing the orientation of the video.

- void PortSIP.PortSIPLib.setLocalVideoWindow (IntPtr localVideoWindow)
  Set the the window that using to display the local video image.
- Int32 PortSIP.PortSIPLib.setRemoteVideoWindow (Int32 sessionId, IntPtr remoteVideoWindow)
  Set the window for a session that using to display the received remote video image.
- Int32 PortSIP.PortSIPLib.displayLocalVideo (Boolean state)
  Start/stop to display the local video image.
- Int32 PortSIP.PortSIPLib.setVideoNackStatus (Boolean state)
  Enable/disable the NACK feature(rfc6642) which help to improve the video quatliy.
- void PortSIP.PortSIPLib.muteMicrophone (Boolean mute)
  Mute the device microphone.it's unavailable for Android and iOS.
- void PortSIP.PortSIPLib.muteSpeaker (Boolean mute)
  Mute the device speaker, it's unavailable for Android and iOS.
- void PortSIP.PortSIPLib.getDynamicVolumeLevel (out Int32 speakerVolume, out Int32 microphoneVolume)
  Obtain the dynamic microphone volume level from current call.

Detailed Description

Function Documentation

Int32 PortSIP.PortSIPLib.setAudioDeviceId (Int32 *recordingDeviceId*, Int32 *playoutDeviceId*)

Set the audio device that will use for audio call.

Parameters:

recordingDeviceId	Device ID(index) for audio record.(Microphone).
playoutDeviceId	Device ID(index) for audio playback(Speaker).

Returns:

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

Int32 PortSIP.PortSIPLib.setVideoDeviceId (Int32 *deviceId*)

Set the video device that will use for video call.

Parameters:

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

Returns:

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

Int32 PortSIP.PortSIPLib.setVideoResolution (VIDEO_RESOLUTION *resolution*)

Set the video capture resolution.

Parameters:

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

Returns:

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

Int32 PortSIP.PortSIPLib.setVideoBitrate (Int32 *bitrateKbps*)

Set the video bit rate.

Parameters:

bitrateKbps	The video bit rate in KBPS.

Returns:

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

Int32 PortSIP.PortSIPLib.setVideoFrameRate (Int32 *frameRate*)

Set the video frame rate.

Parameters:

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

Returns:

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

Remarks:

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

Int32 PortSIP.PortSIPLib.sendVideo (Int32 *sessionId*, Boolean *sendState*)

Send the video to remote side.

Parameters:

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

Returns:

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

Int32 PortSIP.PortSIPLib.setVideoOrientation (Int32 *rotation*)

Changing the orientation of the video.

Parameters:

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

Returns:

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

void PortSIP.PortSIPLib.setLocalVideoWindow (IntPtr *localVideoWindow*)

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

Parameters:

localVideoWindow	The window to display local video image from camera.

Int32 PortSIP.PortSIPLib.setRemoteVideoWindow (Int32 *sessionId*, IntPtr *remoteVideoWindow*)

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

Parameters:

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

Returns:

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

Int32 PortSIP.PortSIPLib.displayLocalVideo (Boolean *state*)

Start/stop to display the local video image.

Parameters:

state	state Set to true to display local video iamge.

Returns:

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

Int32 PortSIP.PortSIPLib.setVideoNackStatus (Boolean *state*)

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

Parameters:

state	state Set to true to enable.

Returns:

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

void PortSIP.PortSIPLib.muteMicrophone (Boolean *mute*)

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

Parameters:

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

void PortSIP.PortSIPLib.muteSpeaker (Boolean *mute*)

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

Parameters:

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

void PortSIP.PortSIPLib.getDynamicVolumeLevel (out Int32 *speakerVolume*, out Int32 *microphoneVolume*)

Obtain the dynamic microphone volume level from current call.

Parameters:

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

Remarks:

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

Call functions

Functions

- Int32 PortSIP.PortSIPLib.call (String callee, Boolean sendSdp, Boolean videoCall)
  Make a call.
- Int32 PortSIP.PortSIPLib.rejectCall (Int32 sessionId, int code)
  rejectCall Reject the incoming call.
- Int32 PortSIP.PortSIPLib.hangUp (Int32 sessionId)
  hangUp Hang up the call.
- Int32 PortSIP.PortSIPLib.answerCall (Int32 sessionId, Boolean videoCall)
  answerCall Answer the incoming call.

- Int32 PortSIP.PortSIPLib.updateCall (Int32 sessionId, bool enableAudio, bool enableVideo)
  Use the re-INVITE to update the established call.
- Int32 PortSIP.PortSIPLib.hold (Int32 sessionId)
  To place a call on hold.
- Int32 PortSIP.PortSIPLib.unHold (Int32 sessionId)
  Take off hold.
- Int32 PortSIP.PortSIPLib.muteSession (Int32 sessionId, Boolean muteIncomingAudio, Boolean muteOutgoingAudio, Boolean muteIncomingVideo, Boolean muteOutgoingVideo)
  Mute the specified session audio or video.
- Int32 PortSIP.PortSIPLib.forwardCall (Int32 sessionId, String forwardTo)
  Forward call to another one when received the incoming call.
- Int32 PortSIP.PortSIPLib.sendDtmf (Int32 sessionId, DTMF_METHOD dtmfMethod, int code, int dtmfDuration, bool playDtmfTone)
  Send DTMF tone.

Detailed Description

Function Documentation

Int32 PortSIP.PortSIPLib.call (String *callee*, Boolean *sendSdp*, Boolean *videoCall*)

Make a call.

Parameters:

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

Returns:

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

Int32 PortSIP.PortSIPLib.rejectCall (Int32 *sessionId*, int *code*)

rejectCall Reject the incoming call.

Parameters:

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

Returns:

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

Int32 PortSIP.PortSIPLib.hangUp (Int32 *sessionId*)

hangUp Hang up the call.

Parameters:

sessionId	Session ID of the call.

Returns:

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

Int32 PortSIP.PortSIPLib.answerCall (Int32 *sessionId*, Boolean *videoCall*)

answerCall Answer the incoming call.

Parameters:

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

Returns:

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

Int32 PortSIP.PortSIPLib.updateCall (Int32 *sessionId*, bool *enableAudio*, bool *enableVideo*)

Use the re-INVITE to update the established call.

Parameters:

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

Returns:

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

Remarks:

Example usage:

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

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

Example 2: Remove video stream from currently conversation.

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

Int32 PortSIP.PortSIPLib.hold (Int32 *sessionId*)

To place a call on hold.

Parameters:

sessionId	The session ID of call.

Returns:

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

Int32 PortSIP.PortSIPLib.unHold (Int32 *sessionId*)

Take off hold.

Parameters:

sessionId	The session ID of call.

Returns:

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

Int32 PortSIP.PortSIPLib.muteSession (Int32 *sessionId*, Boolean *muteIncomingAudio*, Boolean *muteOutgoingAudio*, Boolean *muteIncomingVideo*, Boolean *muteOutgoingVideo*)

Mute the specified session audio or video.

Parameters:

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

Returns:

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

Int32 PortSIP.PortSIPLib.forwardCall (Int32 *sessionId*, String *forwardTo*)

Forward call to another one when received the incoming call.

Parameters:

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

Returns:

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

Int32 PortSIP.PortSIPLib.sendDtmf (Int32 *sessionId*, DTMF_METHOD *dtmfMethod*, int *code*, int *dtmfDuration*, bool *playDtmfTone*)

Send DTMF tone.

Parameters:

sessionId	The session ID of the call.
dtmfMethod	Support send DTMF tone with two methods: DTMF_RFC2833 and DTMF_INFO. The DTMF_RFC2833 is recommend.
code	The DTMF tone(0-16).

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

Parameters:

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

Returns:

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

Refer functions

Functions

- Int32 PortSIP.PortSIPLib.refer (Int32 sessionId, String referTo)
  Refer the currently call to another one.

- Int32 PortSIP.PortSIPLib.attendedRefer (Int32 sessionId, Int32 replaceSessionId, String referTo)
  Make an attended refer.
- Int32 PortSIP.PortSIPLib.acceptRefer (Int32 referId, String referSignalingMessage)
  Accept the REFER request, a new call will be make if called this function, usuall called after onReceivedRefer callback event.
- Int32 PortSIP.PortSIPLib.rejectRefer (Int32 referId)
  Reject the REFER request.

Detailed Description

Function Documentation

Int32 PortSIP.PortSIPLib.refer (Int32 *sessionId*, String *referTo*)

Refer the currently call to another one.

Parameters:

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

Returns:

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

Remarks:

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

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

Int32 PortSIP.PortSIPLib.attendedRefer (Int32 *sessionId*, Int32 *replaceSessionId*, String *referTo*)

Make an attended refer.

Parameters:

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

Returns:

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

Remarks:

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

Int32 PortSIP.PortSIPLib.acceptRefer (Int32 *referId*, String *referSignalingMessage*)

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

Parameters:

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

Returns:

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

Int32 PortSIP.PortSIPLib.rejectRefer (Int32 *referId*)

Reject the REFER request.

Parameters:

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

Returns:

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

Send audio and video stream functions

Functions

- Int32 PortSIP.PortSIPLib.enableSendPcmStreamToRemote (Int32 sessionId, Boolean state, Int32 streamSamplesPerSec)
  Enable the SDK send PCM stream data to remote side from another source to instread of microphone.
- Int32 PortSIP.PortSIPLib.sendPcmStreamToRemote (Int32 sessionId, byte[] data, Int32 dataLength)

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

- Int32 PortSIP.PortSIPLib.enableSendVideoStreamToRemote (Int32 sessionId, Boolean state)
  Enable the SDK send video stream data to remote side from another source to instread of camera.
- Int32 PortSIP.PortSIPLib.sendVideoStreamToRemote (Int32 sessionId, byte[] data, Int32 dataLength, Int32 width, Int32 height)
  Send the video stream to remote.

Detailed Description

Function Documentation

Int32 PortSIP.PortSIPLib.enableSendPcmStreamToRemote (Int32 *sessionId*, Boolean *state*, Int32 *streamSamplesPerSec*)

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

Parameters:

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

Returns:

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

Remarks:

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

Int32 PortSIP.PortSIPLib.sendPcmStreamToRemote (Int32 *sessionId*, byte[] *data*, Int32 *dataLength*)

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

Parameters:

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

Returns:

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

Remarks:

Usually we should use it like below:

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

Int32 PortSIP.PortSIPLib.enableSendVideoStreamToRemote (Int32 *sessionId*, Boolean *state*)

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

Parameters:

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

Returns:

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

Int32 PortSIP.PortSIPLib.sendVideoStreamToRemote (Int32 *sessionId*, byte[] *data*, Int32 *dataLength*, Int32 *width*, Int32 *height*)

Send the video stream to remote.

Parameters:

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

Returns:

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

Remarks:

Send the video stream in i420 from another source to instead of video device capture(camera). Before called this funtion,you MUST call the enableSendVideoStreamToRemote function. Usually we should use it like below:

```
enableSendVideoStreamToRemote(sessionId, true);
sendVideoStreamToRemote(sessionId, data, dataSize, 352, 288);
```

RTP packets, Audio stream and video stream callback functions

Functions

- Int32 PortSIP.PortSIPLib.setRtpCallback (Int32 callbackObject, Boolean enable)
  Set the RTP callbacks to allow access the sending and received RTP packets.
- Int32 PortSIP.PortSIPLib.enableAudioStreamCallback (Int32 callbackObject, Int32 sessionId, Boolean enable, AUDIOSTREAM_CALLBACK_MODE callbackMode)
  Enable/disable the audio stream callback.

- Int32 PortSIP.PortSIPLib.enableVideoStreamCallback (Int32 callbackObject, Int32 sessionId, VIDEOSTREAM_CALLBACK_MODE callbackMode)
  Enable/disable the video stream callback.

Detailed Description

Function Documentation

Int32 PortSIP.PortSIPLib.setRtpCallback (Int32 *callbackObject*, Boolean *enable*)

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

Parameters:

callbackObject	The callback object that you passed in and can access it once callback function triggered.
enable	Set to true to enable the RTP callback for received and sending RTP packets, the onSendingRtpPacket and onReceivedRtpPacket events will be triggered.

Returns:

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

Int32 PortSIP.PortSIPLib.enableAudioStreamCallback (Int32 *callbackObject*, Int32 *sessionId*, Boolean *enable*, AUDIOSTREAM_CALLBACK_MODE *callbackMode*)

Enable/disable the audio stream callback.

Parameters:

callbackObject	The callback object that you passed in and can access it once callback function triggered.
sessionId	The session ID of call.
enable	Set to true to enable audio stream callback, false to stop the callback.
callbackMode	The audio stream callback mode

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

Returns:

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

Remarks:

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

Int32 PortSIP.PortSIPLib.enableVideoStreamCallback (Int32 *callbackObject*, Int32 *sessionId*, VIDEOSTREAM_CALLBACK_MODE *callbackMode*)

Enable/disable the video stream callback.

