编号：CIN/CC-13

SIPPhone-SDK for IOS 开发手册

V2.0.0.5

北京新方通信技术有限公司

2023 年 2 月

目 录

第1章 介绍

1.1 范围

本文是一个标准的 ios 版本的 sip 电话 API。

1.2 参考

《rfc3621》
《rfc2327》

1.3 术语及约定

第2章 整体结构

结构：主要包括几个部分，sip 协议栈、rtp 协议栈等，对外提供属性、方法、事件。具体内容下面详细描述。

SDK 中有一个对外接口的全局静态对象 SipPhoneCtrl.getInstance()，以前属性方法和事件都是来自这个对象。

第3章 接口内容

3.1 属性

【功能】

设置/得到属性

【函数】

```
String GetXXX()或者SetXXX(string strValue)

int GetXXX()或者SetXXX(int nValue)
```

【参数 XXX】

属性	说明	备注

属性	说明	备注
Server	String：服务器 IP 地址	必填
Domain	String：服务器域名地址，通常和 Server 一样	必填
ServerPort	Int ：服务器端口号	必填
Number	String：电话号码	必填
PassWord	String：密码	必填
AuthName	String：鉴权名	必填
AuthType	integer ：鉴权类型 0：不鉴权 1：DEGIST(默认)	必填
SurpportCodecs	String ：设置支持的编解码集合， 如”0|8|102|120”	
DefaultCode	int：默认编辑码 ulaw 0:alaw 8:ilbc 102 opus:120	
AutoAnswer	int:自动接通 1：自动接通 2：不自动接通	
SipTraceFlag	int:日志：0：没有日志 1：核心日志 2：协议日志 3：所有日志	
SetSipFileLogFlag	int:日志： 0：不写文件，通过 OnDebugMessage 事件上报； TraceFlag 为 0，日志不上报 1：写文件，不上报，可以通过 UploadLogFile 上传；	
AllCodecs	String[] ：得到 SIP 电话支持的所有编解码，只支持 get AllCodecs，目前支持： "Ulaw;0", "Alaw;8", "ILbc;102", "Opus;120"	