Parameters:

callbackObject	The callback object that you passed in and can access it once callback function triggered.
sessionId	The session ID of call.
callbackMode	The video stream callback mode.

Mode	Description
VIDEOSTREAM_NONE	Disable video stream callback.
VIDEOSTREAM_LOCAL	Local video stream callback.
VIDEOSTREAM_REMOTE	Remote video stream callback.
VIDEOSTREAM_BOTH	Both of local and remote video stream callback.

Returns:

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

Remarks:

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

Record functions

Functions

- Int32 PortSIP.PortSIPLib.startRecord (Int32 sessionId, String recordFilePath, String recordFileName, Boolean appendTimestamp, AUDIO_RECORDING_FILEFORMAT audioFileFormat, RECORD_MODE audioRecordMode, VIDEOCODEC_TYPE videoFileCodecType, RECORD_MODE videoRecordMode)
  Start record the call.
- Int32 PortSIP.PortSIPLib.stopRecord (Int32 sessionId)
  Stop record.

Detailed Description

Function Documentation

Int32 PortSIP.PortSIPLib.startRecord (Int32 *sessionId*, String *recordFilePath*, String *recordFileName*, Boolean *appendTimestamp*, AUDIO_RECORDING_FILEFORMAT *audioFileFormat*, RECORD_MODE *audioRecordMode*, VIDEOCODEC_TYPE *videoFileCodecType*, RECORD_MODE *videoRecordMode*)

Start record the call.

Parameters:

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

Returns:

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

Int32 PortSIP.PortSIPLib.stopRecord (Int32 *sessionId*)

Stop record.

Parameters:

sessionId	The session ID of call conversation.

Returns:

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

Play audio and video file to remoe functions

Functions

- Int32 PortSIP.PortSIPLib.playVideoFileToRemote (Int32 sessionId, String fileName, Boolean loop, Boolean playAudio)
  Play an AVI file to remote party.
- Int32 PortSIP.PortSIPLib.stopPlayVideoFileToRemote (Int32 sessionId)
  Stop play video file to remote side.
- Int32 PortSIP.PortSIPLib.playAudioFileToRemote (Int32 sessionId, String fileName, Int32 fileSamplesPerSec, Boolean loop)
  Play an wave file to remote party.

- Int32 PortSIP.PortSIPLib.stopPlayAudioFileToRemote (Int32 sessionId)
  Stop play wave file to remote side.
- Int32 PortSIP.PortSIPLib.playAudioFileToRemoteAsBackground (Int32 sessionId, String fileName, Int32 fileSamplesPerSec)
  Play an wave file to remote party as conversation background sound.
- Int32 PortSIP.PortSIPLib.stopPlayAudioFileToRemoteAsBackground (Int32 sessionId)
  Stop play an wave file to remote party as conversation background sound.

Detailed Description

Function Documentation

Int32 PortSIP.PortSIPLib.playVideoFileToRemote (Int32 *sessionId*, String *fileName*, Boolean *loop*, Boolean *playAudio*)

Play an AVI file to remote party.

Parameters:

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

Returns:

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

Int32 PortSIP.PortSIPLib.stopPlayVideoFileToRemote (Int32 *sessionId*)

Stop play video file to remote side.

Parameters:

sessionId	Session ID of the call.

Returns:

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

Int32 PortSIP.PortSIPLib.playAudioFileToRemote (Int32 *sessionId*, String *fileName*, Int32 *fileSamplesPerSec*, Boolean *loop*)

Play an wave file to remote party.

Parameters:

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

Returns:

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

Int32 PortSIP.PortSIPLib.stopPlayAudioFileToRemote (Int32 *sessionId*)

Stop play wave file to remote side.

Parameters:

sessionId	Session ID of the call.

Returns:

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

Int32 PortSIP.PortSIPLib.playAudioFileToRemoteAsBackground (Int32 *sessionId*, String *fileName*, Int32 *fileSamplesPerSec*)

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

Parameters:

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

Returns:

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

Int32 PortSIP.PortSIPLib.stopPlayAudioFileToRemoteAsBackground (Int32 *sessionId*)

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

Parameters:

sessionId	Session ID of the call.

Returns:

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

Conference functions

Functions

- Int32 PortSIP.PortSIPLib.createConference (IntPtr conferenceVideoWindow, VIDEO_RESOLUTION videoResolution, Boolean displayLocalVideoInConference)
  Create a conference. It's failures if the exists conference isn't destroy yet.
- void PortSIP.PortSIPLib.destroyConference ()
  Destroy the exist conference.
- Int32 PortSIP.PortSIPLib.setConferenceVideoWindow (IntPtr videoWindow)
  Set the window for a conference that using to display the received remote video image.
- Int32 PortSIP.PortSIPLib.joinToConference (Int32 sessionId)
  Join a session into exist conference, if the call is in hold, it will be un-hold automatically.
- Int32 PortSIP.PortSIPLib.removeFromConference (Int32 sessionId)
  Remove a session from an exist conference.

Detailed Description

Function Documentation

Int32 PortSIP.PortSIPLib.createConference (IntPtr *conferenceVideoWindow*, VIDEO_RESOLUTION *videoResolution*, Boolean *displayLocalVideoInConference*)

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

Parameters:

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

Returns:

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

Int32 PortSIP.PortSIPLib.setConferenceVideoWindow (IntPtr *videoWindow*)

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

Parameters:

videoWindow	The UIView which using to display the conference video.

Returns:

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

Int32 PortSIP.PortSIPLib.joinToConference (Int32 *sessionId*)

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

Parameters:

sessionId	Session ID of the call.

Returns:

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

Int32 PortSIP.PortSIPLib.removeFromConference (Int32 *sessionId*)

Remove a session from an exist conference.

Parameters:

sessionId	Session ID of the call.

Returns:

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

RTP and RTCP QOS functions

Functions

- Int32 PortSIP.PortSIPLib.setAudioRtcpBandwidth (Int32 sessionId, Int32 BitsRR, Int32 BitsRS, Int32 KBitsAS)
  Set the audio RTCP bandwidth parameters as the RFC3556.
- Int32 PortSIP.PortSIPLib.setVideoRtcpBandwidth (Int32 sessionId, Int32 BitsRR, Int32 BitsRS, Int32 KBitsAS)
  Set the video RTCP bandwidth parameters as the RFC3556.
- Int32 PortSIP.PortSIPLib.setAudioQos (Boolean state, Int32 DSCPValue, Int32 priority)
  Set the DSCP(differentiated services code point) value of QoS(Quality of Service) for audio channel.
- Int32 PortSIP.PortSIPLib.setVideoQos (Boolean state, Int32 DSCPValue)
  Set the DSCP(differentiated services code point) value of QoS(Quality of Service) for video channel.

Detailed Description

Function Documentation

Int32 PortSIP.PortSIPLib.setAudioRtcpBandwidth (Int32 *sessionId*, Int32 *BitsRR*, Int32 *BitsRS*, Int32 *KBitsAS*)

Set the audio RTCP bandwidth parameters as the RFC3556.

Parameters:

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

Returns:

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

Int32 PortSIP.PortSIPLib.setVideoRtcpBandwidth (Int32 *sessionId*, Int32 *BitsRR*, Int32 *BitsRS*, Int32 *KBitsAS*)

Set the video RTCP bandwidth parameters as the RFC3556.

Parameters:

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

Returns:

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

Int32 PortSIP.PortSIPLib.setAudioQos (Boolean *state*, Int32 *DSCPValue*, Int32 *priority*)

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

Parameters:

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

Returns:

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

Int32 PortSIP.PortSIPLib.setVideoQos (Boolean *state*, Int32 *DSCPValue*)

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

Parameters:

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

Returns:

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

RTP statistics functions

Functions

- Int32 PortSIP.PortSIPLib.getNetworkStatistics (Int32 sessionId, out Int32 currentBufferSize, out Int32 preferredBufferSize, out Int32 currentPacketLossRate, out Int32 currentDiscardRate, out Int32 currentExpandRate, out Int32 currentPreemptiveRate, out Int32 currentAccelerateRate)
  Get the "in-call" statistics. The statistics are reset after the query.
- Int32 PortSIP.PortSIPLib.getAudioRtpStatistics (Int32 sessionId, out Int32 averageJitterMs, out Int32 maxJitterMs, out Int32 discardedPackets)
  Obtain the RTP statisics of audio channel.
- Int32 PortSIP.PortSIPLib.getAudioRtcpStatistics (Int32 sessionId, out Int32 bytesSent, out Int32 packetsSent, out Int32 bytesReceived, out Int32 packetsReceived, out Int32 sendFractionLost, out Int32 sendCumulativeLost, out Int32 recvFractionLost, out Int32 recvCumulativeLost)
  Obtain the RTCP statisics of audio channel.
- Int32 PortSIP.PortSIPLib.getVideoRtpStatistics (Int32 sessionId, out Int32 bytesSent, out Int32 packetsSent, out Int32 bytesReceived, out Int32 packetsReceived)
  Obtain the RTP statisics of video.

Detailed Description

Function Documentation

Int32 PortSIP.PortSIPLib.getNetworkStatistics (Int32 *sessionId*, out Int32 *currentBufferSize*, out Int32 *preferredBufferSize*, out Int32 *currentPacketLossRate*, out Int32 *currentDiscardRate*, out Int32 *currentExpandRate*, out Int32 *currentPreemptiveRate*, out Int32 *currentAccelerateRate*)

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

Parameters:

sessionId	The session ID of call conversation.

currentBufferSize	Current jitter buffer size in ms.
preferredBufferSize	Preferred (optimal) buffer size in ms.
currentPacketLossRate	Loss rate (network + late) in percent.
currentDiscardRate	Late loss rate in percent.
currentExpandRate	Fraction (of original stream) of synthesized speech inserted through expansion.
currentPreemptiveRate	Fraction of synthesized speech inserted through pre-emptive expansion.
currentAccelerateRate	Fraction of data removed through acceleration through acceleration.

Returns:

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

Int32 PortSIP.PortSIPLib.getAudioRtpStatistics (Int32 *sessionId*, out Int32 *averageJitterMs*, out Int32 *maxJitterMs*, out Int32 *discardedPackets*)

Obtain the RTP statisics of audio channel.

Parameters:

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

Returns:

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

Int32 PortSIP.PortSIPLib.getAudioRtcpStatistics (Int32 *sessionId*, out Int32 *bytesSent*, out Int32 *packetsSent*, out Int32 *bytesReceived*, out Int32 *packetsReceived*, out Int32 *sendFractionLost*, out Int32 *sendCumulativeLost*, out Int32 *recvFractionLost*, out Int32 *recvCumulativeLost*)

Obtain the RTCP statisics of audio channel.

Parameters:

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

Returns:

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

Int32 PortSIP.PortSIPLib.getVideoRtpStatistics (Int32 *sessionId*, out Int32 *bytesSent*, out Int32 *packetsSent*, out Int32 *bytesReceived*, out Int32 *packetsReceived*)

Obtain the RTP statisics of video.

Parameters:

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

Returns:

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

Audio effect functions

Functions

- void PortSIP.PortSIPLib.enableVAD (Boolean state)
  Enable/disable Voice Activity Detection(VAD).
- void PortSIP.PortSIPLib.enableAEC (EC_MODES ecMode)
  Enable/disable AEC (Acoustic Echo Cancellation).
- void PortSIP.PortSIPLib.enableCNG (Boolean state)
  Enable/disable Comfort Noise Generator(CNG).
- void PortSIP.PortSIPLib.enableAGC (AGC_MODES agcMode)
  Enable/disable Automatic Gain Control(AGC).
- void PortSIP.PortSIPLib.enableANS (NS_MODES nsMode)
  Enable/disable Audio Noise Suppression(ANS).

Detailed Description

Function Documentation

void PortSIP.PortSIPLib.enableVAD (Boolean *state*)

Enable/disable Voice Activity Detection(VAD).

Parameters:

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

void PortSIP.PortSIPLib.enableAEC (EC_MODES *ecMode*)

Enable/disable AEC (Acoustic Echo Cancellation).

Parameters:

ecMode	Allow set the AEC mode to effect for different scenarios.

Mode	Description
EC_NONE	Disable AEC.
EC_DEFAULT	Platform default AEC.
EC_CONFERENCE	Desktop platform(windows,MAC) Conferencing default (aggressive AEC).

void PortSIP.PortSIPLib.enableCNG (Boolean *state*)

Enable/disable Comfort Noise Generator(CNG).

Parameters:

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

void PortSIP.PortSIPLib.enableAGC (AGC_MODES *agcMode*)

Enable/disable Automatic Gain Control(AGC).

Parameters:

agcMode	Allow set the AGC mode to effect for different scenarios.

Mode	Description
AGC_DEFAULT	Disable AGC.
AGC_DEFAULT	Platform default.
AGC_ADAPTIVE_ANALOG	Desktop platform(windows,MAC) adaptive mode for use when analog volume control exists.
AGC_ADAPTIVE_DIGITAL	scaling takes place in the digital domain (e.g. for conference servers and embedded devices).
AGC_FIXED_DIGITAL	can be used on embedded devices where the capture signal level is predictable.

void PortSIP.PortSIPLib.enableANS (NS_MODES *nsMode*)

Enable/disable Audio Noise Suppression(ANS).

Parameters:

nsMode	Allow set the NS mode to effect for different scenarios.

Mode	Description
NS_NONE	Disable NS.
NS_DEFAULT	Platform default.
NS_Conference	conferencing default.
NS_LOW_SUPPRESSION	lowest suppression.
NS_MODERATE_SUPPRESSION	moderate suppression.
NS_HIGH_SUPPRESSION	high suppression
NS_VERY_HIGH_SUPPRESSION	highest suppression.

Send OPTIONS/INFO/MESSAGE functions

Functions

- Int32 PortSIP.PortSIPLib.sendOptions (String to, String sdp)
  Send OPTIONS message.
- Int32 PortSIP.PortSIPLib.sendInfo (Int32 sessionId, String mimeType, String subMimeType, String infoContents)
  Send a INFO message to remote side in dialog.
- Int32 PortSIP.PortSIPLib.sendMessage (Int32 sessionId, String mimeType, String subMimeType, byte[] message, Int32 messageLength)
  Send a MESSAGE message to remote side in dialog.
- Int32 PortSIP.PortSIPLib.sendOutOfDialogMessage (String to, String mimeType, String subMimeType, byte[] message, Int32 messageLength)
  Send a out of dialog MESSAGE message to remote side.

Detailed Description

Function Documentation

Int32 PortSIP.PortSIPLib.sendOptions (String *to*, String *sdp*)

Send OPTIONS message.

Parameters:

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

Returns:

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

Int32 PortSIP.PortSIPLib.sendInfo (Int32 *sessionId*, String *mimeType*, String *subMimeType*, String *infoContents*)

Send a INFO message to remote side in dialog.

Parameters:

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

Returns:

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

Int32 PortSIP.PortSIPLib.sendMessage (Int32 *sessionId*, String *mimeType*, String *subMimeType*, byte[] *message*, Int32 *messageLength*)

Send a MESSAGE message to remote side in dialog.

Parameters:

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

Returns:

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

Remarks:

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

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

Example 2: send a binary message.

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

Int32 PortSIP.PortSIPLib.sendOutOfDialogMessage (String *to*, String *mimeType*, String *subMimeType*, byte[] *message*, Int32 *messageLength*)

Send a out of dialog MESSAGE message to remote side.

Parameters:

to	The message receiver. Likes sip:receiver@portsip.com

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

Returns:

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

Remarks:

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

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

Example 2: send a binary message.

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

Presence functions

Functions

- Int32 PortSIP.PortSIPLib.presenceSubscribeContact (String contact, String subject)
  Send a SUBSCRIBE message for presence to a contact.
- Int32 PortSIP.PortSIPLib.presenceRejectSubscribe (Int32 subscribeId)
  Accept the presence SUBSCRIBE request which received from contact.
- Int32 PortSIP.PortSIPLib.presenceAcceptSubscribe (Int32 subscribeId)
  Reject a presence SUBSCRIBE request which received from contact.
- Int32 PortSIP.PortSIPLib.presenceOnline (Int32 subscribeId, String stateText)
  Send a NOTIFY message to contact to notify that presence status is online/changed.
- Int32 PortSIP.PortSIPLib.presenceOffline (Int32 subscribeId)
  Send a NOTIFY message to contact to notify that presence status is offline.

Detailed Description

Function Documentation

Int32 PortSIP.PortSIPLib.presenceSubscribeContact (String *contact*, String *subject*)

Send a SUBSCRIBE message for presence to a contact.

Parameters:

contact	The target contact, it must likes sip:contact001@sip.portsip.com.
subject	This subject text will be insert into the SUBSCRIBE message. For example:

	"Hello, I'm Jason". The subject maybe is UTF8 format, you should use UTF8 to decode it.

Returns:

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

Int32 PortSIP.PortSIPLib.presenceRejectSubscribe (Int32 *subscribeId*)

Accept the presence SUBSCRIBE request which received from contact.

Parameters:

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

Returns:

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

Int32 PortSIP.PortSIPLib.presenceAcceptSubscribe (Int32 *subscribeId*)

Reject a presence SUBSCRIBE request which received from contact.

Parameters:

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

Returns:

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

Int32 PortSIP.PortSIPLib.presenceOnline (Int32 *subscribeId*, String *stateText*)

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

Parameters:

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

Returns:

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

Int32 PortSIP.PortSIPLib.presenceOffline (Int32 *subscribeId*)

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

Parameters:

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

Returns:

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

Device Manage functions.

Functions

- Int32 PortSIP.PortSIPLib.getNumOfRecordingDevices ()
  Gets the number of audio devices available for audio recording.
- Int32 PortSIP.PortSIPLib.getNumOfPlayoutDevices ()
  Gets the number of audio devices available for audio playout.
- Int32 PortSIP.PortSIPLib.getRecordingDeviceName (Int32 deviceIndex, StringBuilder nameUTF8, Int32 nameUTF8Length)
  Gets the name of a specific recording device given by an index.
- Int32 PortSIP.PortSIPLib.getPlayoutDeviceName (Int32 deviceIndex, StringBuilder nameUTF8, Int32 nameUTF8Length)
  Gets the name of a specific playout device given by an index.
- Int32 PortSIP.PortSIPLib.setSpeakerVolume (Int32 volume)
  Set the speaker volume level,.
- Int32 PortSIP.PortSIPLib.getSpeakerVolume ()
  Gets the speaker volume level.
- Int32 PortSIP.PortSIPLib.setSystemOutputMute (Boolean enable)
  Mutes the speaker device completely in the OS.
- Boolean PortSIP.PortSIPLib.getSystemOutputMute ()
  Retrieves the output device mute state in the operating system.
- Int32 PortSIP.PortSIPLib.setMicVolume (Int32 volume)
  Sets the microphone volume level.
- Int32 PortSIP.PortSIPLib.getMicVolume ()
  Retrieves the current microphone volume.
- Int32 PortSIP.PortSIPLib.setSystemInputMute (Boolean enable)
  Mute the microphone input device completely in the OS.
- Boolean PortSIP.PortSIPLib.getSystemInputMute ()
  Gets the mute state of the input device in the operating system.
- void PortSIP.PortSIPLib.audioPlayLoopbackTest (Boolean enable)
  Use to do the audio device loop back test.
- Int32 PortSIP.PortSIPLib.getNumOfVideoCaptureDevices ()
  Gets the number of available capture devices.
- Int32 PortSIP.PortSIPLib.getVideoCaptureDeviceName (Int32 deviceIndex, StringBuilder uniqueIdUTF8, Int32 uniqueIdUTF8Length, StringBuilder deviceNameUTF8, Int32 deviceNameUTF8Length)
  Gets the name of a specific video capture device given by an index.
- Int32 PortSIP.PortSIPLib.showVideoCaptureSettingsDialogBox (String uniqueIdUTF8, Int32 uniqueIdUTF8Length, String dialogTitle, IntPtr parentWindow, Int32 x, Int32 y)

Display the capture device property dialog box for the specified capture device.

Detailed Description

Function Documentation

Int32 PortSIP.PortSIPLib.getNumOfRecordingDevices ()

Gets the number of audio devices available for audio recording.

Returns:

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

Int32 PortSIP.PortSIPLib.getNumOfPlayoutDevices ()

Gets the number of audio devices available for audio playout.

Returns:

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

Int32 PortSIP.PortSIPLib.getRecordingDeviceName (Int32 *deviceIndex*, StringBuilder *nameUTF8*, Int32 *nameUTF8Length*)

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

Parameters:

deviceIndex	Device index (0, 1, 2, ..., N-1), where N is given by getNumOfRecordingDevices (). Also -1 is a valid value and will return the name of the default recording device.
nameUTF8	A character buffer to which the device name will be copied as a null-terminated string in UTF8 format.
nameUTF8Length	The size of nameUTF8 buffer, don't let it less than 128.

Returns:

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

Int32 PortSIP.PortSIPLib.getPlayoutDeviceName (Int32 *deviceIndex*, StringBuilder *nameUTF8*, Int32 *nameUTF8Length*)

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

Parameters:

deviceIndex	
deviceIndex	Device index (0, 1, 2, ..., N-1), where N is given by getNumOfRecordingDevices (). Also -1 is a valid value and will return the name of the default recording device.
nameUTF8	A character buffer to which the device name will be copied as a null-terminated string in UTF8 format.
nameUTF8Length	The size of nameUTF8 buffer, don't let it less than 128.

Returns:

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

Int32 PortSIP.PortSIPLib.setSpeakerVolume (Int32 *volume*)

Set the speaker volume level,.

Parameters:

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

Returns:

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

Int32 PortSIP.PortSIPLib.getSpeakerVolume ()

Gets the speaker volume level.

Returns:

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

Int32 PortSIP.PortSIPLib.setSystemOutputMute (Boolean *enable*)

Mutes the speaker device completely in the OS.

Parameters:

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

Returns:

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

Boolean PortSIP.PortSIPLib.getSystemOutputMute ()

Retrieves the output device mute state in the operating system.

Returns:

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

Int32 PortSIP.PortSIPLib.setMicVolume (Int32 *volume*)

Sets the microphone volume level.

Parameters:

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

Returns:

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

Int32 PortSIP.PortSIPLib.getMicVolume ()

Retrieves the current microphone volume.

Returns:

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

Int32 PortSIP.PortSIPLib.setSystemInputMute (Boolean *enable*)

Mute the microphone input device completely in the OS.

Parameters:

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

Returns:

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

Boolean PortSIP.PortSIPLib.getSystemInputMute ()

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

Returns:

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

void PortSIP.PortSIPLib.audioPlayLoopbackTest (Boolean *enable*)

Use to do the audio device loop back test.

Parameters:

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

Int32 PortSIP.PortSIPLib.getNumOfVideoCaptureDevices ()

Gets the number of available capture devices.

Returns:

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

Int32 PortSIP.PortSIPLib.getVideoCaptureDeviceName (Int32 *deviceIndex*, StringBuilder *uniqueIdUTF8*, Int32 *uniqueIdUTF8Length*, StringBuilder *deviceNameUTF8*, Int32 *deviceNameUTF8Length*)

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

Parameters:

deviceIndex	Device index (0, 1, 2, ..., N-1), where N is given by getNumOfVideoCaptureDevices (). Also -1 is a valid value and will return the name of the default capture device.
uniqueIdUTF8	Unique identifier of the capture device.
uniqueIdUTF8Length	Size in bytes of uniqueIdUTF8.
deviceNameUTF8	A character buffer to which the device name will be copied as a null-terminated string in UTF8 format.
deviceNameUTF8 Length	The size of nameUTF8 buffer, don't let it less than 128.

Returns:

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

Int32 PortSIP.PortSIPLib.showVideoCaptureSettingsDialogBox (String *uniqueIdUTF8*, Int32 *uniqueIdUTF8Length*, String *dialogTitle*, IntPtr *parentWindow*, Int32 *x*, Int32 *y*)

Display the capture device property dialog box for the specified capture device.

Parameters:

uniqueIdUTF8	Unique identifier of the capture device.
uniqueIdUTF8Length	Size in bytes of uniqueIdUTF8.
dialogTitle	The title of the video settings dialog.
parentWindow	Parent window to use for the dialog box, should originally be a HWND.
x	Horizontal position for the dialog relative to the parent window, in pixels.
y	Vertical position for the dialog relative to the parent window, in pixels.

Returns:

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

SDK Callback events

Modules

- Register events
- Call events
- Refer events
- Signaling events
- MWI events
- DTMF events
- INFO/OPTIONS message events
- Presence events
- Play audio and video file finished events
- RTP callback events
- Audio and video stream callback events

Detailed Description

SDK Callback events

Register events

Functions

- Int32 PortSIP.SIPCallbackEvents.onRegisterSuccess (Int32 callbackIndex, Int32 callbackObject, String statusText, Int32 statusCode)
- Int32 PortSIP.SIPCallbackEvents.onRegisterFailure (Int32 callbackIndex, Int32 callbackObject, String statusText, Int32 statusCode)

Detailed Description

Register events

Function Documentation

Int32 PortSIP.SIPCallbackEvents.onRegisterSuccess (Int32 *callbackIndex*, Int32 *callbackObject*, String *statusText*, Int32 *statusCode*)

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

Parameters:

callbackIndex	This is a callback index which passed in when create the SDK library.
callbackObject	This is a callback object which passed in when create the SDK library.
statusText	The status text.
statusCode	The status code.

Int32 PortSIP.SIPCallbackEvents.onRegisterFailure (Int32 *callbackIndex*, Int32 *callbackObject*, String *statusText*, Int32 *statusCode*)

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

Parameters:

statusText	The status text.
statusCode	The status code.

Call events

Functions

- Int32 PortSIP.SIPCallbackEvents.onInviteIncoming (Int32 callbackIndex, Int32 callbackObject, Int32 sessionId, String callerDisplayName, String caller, String calleeDisplayName, String callee, String audioCodecNames, String videoCodecNames, Boolean existsAudio, Boolean existsVideo)
- Int32 PortSIP.SIPCallbackEvents.onInviteTrying (Int32 callbackIndex, Int32 callbackObject, Int32 sessionId)
- Int32 PortSIP.SIPCallbackEvents.onInviteSessionProgress (Int32 callbackIndex, Int32 callbackObject, Int32 sessionId, String audioCodecNames, String videoCodecNames, Boolean existsEarlyMedia, Boolean existsAudio, Boolean existsVideo)
- Int32 PortSIP.SIPCallbackEvents.onInviteRinging (Int32 callbackIndex, Int32 callbackObject, Int32 sessionId, String statusText, Int32 statusCode)
- Int32 PortSIP.SIPCallbackEvents.onInviteAnswered (Int32 callbackIndex, Int32 callbackObject, Int32 sessionId, String callerDisplayName, String caller, String calleeDisplayName, String callee, String audioCodecNames, String videoCodecNames, Boolean existsAudio, Boolean existsVideo)
- Int32 PortSIP.SIPCallbackEvents.onInviteFailure (Int32 callbackIndex, Int32 callbackObject, Int32 sessionId, String reason, Int32 code)
- Int32 PortSIP.SIPCallbackEvents.onInviteUpdated (Int32 callbackIndex, Int32 callbackObject, Int32 sessionId, String audioCodecNames, String videoCodecNames, Boolean existsAudio, Boolean existsVideo)
- Int32 PortSIP.SIPCallbackEvents.onInviteConnected (Int32 callbackIndex, Int32 callbackObject, Int32 sessionId)
- Int32 PortSIP.SIPCallbackEvents.onInviteBeginingForward (Int32 callbackIndex, Int32 callbackObject, String forwardTo)
- Int32 PortSIP.SIPCallbackEvents.onInviteClosed (Int32 callbackIndex, Int32 callbackObject, Int32 sessionId)
- Int32 PortSIP.SIPCallbackEvents.onRemoteHold (Int32 callbackIndex, Int32 callbackObject, Int32 sessionId)
- Int32 PortSIP.SIPCallbackEvents.onRemoteUnHold (Int32 callbackIndex, Int32 callbackObject, Int32 sessionId, String audioCodecNames, String videoCodecNames, Boolean existsAudio, Boolean existsVideo)

Detailed Description

Function Documentation

Int32 PortSIP.SIPCallbackEvents.onInviteIncoming (Int32 *callbackIndex*, Int32 *callbackObject*, Int32 *sessionId*, String *callerDisplayName*, String *caller*, String

***calleeDisplayName*, String *callee*, String *audioCodecNames*, String *videoCodecNames*, Boolean *existsAudio*, Boolean *existsVideo*)**

When the call is coming, this event was triggered.

Parameters:

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

Int32 PortSIP.SIPCallbackEvents.onInviteTrying (Int32 *callbackIndex*, Int32 *callbackObject*, Int32 *sessionId*)

If the outgoing call was processing, this event triggered.

Parameters:

sessionId	The session ID of the call.

Int32 PortSIP.SIPCallbackEvents.onInviteSessionProgress (Int32 *callbackIndex*, Int32 *callbackObject*, Int32 *sessionId*, String *audioCodecNames*, String *videoCodecNames*, Boolean *existsEarlyMedia*, Boolean *existsAudio*, Boolean *existsVideo*)

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

Parameters:

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

Int32 PortSIP.SIPCallbackEvents.onInviteRinging (Int32 *callbackIndex*, Int32 *callbackObject*, Int32 *sessionId*, String *statusText*, Int32 *statusCode*)

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

Parameters:

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

Int32 PortSIP.SIPCallbackEvents.onInviteAnswered (Int32 *callbackIndex*, Int32 *callbackObject*, Int32 *sessionId*, String *callerDisplayName*, String *caller*, String *calleeDisplayName*, String *callee*, String *audioCodecNames*, String *videoCodecNames*, Boolean *existsAudio*, Boolean *existsVideo*)

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

Parameters:

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

Int32 PortSIP.SIPCallbackEvents.onInviteFailure (Int32 *callbackIndex*, Int32 *callbackObject*, Int32 *sessionId*, String *reason*, Int32 *code*)

If the outgoing call is fails, this event triggered.

Parameters:

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

Int32 PortSIP.SIPCallbackEvents.onInviteUpdated (Int32 *callbackIndex*, Int32 *callbackObject*, Int32 *sessionId*, String *audioCodecNames*, String *videoCodecNames*, Boolean *existsAudio*, Boolean *existsVideo*)

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

Parameters:

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

Int32 PortSIP.SIPCallbackEvents.onInviteConnected (Int32 *callbackIndex*, Int32 *callbackObject*, Int32 *sessionId*)

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

Parameters:

sessionId	The session ID of the call.

Int32 PortSIP.SIPCallbackEvents.onInviteBeginingForward (Int32 *callbackIndex*, Int32 *callbackObject*, String *forwardTo*)

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

Parameters:

forwardTo	The forward target SIP URI.

Int32 PortSIP.SIPCallbackEvents.onInviteClosed (Int32 *callbackIndex*, Int32 *callbackObject*, Int32 *sessionId*)

This event is triggered once remote side close the call.

Parameters:

sessionId	The session ID of the call.

Int32 PortSIP.SIPCallbackEvents.onRemoteHold (Int32 *callbackIndex*, Int32 *callbackObject*, Int32 *sessionId*)

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

Parameters:

sessionId	The session ID of the call.

Int32 PortSIP.SIPCallbackEvents.onRemoteUnHold (Int32 *callbackIndex*, Int32 *callbackObject*, Int32 *sessionId*, String *audioCodecNames*, String *videoCodecNames*, Boolean *existsAudio*, Boolean *existsVideo*)

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

Parameters:

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

Refer events

Functions

- Int32 PortSIP.SIPCallbackEvents.onReceivedRefer (Int32 callbackIndex, Int32 callbackObject, Int32 sessionId, Int32 referId, String to, String from, String referSipMessage)
- Int32 PortSIP.SIPCallbackEvents.onReferAccepted (Int32 callbackIndex, Int32 callbackObject, Int32 sessionId)
- Int32 PortSIP.SIPCallbackEvents.onReferRejected (Int32 callbackIndex, Int32 callbackObject, Int32 sessionId, String reason, Int32 code)
- Int32 PortSIP.SIPCallbackEvents.onTransferTrying (Int32 callbackIndex, Int32 callbackObject, Int32 sessionId)
- Int32 PortSIP.SIPCallbackEvents.onTransferRinging (Int32 callbackIndex, Int32 callbackObject, Int32 sessionId)
- Int32 PortSIP.SIPCallbackEvents.onACTVTransferSuccess (Int32 callbackIndex, Int32 callbackObject, Int32 sessionId)
- Int32 PortSIP.SIPCallbackEvents.onACTVTransferFailure (Int32 callbackIndex, Int32 callbackObject, Int32 sessionId, String reason, Int32 code)

Detailed Description

Function Documentation

Int32 PortSIP.SIPCallbackEvents.onReceivedRefer (Int32 *callbackIndex*, Int32 *callbackObject*, Int32 *sessionId*, Int32 *referId*, String *to*, String *from*, String *referSipMessage*)

This event will be triggered once received a REFER message.

Parameters:

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

Int32 PortSIP.SIPCallbackEvents.onReferAccepted (Int32 *callbackIndex*, Int32 *callbackObject*, Int32 *sessionId*)

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

Parameters:

sessionId	The session ID of the call.

Int32 PortSIP.SIPCallbackEvents.onReferRejected (Int32 *callbackIndex*, Int32 *callbackObject*, Int32 *sessionId*, String *reason*, Int32 *code*)

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

Parameters:

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

Int32 PortSIP.SIPCallbackEvents.onTransferTrying (Int32 *callbackIndex*, Int32 *callbackObject*, Int32 *sessionId*)

When the refer call is processing, this event trigged.

Parameters:

sessionId	The session ID of the call.

Int32 PortSIP.SIPCallbackEvents.onTransferRinging (Int32 *callbackIndex*, Int32 *callbackObject*, Int32 *sessionId*)

When the refer call is ringing, this event trigged.

Parameters:

sessionId	The session ID of the call.

Int32 PortSIP.SIPCallbackEvents.onACTVTransferSuccess (Int32 *callbackIndex*, Int32 *callbackObject*, Int32 *sessionId*)

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

Parameters:

sessionId	The session ID of the call.

Int32 PortSIP.SIPCallbackEvents.onACTVTransferFailure (Int32 *callbackIndex*, Int32 *callbackObject*, Int32 *sessionId*, String *reason*, Int32 *code*)

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

Parameters:

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

Signaling events

Functions

- Int32 PortSIP.SIPCallbackEvents.onReceivedSignaling (Int32 callbackIndex, Int32 callbackObject, Int32 sessionId, StringBuilder signaling)
- Int32 PortSIP.SIPCallbackEvents.onSendingSignaling (Int32 callbackIndex, Int32 callbackObject, Int32 sessionId, StringBuilder signaling)

Detailed Description

Function Documentation

Int32 PortSIP.SIPCallbackEvents.onReceivedSignaling (Int32 *callbackIndex*, Int32 *callbackObject*, Int32 *sessionId*, StringBuilder *signaling*)

This event will be triggered when received a SIP message.

Parameters:

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

Int32 PortSIP.SIPCallbackEvents.onSendingSignaling (Int32 *callbackIndex*, Int32 *callbackObject*, Int32 *sessionId*, StringBuilder *signaling*)

This event will be triggered when sent a SIP message.

Parameters:

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

MWI events

Functions

- Int32 PortSIP.SIPCallbackEvents.onWaitingVoiceMessage (Int32 callbackIndex, Int32 callbackObject, String messageAccount, Int32 urgentNewMessageCount, Int32 urgentOldMessageCount, Int32 newMessageCount, Int32 oldMessageCount)
- Int32 PortSIP.SIPCallbackEvents.onWaitingFaxMessage (Int32 callbackIndex, Int32 callbackObject, String messageAccount, Int32 urgentNewMessageCount, Int32 urgentOldMessageCount, Int32 newMessageCount, Int32 oldMessageCount)

Detailed Description

Function Documentation

Int32 PortSIP.SIPCallbackEvents.onWaitingVoiceMessage (Int32 *callbackIndex*, Int32 *callbackObject*, String *messageAccount*, Int32 *urgentNewMessageCount*, Int32 *urgentOldMessageCount*, Int32 *newMessageCount*, Int32 *oldMessageCount*)

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

Parameters:

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

Int32 PortSIP.SIPCallbackEvents.onWaitingFaxMessage (Int32 *callbackIndex*, Int32 *callbackObject*, String *messageAccount*, Int32 *urgentNewMessageCount*, Int32 *urgentOldMessageCount*, Int32 *newMessageCount*, Int32 *oldMessageCount*)

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

Parameters:

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

DTMF events

Functions

- Int32 PortSIP.SIPCallbackEvents.onRecvDtmfTone (Int32 callbackIndex, Int32 callbackObject, Int32 sessionId, Int32 tone)

Detailed Description

Function Documentation

Int32 PortSIP.SIPCallbackEvents.onRecvDtmfTone (Int32 *callbackIndex*, Int32 *callbackObject*, Int32 *sessionId*, Int32 *tone*)

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

Parameters:

sessionId	The session ID of the call.
tone	Dtmf tone.

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

INFO/OPTIONS message events

Functions

- Int32 PortSIP.SIPCallbackEvents.onRecvOptions (Int32 callbackIndex, Int32 callbackObject, StringBuilder optionsMessage)
- Int32 PortSIP.SIPCallbackEvents.onRecvInfo (Int32 callbackIndex, Int32 callbackObject, StringBuilder infoMessage)

Detailed Description

Function Documentation

Int32 PortSIP.SIPCallbackEvents.onRecvOptions (Int32 *callbackIndex*, Int32 *callbackObject*, StringBuilder *optionsMessage*)

This event will be triggered when received the OPTIONS message.

Parameters:

optionsMessage	The received whole OPTIONS message in text format.

Int32 PortSIP.SIPCallbackEvents.onRecvInfo (Int32 *callbackIndex*, Int32 *callbackObject*, StringBuilder *infoMessage*)

This event will be triggered when received the INFO message.

Parameters:

infoMessage	The received whole INFO message in text format.

Presence events

Functions

- Int32 PortSIP.SIPCallbackEvents.onPresenceRecvSubscribe (Int32 callbackIndex, Int32 callbackObject, Int32 subscribeId, String fromDisplayName, String from, String subject)
- Int32 PortSIP.SIPCallbackEvents.onPresenceOnline (Int32 callbackIndex, Int32 callbackObject, String fromDisplayName, String from, String stateText)
- Int32 PortSIP.SIPCallbackEvents.onPresenceOffline (Int32 callbackIndex, Int32 callbackObject, String fromDisplayName, String from)
- Int32 PortSIP.SIPCallbackEvents.onRecvMessage (Int32 callbackIndex, Int32 callbackObject, Int32 sessionId, String mimeType, String subMimeType, byte[] messageData, Int32 messageDataLength)
- Int32 PortSIP.SIPCallbackEvents.onRecvOutOfDialogMessage (Int32 callbackIndex, Int32 callbackObject, String fromDisplayName, String from, String toDisplayName, String to, String mimeType, String subMimeType, byte[] messageData, Int32 messageDataLength)
- Int32 PortSIP.SIPCallbackEvents.onSendMessageSuccess (Int32 callbackIndex, Int32 callbackObject, Int32 sessionId, Int32 messageId)

- Int32 PortSIP.SIPCallbackEvents.onSendMessageFailure (Int32 callbackIndex, Int32 callbackObject, Int32 sessionId, Int32 messageId, String reason, Int32 code)
- Int32 PortSIP.SIPCallbackEvents.onSendOutOfDialogMessageSuccess (Int32 callbackIndex, Int32 callbackObject, Int32 messageId, String fromDisplayName, String from, String toDisplayName, String to)
- Int32 PortSIP.SIPCallbackEvents.onSendOutOfDialogMessageFailure (Int32 callbackIndex, Int32 callbackObject, Int32 messageId, String fromDisplayName, String from, String toDisplayName, String to, String reason, Int32 code)

Detailed Description

Function Documentation

Int32 PortSIP.SIPCallbackEvents.onPresenceRecvSubscribe (Int32 *callbackIndex*, Int32 *callbackObject*, Int32 *subscribeId*, String *fromDisplayName*, String *from*, String *subject*)

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

Parameters:

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

Int32 PortSIP.SIPCallbackEvents.onPresenceOnline (Int32 *callbackIndex*, Int32 *callbackObject*, String *fromDisplayName*, String *from*, String *stateText*)

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

Parameters:

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

Int32 PortSIP.SIPCallbackEvents.onPresenceOffline (Int32 *callbackIndex*, Int32 *callbackObject*, String *fromDisplayName*, String *from*)

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

Parameters:

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

Int32 PortSIP.SIPCallbackEvents.onRecvMessage (Int32 *callbackIndex*, Int32 *callbackObject*, Int32 *sessionId*, String *mimeType*, String *subMimeType*, byte[] *messageData*, Int32 *messageDataLength*)

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

Parameters:

sessionId	The session ID of the call.
mimeType	The message mime type.
subMimeType	The message sub mime type.

messageData	The received message body, it's can be text or binary data.
messageDataLength	The length of "messageData".

Int32 PortSIP.SIPCallbackEvents.onRecvOutOfDialogMessage (Int32 *callbackIndex*, Int32 *callbackObject*, String *fromDisplayName*, String *from*, String *toDisplayName*, String *to*, String *mimeType*, String *subMimeType*, byte[] *messageData*, Int32 *messageDataLength*)

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

Parameters:

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

Int32 PortSIP.SIPCallbackEvents.onSendMessageSuccess (Int32 *callbackIndex*, Int32 *callbackObject*, Int32 *sessionId*, Int32 *messageId*)

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

Parameters:

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

Int32 PortSIP.SIPCallbackEvents.onSendMessageFailure (Int32 *callbackIndex*, Int32 *callbackObject*, Int32 *sessionId*, Int32 *messageId*, String *reason*, Int32 *code*)

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

Parameters:

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

Int32 PortSIP.SIPCallbackEvents.onSendOutOfDialogMessageSuccess (Int32 *callbackIndex*, Int32 *callbackObject*, Int32 *messageId*, String *fromDisplayName*, String *from*, String *toDisplayName*, String *to*)

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

Parameters:

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

Int32 PortSIP.SIPCallbackEvents.onSendOutOfDialogMessageFailure (Int32 *callbackIndex*, Int32 *callbackObject*, Int32 *messageId*, String *fromDisplayName*, String *from*, String *toDisplayName*, String *to*, String *reason*, Int32 *code*)

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

Parameters:

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

Play audio and video file finished events

Functions

- Int32 PortSIP.SIPCallbackEvents.onPlayAudioFileFinished (Int32 callbackIndex, Int32 callbackObject, Int32 sessionId, String fileName)
- Int32 PortSIP.SIPCallbackEvents.onPlayVideoFileFinished (Int32 callbackIndex, Int32 callbackObject, Int32 sessionId)

Detailed Description

Function Documentation

Int32 PortSIP.SIPCallbackEvents.onPlayAudioFileFinished (Int32 *callbackIndex*, Int32 *callbackObject*, Int32 *sessionId*, String *fileName*)

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

Parameters:

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

Int32 PortSIP.SIPCallbackEvents.onPlayVideoFileFinished (Int32 *callbackIndex*, Int32 *callbackObject*, Int32 *sessionId*)

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

Parameters:

sessionId	The session ID of the call.

RTP callback events

Functions

- Int32 PortSIP.SIPCallbackEvents.onReceivedRtpPacket (IntPtr callbackObject, Int32 sessionId, Boolean isAudio, byte[] RTPPacket, Int32 packetSize)
- Int32 PortSIP.SIPCallbackEvents.onSendingRtpPacket (IntPtr callbackObject, Int32 sessionId, Boolean isAudio, byte[] RTPPacket, Int32 packetSize)

Detailed Description

Function Documentation

Int32 PortSIP.SIPCallbackEvents.onReceivedRtpPacket (IntPtr *callbackObject*, Int32 *sessionId*, Boolean *isAudio*, byte[] *RTPPacket*, Int32 *packetSize*)

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

Parameters:

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

Note:

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

Int32 PortSIP.SIPCallbackEvents.onSendingRtpPacket (IntPtr *callbackObject*, Int32 *sessionId*, Boolean *isAudio*, byte[] *RTPPacket*, Int32 *packetSize*)

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

Parameters:

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

Note:

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

Audio and video stream callback events

Functions

- Int32 PortSIP.SIPCallbackEvents.onAudioRawCallback (IntPtr callbackObject, Int32 sessionId, Int32 callbackType, byte[] data, Int32 dataLength, Int32 samplingFreqHz)
- Int32 PortSIP.SIPCallbackEvents.onVideoRawCallback (IntPtr callbackObject, Int32 sessionId, Int32 callbackType, Int32 width, Int32 height, byte[] data, Int32 dataLength)

Detailed Description

Function Documentation

Int32 PortSIP.SIPCallbackEvents.onAudioRawCallback (IntPtr *callbackObject*, Int32 *sessionId*, Int32 *callbackType*, byte[] *data*, Int32 *dataLength*, Int32 *samplingFreqHz*)

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

Parameters:

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

Note:

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

Int32 PortSIP.SIPCallbackEvents.onVideoRawCallback (IntPtr *callbackObject*, Int32 *sessionId*, Int32 *callbackType*, Int32 *width*, Int32 *height*, byte[] *data*, Int32 *dataLength*)

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

Parameters:

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

Note:

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

Namespace Documentation

Package PortSIP

Classes

- class PortSIP_Errors
- class PortSIPLib
- *The PortSIP VoIP SDK class.* interface SIPCallbackEvents

Enumerations

- enum AUDIOCODEC_TYPE : int { **AUDIOCODEC_NONE** = -1,
  AUDIOCODEC_TYPE.AUDIOCODEC_G729 = 18,
  AUDIOCODEC_TYPE.AUDIOCODEC_PCMA = 8,
  AUDIOCODEC_TYPE.AUDIOCODEC_PCMU = 0, AUDIOCODEC_TYPE.AUDIOCODEC_GSM = 3, AUDIOCODEC_TYPE.AUDIOCODEC_G722 = 9,
  AUDIOCODEC_TYPE.AUDIOCODEC_ILBC = 97, AUDIOCODEC_TYPE.AUDIOCODEC_AMR = 98, AUDIOCODEC_TYPE.AUDIOCODEC_AMRWB = 99,
  AUDIOCODEC_TYPE.AUDIOCODEC_SPEEX = 100,
  AUDIOCODEC_TYPE.AUDIOCODEC_SPEEXWB = 102,
  AUDIOCODEC_TYPE.AUDIOCODEC_ISACWB = 103,
  AUDIOCODEC_TYPE.AUDIOCODEC_ISACSWB = 104,
  AUDIOCODEC_TYPE.AUDIOCODEC_G7221 = 121,
  AUDIOCODEC_TYPE.AUDIOCODEC_OPUS = 105,
  AUDIOCODEC_TYPE.AUDIOCODEC_DTMF = 101 }
- *Audio codec type.* enum VIDEOCODEC_TYPE : int
  { VIDEOCODEC_TYPE.VIDEO_CODE_NONE = -1,
  VIDEOCODEC_TYPE.VIDEO_CODEC_I420 = 113,
  VIDEOCODEC_TYPE.VIDEO_CODEC_H263 = 34,
  VIDEOCODEC_TYPE.VIDEO_CODEC_H263_1998 = 115,
  VIDEOCODEC_TYPE.VIDEO_CODEC_H264 = 125,
  VIDEOCODEC_TYPE.VIDEO_CODEC_VP8 = 120 }
- *Video codec type.* enum VIDEO_RESOLUTION : int { **VIDEO_NONE** = 0,
  VIDEO_RESOLUTION.VIDEO_QCIF = 1, VIDEO_RESOLUTION.VIDEO_CIF = 2,
  VIDEO_RESOLUTION.VIDEO_VGA = 3, VIDEO_RESOLUTION.VIDEO_SVGA = 4,
  VIDEO_RESOLUTION.VIDEO_XVGA = 5, VIDEO_RESOLUTION.VIDEO_720P = 6,
  VIDEO_RESOLUTION.VIDEO_QVGA = 7 }
- *Video Resolution.* enum AUDIO_RECORDING_FILEFORMAT : int
  { AUDIO_RECORDING_FILEFORMAT.FILEFORMAT_WAVE = 1,
  AUDIO_RECORDING_FILEFORMAT.FILEFORMAT_AMR }
- *The audio record file format.* enum RECORD_MODE : int { RECORD_MODE.RECORD_NONE = 0,
  RECORD_MODE.RECORD_RECV = 1, RECORD_MODE.RECORD_SEND,
  RECORD_MODE.RECORD_BOTH }
- *The audio/Video record mode.* enum **CALLBACK_SESSION_ID** : int
  { **PORTSIP_LOCAL_MIX_ID** = -1, **PORTSIP_REMOTE_MIX_ID** = -2 }
- enum AUDIOSTREAM_CALLBACK_MODE : int { **AUDIOSTREAM_NONE** = 0,
  AUDIOSTREAM_CALLBACK_MODE.AUDIOSTREAM_LOCAL_MIX,
  AUDIOSTREAM_CALLBACK_MODE.AUDIOSTREAM_LOCAL_PER_CHANNEL,
  AUDIOSTREAM_CALLBACK_MODE.AUDIOSTREAM_REMOTE_MIX,
  AUDIOSTREAM_CALLBACK_MODE.AUDIOSTREAM_REMOTE_PER_CHANNEL }
- *The audio stream callback mode.* enum VIDEOSTREAM_CALLBACK_MODE : int
  { VIDEOSTREAM_CALLBACK_MODE.VIDEOSTREAM_NONE = 0,
  VIDEOSTREAM_CALLBACK_MODE.VIDEOSTREAM_LOCAL,

VIDEOSTREAM_CALLBACK_MODE.VIDEOSTREAM_REMOTE, VIDEOSTREAM_CALLBACK_MODE.VIDEOSTREAM_BOTH }

- *The video stream callback mode.* enum PORTSIP_LOG_LEVEL : int { **PORTSIP_LOG_NONE** = -1, **PORTSIP_LOG_ERROR** = 1, **PORTSIP_LOG_WARNING** = 2, **PORTSIP_LOG_INFO** = 3, **PORTSIP_LOG_DEBUG** = 4 }
- *Log level.* enum SRTP_POLICY : int { SRTP_POLICY.SRTP_POLICY_NONE = 0, SRTP_POLICY.SRTP_POLICY_FORCE, SRTP_POLICY.SRTP_POLICY_PREFER }
- *SRTP Policy.* enum TRANSPORT_TYPE : int { TRANSPORT_TYPE.TRANSPORT_UDP = 0, TRANSPORT_TYPE.TRANSPORT_TLS, TRANSPORT_TYPE.TRANSPORT_TCP, TRANSPORT_TYPE.TRANSPORT_PERS }
- *Transport for SIP signaling.* enum SESSION_REFRESH_MODE : int { SESSION_REFRESH_MODE.SESSION_REFERESH_UAC = 0, SESSION_REFRESH_MODE.SESSION_REFERESH_UAS }
- *The session refresh by UAC or UAS.* enum DTMF_METHOD { DTMF_METHOD.DTMF_RFC2833 = 0, DTMF_METHOD.DTMF_INFO = 1 }
- *send DTMF tone with two methods* enum EC_MODES { **EC_NONE** = 0, **EC_DEFAULT** = 1, **EC_CONFERENCE** = 2, **EC_AEC** = 3, **EC_AECM_1** = 4, **EC_AECM_2** = 5, **EC_AECM_3** = 6, **EC_AECM_4** = 7 }
- *type of Echo Control* enum AGC_MODES { **AGC_NONE** = 0, **AGC_DEFAULT**, **AGC_ADAPTIVE_ANALOG**, **AGC_ADAPTIVE_DIGITAL**, **AGC_FIXED_DIGITAL** }
- *type of Automatic Gain Control* enum NS_MODES { **NS_NONE** = 0, **NS_DEFAULT**, **NS_Conference**, **NS_LOW_SUPPRESSION**, **NS_MODERATE_SUPPRESSION**, **NS_HIGH_SUPPRESSION**, **NS_VERY_HIGH_SUPPRESSION** }

type of Noise Suppression

Detailed Description

PortSIP The PortSIP VoIP SDK namespace

Enumeration Type Documentation

enum PortSIP.AUDIOCODEC_TYPE : int

Audio codec type.

Enumerator

AUDIOCODEC_G729 G729 8KHZ 8kbit/s.

AUDIOCODEC_PCMA PCMA/G711 A-law 8KHZ 64kbit/s.

AUDIOCODEC_PCMU PCMU/G711 Î¼-law 8KHZ 64kbit/s.

AUDIOCODEC_GSM GSM 8KHZ 13kbit/s.

AUDIOCODEC_G722 G722 16KHZ 64kbit/s.

AUDIOCODEC_ILBC iLBC 8KHZ 30ms-13kbit/s 20 ms-15kbit/s

AUDIOCODEC_AMR Adaptive Multi-Rate (AMR) 8KHZ (4.75,5.15,5.90,6.70,7.40,7.95,10.20,12.20)kbit/s.

AUDIOCODEC_AMRWB Adaptive Multi-Rate Wideband (AMR-WB)16KHZ (6.60,8.85,12.65,14.25,15.85,18.25,19.85,23.05,23.85)kbit/s.

AUDIOCODEC_SPEEX SPEEX 8KHZ (2-24)kbit/s.

AUDIOCODEC_SPEEXWB SPEEX 16KHZ (4-42)kbit/s.

AUDIOCODEC_ISACWB internet Speech Audio Codec(iSAC) 16KHZ (32-54)kbit/s

AUDIOCODEC_ISACSWB internet Speech Audio Codec(iSAC) 16KHZ (32-160)kbit/s

AUDIOCODEC_G7221 G722.1 16KHZ (16,24,32)kbit/s.

AUDIOCODEC_OPUS OPUS 48KHZ 32kbit/s.

AUDIOCODEC_DTMF DTMF RFC 2833.

enum PortSIP.VIDEOCODEC_TYPE : int

Video codec type.

Enumerator

VIDEO_CODE_NONE Not use Video codec.

VIDEO_CODEC_I420 I420/YUV420 Raw Video format, just use with startRecord.

VIDEO_CODEC_H263 H263 video codec.

VIDEO_CODEC_H263_1998 H263+/H263 1998 video codec.

VIDEO_CODEC_H264 H264 video codec.

VIDEO_CODEC_VP8 VP8 video code.

enum PortSIP.VIDEO_RESOLUTION : int

Video Resolution.

Enumerator

VIDEO_QCIF 176X144 - for H.263, H.263-1998, H.264, VP8

VIDEO_CIF 352X288 - for H.263, H.263-1998, H.264, VP8

VIDEO_VGA 640X480 - for H.264, VP8

VIDEO_SVGA 800X600 - for H.264, VP8

VIDEO_XVGA 1024X768 - for H.264, VP8

VIDEO_720P 1280X720 - for H.264, VP8

VIDEO_QVGA 320X240 - for H.264, VP8

enum PortSIP.AUDIO_RECORDING_FILEFORMAT : int

The audio record file format.

Enumerator

FILEFORMAT_WAVE The record audio file is WAVE format.

FILEFORMAT_AMR The record audio file is AMR format - all voice data is compressed by AMR codec.

enum PortSIP.RECORD_MODE : int

The audio/Video record mode.

Enumerator

RECORD_NONE Not Record.

RECORD_RECV Only record the received data.

RECORD_SEND Only record send data.

RECORD_BOTH The record audio file is WAVE format.

enum PortSIP.AUDIOSTREAM_CALLBACK_MODE : int

The audio stream callback mode.

Enumerator

AUDIOSTREAM_LOCAL_MIX Callback the audio stream from microphone for all channels.

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

AUDIOSTREAM_REMOTE_MIX Callback the received audio stream that mixed including all channels.

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

enum PortSIP.VIDEOSTREAM_CALLBACK_MODE : int

The video stream callback mode.

Enumerator

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.

enum PortSIP.SRTP_POLICY : int

SRTP Policy.

Enumerator

SRTP_POLICY_NONE No use SRTP, The SDK can receive the encrypted call(SRTP) and unencrypted call both, but can't place outgoing encrypted call.

SRTP_POLICY_FORCE All calls must use SRTP, The SDK just allows receive encrypted Call and place outgoing encrypted call only.

SRTP_POLICY_PREFER Top priority to use SRTP, The SDK allows receive encrypted and decrypted call, and allows place outgoing encrypted call and unencrypted call.

enum PortSIP.TRANSPORT_TYPE : int

Transport for SIP signaling.

Enumerator

TRANSPORT_UDP UDP Transport.

TRANSPORT_TLS Tls Transport.

TRANSPORT_TCP TCP Transport.

TRANSPORT_PERS PERS is the PortSIP private transport for anti the SIP blocking, it must using with the PERS Server http://www.portsip.com/pers.html.

enum PortSIP.SESSION_REFRESH_MODE : int

The session refresh by UAC or UAS.

Enumerator

SESSION_REFERESH_UAC The session refresh by UAC.

SESSION_REFERESH_UAS The session refresh by UAS.

enum PortSIP.DTMF_METHOD

send DTMF tone with two methods

Enumerator

DTMF_RFC2833 send DTMF tone with RFC 2833, recommend.

DTMF_INFO send DTMF tone with SIP INFO.

Class Documentation

PortSIP.PortSIP_Errors Class Reference

Static Public Attributes

- static readonly int **INVALID_SESSION_ID** = -1
- static readonly int **ECoreAlreadyInitialized** = -60000
- static readonly int **ECoreNotInitialized** = -60001
- static readonly int **ECoreSDKObjectNull** = -60002
- static readonly int **ECoreArgumentNull** = -60003
- static readonly int **ECoreInitializeWinsockFailure** = -60004
- static readonly int **ECoreUserNameAuthNameEmpty** = -60005
- static readonly int **ECoreInitiazeStackFailure** = -60006
- static readonly int **ECorePortOutOfRange** = -60007
- static readonly int **ECoreAddTcpTransportFailure** = -60008
- static readonly int **ECoreAddTlsTransportFailure** = -60009
- static readonly int **ECoreAddUdpTransportFailure** = -60010
- static readonly int **ECoreMiniAudioPortOutOfRange** = -60011
- static readonly int **ECoreMaxAudioPortOutOfRange** = -60012
- static readonly int **ECoreMiniVideoPortOutOfRange** = -60013
- static readonly int **ECoreMaxVideoPortOutOfRange** = -60014
- static readonly int **ECoreMiniAudioPortNotEvenNumber** = -60015
- static readonly int **ECoreMaxAudioPortNotEvenNumber** = -60016
- static readonly int **ECoreMiniVideoPortNotEvenNumber** = -60017
- static readonly int **ECoreMaxVideoPortNotEvenNumber** = -60018
- static readonly int **ECoreAudioVideoPortOverlapped** = -60019
- static readonly int **ECoreAudioVideoPortRangeTooSmall** = -60020
- static readonly int **ECoreAlreadyRegistered** = -60021
- static readonly int **ECoreSIPServerEmpty** = -60022
- static readonly int **ECoreExpiresValueTooSmall** = -60023
- static readonly int **ECoreCallIdNotFound** = -60024
- static readonly int **ECoreNotRegistered** = -60025
- static readonly int **ECoreCalleeEmpty** = -60026
- static readonly int **ECoreInvalidUri** = -60027
- static readonly int **ECoreAudioVideoCodecEmpty** = -60028
- static readonly int **ECoreNoFreeDialogSession** = -60029
- static readonly int **ECoreCreateAudioChannelFailed** = -60030
- static readonly int **ECoreSessionTimerValueTooSmall** = -60040
- static readonly int **ECoreAudioHandleNull** = -60041
- static readonly int **ECoreVideoHandleNull** = -60042
- static readonly int **ECoreCallIsClosed** = -60043
- static readonly int **ECoreCallAlreadyHold** = -60044
- static readonly int **ECoreCallNotEstablished** = -60045
- static readonly int **ECoreCallNotHold** = -60050
- static readonly int **ECoreSipMessaegEmpty** = -60051
- static readonly int **ECoreSipHeaderNotExist** = -60052
- static readonly int **ECoreSipHeaderValueEmpty** = -60053
- static readonly int **ECoreSipHeaderBadFormed** = -60054
- static readonly int **ECoreBufferTooSmall** = -60055
- static readonly int **ECoreSipHeaderValueListEmpty** = -60056
- static readonly int **ECoreSipHeaderParserEmpty** = -60057
- static readonly int **ECoreSipHeaderValueListNull** = -60058
- static readonly int **ECoreSipHeaderNameEmpty** = -60059

- static readonly int **ECoreAudioSampleNotmultiple** = -60060
- static readonly int **ECoreAudioSampleOutOfRange** = -60061
- static readonly int **ECoreInviteSessionNotFound** = -60062
- static readonly int **ECoreStackException** = -60063
- static readonly int **ECoreMimeTypeUnknown** = -60064
- static readonly int **ECoreDataSizeTooLarge** = -60065
- static readonly int **ECoreSessionNumsOutOfRange** = -60066
- static readonly int **ECoreNotSupportCallbackMode** = -60067
- static readonly int **ECoreNotFoundSubscribeId** = -60068
- static readonly int **ECoreCodecNotSupport** = -60069
- static readonly int **ECoreCodecParameterNotSupport** = -60070
- static readonly int **ECorePayloadOutofRange** = -60071
- static readonly int **ECorePayloadHasExist** = -60072
- static readonly int **ECoreFixPayloadCantChange** = -60073
- static readonly int **ECoreCodecTypeInvalid** = -60074
- static readonly int **ECoreCodecWasExist** = -60075
- static readonly int **ECorePayloadTypeInvalid** = -60076
- static readonly int **ECoreArgumentTooLong** = -60077
- static readonly int **ECoreMiniRtpPortMustIsEvenNum** = -60078
- static readonly int **ECoreCallInHold** = -60079
- static readonly int **ECoreNotIncomingCall** = -60080
- static readonly int **ECoreCreateMediaEngineFailure** = -60081
- static readonly int **ECoreAudioCodecEmptyButAudioEnabled** = -60082
- static readonly int **ECoreVideoCodecEmptyButVideoEnabled** = -60083
- static readonly int **ECoreNetworkInterfaceUnavailable** = -60084
- static readonly int **ECoreWrongDTMFTone** = -60085
- static readonly int **ECoreWrongLicenseKey** = -60086
- static readonly int **ECoreTrialVersionLicenseKey** = -60087
- static readonly int **ECoreOutgoingAudioMuted** = -60088
- static readonly int **ECoreOutgoingVideoMuted** = -60089
- static readonly int **EAudioFileNameEmpty** = -70000
- static readonly int **EAudioChannelNotFound** = -70001
- static readonly int **EAudioStartRecordFailure** = -70002
- static readonly int **EAudioRegisterRecodingFailure** = -70003
- static readonly int **EAudioRegisterPlaybackFailure** = -70004
- static readonly int **EAudioGetStatisticsFailure** = -70005
- static readonly int **EAudioIsPlaying** = -70006
- static readonly int **EAudioPlayObjectNotExist** = -70007
- static readonly int **EAudioPlaySteamNotEnabled** = -70008
- static readonly int **EAudioRegisterCallbackFailure** = -70009
- static readonly int **EAudioCreateAudioConferenceFailure** = -70010
- static readonly int **EAudioOpenPlayFileFailure** = -70011
- static readonly int **EAudioPlayFileModeNotSupport** = -70012
- static readonly int **EAudioPlayFileFormatNotSupport** = -70013
- static readonly int **EAudioPlaySteamAlreadyEnabled** = -70014
- static readonly int **EAudioCreateRecordFileFailure** = -70015
- static readonly int **EAudioCodecNotSupport** = -70016
- static readonly int **EAudioPlayFileNotEnabled** = -70017
- static readonly int **EAudioPlayFileUnknowSeekOrigin** = -70018
- static readonly int **EAudioCantSetDeviceIdDuringCall** = -70019
- static readonly int **EVideoFileNameEmpty** = -80000
- static readonly int **EVideoGetDeviceNameFailure** = -80001
- static readonly int **EVideoGetDeviceIdFailure** = -80002
- static readonly int **EVideoStartCaptureFailure** = -80003
- static readonly int **EVideoChannelNotFound** = -80004
- static readonly int **EVideoStartSendFailure** = -80005

- static readonly int **EVideoGetStatisticsFailure** = -80006
- static readonly int **EVideoStartPlayAviFailure** = -80007
- static readonly int **EVideoSendAviFileFailure** = -80008
- static readonly int **EVideoRecordUnknowCodec** = -80009
- static readonly int **EVideoCantSetDeviceIdDuringCall** = -80010
- static readonly int **EVideoUnsupportCaptureRotate** = -80011
- static readonly int **EVideoUnsupportCaptureResolution** = -80012
- static readonly int **EDeviceGetDeviceNameFailure** = -90001

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

- PortSIP_Errors.cs

PortSIP.PortSIPLib Class Reference

The PortSIP VoIP SDK class.

Public Member Functions

- **PortSIPLib** (Int32 callbackIndex, Int32 callbackObject, SIPCallbackEvents calbackevents)
- Boolean **createCallbackHandlers** ()
- void **releaseCallbackHandlers** ()
- Int32 initialize (TRANSPORT_TYPE transportType, PORTSIP_LOG_LEVEL logLevel, String logFilePath, Int32 maxCallLines, String sipAgent, Int32 audioDeviceLayer, Int32 videoDeviceLayer)
  Initialize the SDK.
- void unInitialize ()
  Un-initialize the SDK and release resources.
- Int32 setUser (String userName, String displayName, String authName, String password, String localIp, Int32 localSipPort, String userDomain, String sipServer, Int32 sipServerPort, String stunServer, Int32 stunServerPort, String outboundServer, Int32 outboundServerPort)
  Set user account info.
- Int32 registerServer (Int32 expires, Int32 retryTimes)
  Register to SIP proxy server(login to server)
- Int32 unRegisterServer ()
  Un-register from the SIP proxy server.
- Int32 setLicenseKey (String key)
  Set the license key, must called before setUser function.
- Int32 getNICNums ()
  Get the Network Interface Card numbers.
- Int32 getLocalIpAddress (Int32 index, StringBuilder ip, Int32 ipSize)
  Get the local IP address by Network Interface Card index.
- Int32 addAudioCodec (AUDIOCODEC_TYPE codecType)
  Enable an audio codec, it will be appears in SDP.
- Int32 addVideoCodec (VIDEOCODEC_TYPE codecType)
  Enable a video codec, it will be appears in SDP.
- Boolean isAudioCodecEmpty ()
  Detect enabled audio codecs is empty or not.
- Boolean isVideoCodecEmpty ()
  Detect enabled video codecs is empty or not.
- Int32 setAudioCodecPayloadType (AUDIOCODEC_TYPE codecType, Int32 payloadType)
  Set the RTP payload type for dynamic audio codec.
- Int32 setVideoCodecPayloadType (VIDEOCODEC_TYPE codecType, Int32 payloadType)
  Set the RTP payload type for dynamic Video codec.
- void clearAudioCodec ()
  Remove all enabled audio codecs.
- void clearVideoCodec ()
  Remove all enabled video codecs.
- Int32 setAudioCodecParameter (AUDIOCODEC_TYPE codecType, String parameter)
  Set the codec parameter for audio codec.
- Int32 setVideoCodecParameter (VIDEOCODEC_TYPE codecType, String parameter)
  Set the codec parameter for video codec.

- Int32 setDisplayName (String name)
  Set user display name.
- Int32 getVersion (out Int32 majorVersion, out Int32 minorVersion)
  Get the current version number of the SDK.
- Int32 enableReliableProvisional (Boolean enable)
  Enable/disable PRACK.
- Int32 enable3GppTags (Boolean enable)
  Enable/disable the 3Gpp tags, include "ims.icsi.mmtel" and "g.3gpp.smsip".
- void enableCallbackSendingSignaling (Boolean enable)
  Enable/disable callback the sending SIP messages.
- Int32 setSrtpPolicy (SRTP_POLICY srtpPolicy)
  Set the SRTP policy.
- Int32 setRtpPortRange (Int32 minimumRtpAudioPort, Int32 maximumRtpAudioPort, Int32 minimumRtpVideoPort, Int32 maximumRtpVideoPort)
  Set the RTP ports range for audio and video streaming.
- Int32 setRtcpPortRange (Int32 minimumRtcpAudioPort, Int32 maximumRtcpAudioPort, Int32 minimumRtcpVideoPort, Int32 maximumRtcpVideoPort)
  Set the RTCP ports range for audio and video streaming.
- Int32 enableCallForward (Boolean forBusyOnly, String forwardTo)
  Enable call forward.
- Int32 disableCallForward ()
  Disable the call forward, the SDK is not forward any incoming call after this function is called.
- Int32 enableSessionTimer (Int32 timerSeconds, SESSION_REFRESH_MODE refreshMode)
  Allows to periodically refresh Session Initiation Protocol (SIP) sessions by sending repeated INVITE requests.
- Int32 disableSessionTimer ()
  Disable the session timer.
- void setDoNotDisturb (Boolean state)
  Enable the "Do not disturb" to enable/disable.
- Int32 detectMwi ()
  Use to obtain the MWI status.
- Int32 enableCheckMwi (Boolean state)
  Allows enable/disable the check MWI(Message Waiting Indication).
- Int32 setRtpKeepAlive (Boolean state, Int32 keepAlivePayloadType, Int32 deltaTransmitTimeMS)
  Enable or disable send RTP keep-alive packet during the call is established.
- Int32 setKeepAliveTime (Int32 keepAliveTime)
  Enable or disable send SIP keep-alive packet.
- Int32 setAudioSamples (Int32 ptime, Int32 maxPtime)
  Set the audio capture sample.
- Int32 addSupportedMimeType (String methodName, String mimeType, String subMimeType)
  Set the SDK receive the SIP message that include special mime type.
- Int32 getExtensionHeaderValue (String sipMessage, String headerName, StringBuilder headerValue, Int32 headerValueLength)
  Access the SIP header of SIP message.
- Int32 addExtensionHeader (String headerName, String headerValue)
  Add the extension header(custom header) into every outgoing SIP message.
- Int32 clearAddExtensionHeaders ()
  Clear the added extension headers(custom headers)

- Int32 modifyHeaderValue (String headerName, String headerValue)
  Modify the special SIP header value for every outgoing SIP message.
- Int32 clearModifyHeaders ()
  Clear the modify headers value, no longer modify every outgoing SIP message header values.
- Int32 setAudioDeviceId (Int32 recordingDeviceId, Int32 playoutDeviceId)
  Set the audio device that will use for audio call.
- Int32 setVideoDeviceId (Int32 deviceId)
  Set the video device that will use for video call.
- Int32 setVideoResolution (VIDEO_RESOLUTION resolution)
  Set the video capture resolution.
- Int32 setVideoBitrate (Int32 bitrateKbps)
  Set the video bit rate.
- Int32 setVideoFrameRate (Int32 frameRate)
  Set the video frame rate.
- Int32 sendVideo (Int32 sessionId, Boolean sendState)
  Send the video to remote side.
- Int32 setVideoOrientation (Int32 rotation)
  Changing the orientation of the video.
- void setLocalVideoWindow (IntPtr localVideoWindow)
  Set the the window that using to display the local video image.
- Int32 setRemoteVideoWindow (Int32 sessionId, IntPtr remoteVideoWindow)
  Set the window for a session that using to display the received remote video image.
- Int32 displayLocalVideo (Boolean state)
  Start/stop to display the local video image.
- Int32 setVideoNackStatus (Boolean state)
  Enable/disable the NACK feature(rfc6642) which help to improve the video quatliy.
- void muteMicrophone (Boolean mute)
  Mute the device microphone.it's unavailable for Android and iOS.
- void muteSpeaker (Boolean mute)
  Mute the device speaker, it's unavailable for Android and iOS.
- void getDynamicVolumeLevel (out Int32 speakerVolume, out Int32 microphoneVolume)
  Obtain the dynamic microphone volume level from current call.
- Int32 call (String callee, Boolean sendSdp, Boolean videoCall)
  Make a call.
- Int32 rejectCall (Int32 sessionId, int code)
  rejectCall Reject the incoming call.
- Int32 hangUp (Int32 sessionId)
  hangUp Hang up the call.
- Int32 answerCall (Int32 sessionId, Boolean videoCall)
  answerCall Answer the incoming call.
- Int32 updateCall (Int32 sessionId, bool enableAudio, bool enableVideo)
  Use the re-INVITE to update the established call.
- Int32 hold (Int32 sessionId)
  To place a call on hold.
- Int32 unHold (Int32 sessionId)
  Take off hold.

- Int32 muteSession (Int32 sessionId, Boolean muteIncomingAudio, Boolean muteOutgoingAudio, Boolean muteIncomingVideo, Boolean muteOutgoingVideo)
  Mute the specified session audio or video.
- Int32 forwardCall (Int32 sessionId, String forwardTo)
  Forward call to another one when received the incoming call.
- Int32 sendDtmf (Int32 sessionId, DTMF_METHOD dtmfMethod, int code, int dtmfDuration, bool playDtmfTone)
  Send DTMF tone.
- Int32 refer (Int32 sessionId, String referTo)
  Refer the currently call to another one.

- Int32 attendedRefer (Int32 sessionId, Int32 replaceSessionId, String referTo)
  Make an attended refer.
- Int32 acceptRefer (Int32 referId, String referSignalingMessage)
  Accept the REFER request, a new call will be make if called this function, usuall called after onReceivedRefer callback event.
- Int32 rejectRefer (Int32 referId)
  Reject the REFER request.
- Int32 enableSendPcmStreamToRemote (Int32 sessionId, Boolean state, Int32 streamSamplesPerSec)
  Enable the SDK send PCM stream data to remote side from another source to instread of microphone.
- Int32 sendPcmStreamToRemote (Int32 sessionId, byte[] data, Int32 dataLength)
  Send the audio stream in PCM format from another source to instead of audio device capture(microphone).
- Int32 enableSendVideoStreamToRemote (Int32 sessionId, Boolean state)
  Enable the SDK send video stream data to remote side from another source to instread of camera.
- Int32 sendVideoStreamToRemote (Int32 sessionId, byte[] data, Int32 dataLength, Int32 width, Int32 height)
  Send the video stream to remote.
- Int32 setRtpCallback (Int32 callbackObject, Boolean enable)
  Set the RTP callbacks to allow access the sending and received RTP packets.
- Int32 enableAudioStreamCallback (Int32 callbackObject, Int32 sessionId, Boolean enable, AUDIOSTREAM_CALLBACK_MODE callbackMode)
  Enable/disable the audio stream callback.
- Int32 enableVideoStreamCallback (Int32 callbackObject, Int32 sessionId, VIDEOSTREAM_CALLBACK_MODE callbackMode)
  Enable/disable the video stream callback.
- Int32 startRecord (Int32 sessionId, String recordFilePath, String recordFileName, Boolean appendTimestamp, AUDIO_RECORDING_FILEFORMAT audioFileFormat, RECORD_MODE audioRecordMode, VIDEOCODEC_TYPE videoFileCodecType, RECORD_MODE videoRecordMode)
  Start record the call.
- Int32 stopRecord (Int32 sessionId)
  Stop record.
- Int32 playVideoFileToRemote (Int32 sessionId, String fileName, Boolean loop, Boolean playAudio)
  Play an AVI file to remote party.
- Int32 stopPlayVideoFileToRemote (Int32 sessionId)
  Stop play video file to remote side.

- Int32 playAudioFileToRemote (Int32 sessionId, String fileName, Int32 fileSamplesPerSec, Boolean loop)
  Play an wave file to remote party.
- Int32 stopPlayAudioFileToRemote (Int32 sessionId)
  Stop play wave file to remote side.
- Int32 playAudioFileToRemoteAsBackground (Int32 sessionId, String fileName, Int32 fileSamplesPerSec)
  Play an wave file to remote party as conversation background sound.
- Int32 stopPlayAudioFileToRemoteAsBackground (Int32 sessionId)
  Stop play an wave file to remote party as conversation background sound.
- Int32 createConference (IntPtr conferenceVideoWindow, VIDEO_RESOLUTION videoResolution, Boolean displayLocalVideoInConference)
  Create a conference. It's failures if the exists conference isn't destroy yet.
- void destroyConference ()
  Destroy the exist conference.
- Int32 setConferenceVideoWindow (IntPtr videoWindow)
  Set the window for a conference that using to display the received remote video image.
- Int32 joinToConference (Int32 sessionId)
  Join a session into exist conference, if the call is in hold, it will be un-hold automatically.
- Int32 removeFromConference (Int32 sessionId)
  Remove a session from an exist conference.
- Int32 setAudioRtcpBandwidth (Int32 sessionId, Int32 BitsRR, Int32 BitsRS, Int32 KBitsAS)
  Set the audio RTCP bandwidth parameters as the RFC3556.
- Int32 setVideoRtcpBandwidth (Int32 sessionId, Int32 BitsRR, Int32 BitsRS, Int32 KBitsAS)
  Set the video RTCP bandwidth parameters as the RFC3556.
- Int32 setAudioQos (Boolean state, Int32 DSCPValue, Int32 priority)
  Set the DSCP(differentiated services code point) value of QoS(Quality of Service) for audio channel.
- Int32 setVideoQos (Boolean state, Int32 DSCPValue)
  Set the DSCP(differentiated services code point) value of QoS(Quality of Service) for video channel.
- Int32 getNetworkStatistics (Int32 sessionId, out Int32 currentBufferSize, out Int32 preferredBufferSize, out Int32 currentPacketLossRate, out Int32 currentDiscardRate, out Int32 currentExpandRate, out Int32 currentPreemptiveRate, out Int32 currentAccelerateRate)
  Get the "in-call" statistics. The statistics are reset after the query.
- Int32 getAudioRtpStatistics (Int32 sessionId, out Int32 averageJitterMs, out Int32 maxJitterMs, out Int32 discardedPackets)
  Obtain the RTP statisics of audio channel.
- Int32 getAudioRtcpStatistics (Int32 sessionId, out Int32 bytesSent, out Int32 packetsSent, out Int32 bytesReceived, out Int32 packetsReceived, out Int32 sendFractionLost, out Int32 sendCumulativeLost, out Int32 recvFractionLost, out Int32 recvCumulativeLost)
  Obtain the RTCP statisics of audio channel.
- Int32 getVideoRtpStatistics (Int32 sessionId, out Int32 bytesSent, out Int32 packetsSent, out Int32 bytesReceived, out Int32 packetsReceived)
  Obtain the RTP statisics of video.
- void enableVAD (Boolean state)
  Enable/disable Voice Activity Detection(VAD).
- void enableAEC (EC_MODES ecMode)
  Enable/disable AEC (Acoustic Echo Cancellation).
- void enableCNG (Boolean state)

Enable/disable Comfort Noise Generator(CNG).

- void enableAGC (AGC_MODES agcMode)
  Enable/disable Automatic Gain Control(AGC).
- void enableANS (NS_MODES nsMode)
  Enable/disable Audio Noise Suppression(ANS).
- Int32 sendOptions (String to, String sdp)
  Send OPTIONS message.
- Int32 sendInfo (Int32 sessionId, String mimeType, String subMimeType, String infoContents)
  Send a INFO message to remote side in dialog.
- Int32 sendMessage (Int32 sessionId, String mimeType, String subMimeType, byte[] message, Int32 messageLength)
  Send a MESSAGE message to remote side in dialog.
- Int32 sendOutOfDialogMessage (String to, String mimeType, String subMimeType, byte[] message, Int32 messageLength)
  Send a out of dialog MESSAGE message to remote side.
- Int32 presenceSubscribeContact (String contact, String subject)
  Send a SUBSCRIBE message for presence to a contact.
- Int32 presenceRejectSubscribe (Int32 subscribeId)
  Accept the presence SUBSCRIBE request which received from contact.
- Int32 presenceAcceptSubscribe (Int32 subscribeId)
  Reject a presence SUBSCRIBE request which received from contact.
- Int32 presenceOnline (Int32 subscribeId, String stateText)
  Send a NOTIFY message to contact to notify that presence status is online/changed.
- Int32 presenceOffline (Int32 subscribeId)
  Send a NOTIFY message to contact to notify that presence status is offline.
- Int32 getNumOfRecordingDevices ()
  Gets the number of audio devices available for audio recording.
- Int32 getNumOfPlayoutDevices ()
  Gets the number of audio devices available for audio playout.
- Int32 getRecordingDeviceName (Int32 deviceIndex, StringBuilder nameUTF8, Int32 nameUTF8Length)
  Gets the name of a specific recording device given by an index.
- Int32 getPlayoutDeviceName (Int32 deviceIndex, StringBuilder nameUTF8, Int32 nameUTF8Length)
  Gets the name of a specific playout device given by an index.
- Int32 setSpeakerVolume (Int32 volume)
  Set the speaker volume level,.
- Int32 getSpeakerVolume ()
  Gets the speaker volume level.
- Int32 setSystemOutputMute (Boolean enable)
  Mutes the speaker device completely in the OS.
- Boolean getSystemOutputMute ()
  Retrieves the output device mute state in the operating system.
- Int32 setMicVolume (Int32 volume)
  Sets the microphone volume level.
- Int32 getMicVolume ()
  Retrieves the current microphone volume.
- Int32 setSystemInputMute (Boolean enable)

Mute the microphone input device completely in the OS.

- Boolean getSystemInputMute ()
  Gets the mute state of the input device in the operating system.
- void audioPlayLoopbackTest (Boolean enable)
  Use to do the audio device loop back test.
- Int32 getNumOfVideoCaptureDevices ()
  Gets the number of available capture devices.
- Int32 getVideoCaptureDeviceName (Int32 deviceIndex, StringBuilder uniqueIdUTF8, Int32 uniqueIdUTF8Length, StringBuilder deviceNameUTF8, Int32 deviceNameUTF8Length)
  Gets the name of a specific video capture device given by an index.
- Int32 showVideoCaptureSettingsDialogBox (String uniqueIdUTF8, Int32 uniqueIdUTF8Length, String dialogTitle, IntPtr parentWindow, Int32 x, Int32 y)
  Display the capture device property dialog box for the specified capture device.

Detailed Description

The PortSIP VoIP SDK class.

Author:

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

Version:

11.2

See also:

http://www.PortSIP.com

PortSIP VoIP SD functions class description.

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

- PortSIPLib.cs

PortSIP.SIPCallbackEvents Interface Reference

Public Member Functions

- Int32 onRegisterSuccess (Int32 callbackIndex, Int32 callbackObject, String statusText, Int32 statusCode)
- Int32 onRegisterFailure (Int32 callbackIndex, Int32 callbackObject, String statusText, Int32 statusCode)
- Int32 onInviteIncoming (Int32 callbackIndex, Int32 callbackObject, Int32 sessionId, String callerDisplayName, String caller, String calleeDisplayName, String callee, String audioCodecNames, String videoCodecNames, Boolean existsAudio, Boolean existsVideo)
- Int32 onInviteTrying (Int32 callbackIndex, Int32 callbackObject, Int32 sessionId)
- Int32 onInviteSessionProgress (Int32 callbackIndex, Int32 callbackObject, Int32 sessionId, String audioCodecNames, String videoCodecNames, Boolean existsEarlyMedia, Boolean existsAudio, Boolean existsVideo)
- Int32 onInviteRinging (Int32 callbackIndex, Int32 callbackObject, Int32 sessionId, String statusText, Int32 statusCode)
- Int32 onInviteAnswered (Int32 callbackIndex, Int32 callbackObject, Int32 sessionId, String callerDisplayName, String caller, String calleeDisplayName, String callee, String audioCodecNames, String videoCodecNames, Boolean existsAudio, Boolean existsVideo)
- Int32 onInviteFailure (Int32 callbackIndex, Int32 callbackObject, Int32 sessionId, String reason, Int32 code)
- Int32 onInviteUpdated (Int32 callbackIndex, Int32 callbackObject, Int32 sessionId, String audioCodecNames, String videoCodecNames, Boolean existsAudio, Boolean existsVideo)
- Int32 onInviteConnected (Int32 callbackIndex, Int32 callbackObject, Int32 sessionId)
- Int32 onInviteBeginingForward (Int32 callbackIndex, Int32 callbackObject, String forwardTo)
- Int32 onInviteClosed (Int32 callbackIndex, Int32 callbackObject, Int32 sessionId)
- Int32 onRemoteHold (Int32 callbackIndex, Int32 callbackObject, Int32 sessionId)
- Int32 onRemoteUnHold (Int32 callbackIndex, Int32 callbackObject, Int32 sessionId, String audioCodecNames, String videoCodecNames, Boolean existsAudio, Boolean existsVideo)
- Int32 onReceivedRefer (Int32 callbackIndex, Int32 callbackObject, Int32 sessionId, Int32 referId, String to, String from, String referSipMessage)
- Int32 onReferAccepted (Int32 callbackIndex, Int32 callbackObject, Int32 sessionId)
- Int32 onReferRejected (Int32 callbackIndex, Int32 callbackObject, Int32 sessionId, String reason, Int32 code)
- Int32 onTransferTrying (Int32 callbackIndex, Int32 callbackObject, Int32 sessionId)
- Int32 onTransferRinging (Int32 callbackIndex, Int32 callbackObject, Int32 sessionId)
- Int32 onACTVTransferSuccess (Int32 callbackIndex, Int32 callbackObject, Int32 sessionId)
- Int32 onACTVTransferFailure (Int32 callbackIndex, Int32 callbackObject, Int32 sessionId, String reason, Int32 code)
- Int32 onReceivedSignaling (Int32 callbackIndex, Int32 callbackObject, Int32 sessionId, StringBuilder signaling)
- Int32 onSendingSignaling (Int32 callbackIndex, Int32 callbackObject, Int32 sessionId, StringBuilder signaling)
- Int32 onWaitingVoiceMessage (Int32 callbackIndex, Int32 callbackObject, String messageAccount, Int32 urgentNewMessageCount, Int32 urgentOldMessageCount, Int32 newMessageCount, Int32 oldMessageCount)
- Int32 onWaitingFaxMessage (Int32 callbackIndex, Int32 callbackObject, String messageAccount, Int32 urgentNewMessageCount, Int32 urgentOldMessageCount, Int32 newMessageCount, Int32 oldMessageCount)
- Int32 onRecvDtmfTone (Int32 callbackIndex, Int32 callbackObject, Int32 sessionId, Int32 tone)
- Int32 onRecvOptions (Int32 callbackIndex, Int32 callbackObject, StringBuilder optionsMessage)
- Int32 onRecvInfo (Int32 callbackIndex, Int32 callbackObject, StringBuilder infoMessage)
- Int32 onPresenceRecvSubscribe (Int32 callbackIndex, Int32 callbackObject, Int32 subscribeId, String fromDisplayName, String from, String subject)

- Int32 onPresenceOnline (Int32 callbackIndex, Int32 callbackObject, String fromDisplayName, String from, String stateText)
- Int32 onPresenceOffline (Int32 callbackIndex, Int32 callbackObject, String fromDisplayName, String from)
- Int32 onRecvMessage (Int32 callbackIndex, Int32 callbackObject, Int32 sessionId, String mimeType, String subMimeType, byte[] messageData, Int32 messageDataLength)
- Int32 onRecvOutOfDialogMessage (Int32 callbackIndex, Int32 callbackObject, String fromDisplayName, String from, String toDisplayName, String to, String mimeType, String subMimeType, byte[] messageData, Int32 messageDataLength)
- Int32 onSendMessageSuccess (Int32 callbackIndex, Int32 callbackObject, Int32 sessionId, Int32 messageId)
- Int32 onSendMessageFailure (Int32 callbackIndex, Int32 callbackObject, Int32 sessionId, Int32 messageId, String reason, Int32 code)
- Int32 onSendOutOfDialogMessageSuccess (Int32 callbackIndex, Int32 callbackObject, Int32 messageId, String fromDisplayName, String from, String toDisplayName, String to)
- Int32 onSendOutOfDialogMessageFailure (Int32 callbackIndex, Int32 callbackObject, Int32 messageId, String fromDisplayName, String from, String toDisplayName, String to, String reason, Int32 code)
- Int32 onPlayAudioFileFinished (Int32 callbackIndex, Int32 callbackObject, Int32 sessionId, String fileName)
- Int32 onPlayVideoFileFinished (Int32 callbackIndex, Int32 callbackObject, Int32 sessionId)
- Int32 onReceivedRtpPacket (IntPtr callbackObject, Int32 sessionId, Boolean isAudio, byte[] RTPPacket, Int32 packetSize)
- Int32 onSendingRtpPacket (IntPtr callbackObject, Int32 sessionId, Boolean isAudio, byte[] RTPPacket, Int32 packetSize)
- Int32 onAudioRawCallback (IntPtr callbackObject, Int32 sessionId, Int32 callbackType, byte[] data, Int32 dataLength, Int32 samplingFreqHz)
- Int32 onVideoRawCallback (IntPtr callbackObject, Int32 sessionId, Int32 callbackType, Int32 width, Int32 height, byte[] data, Int32 dataLength)

Detailed Description

SIPCallbackEvents PortSIP VoIP SDK Callback events

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

- SIPCallbackEvents.cs

Index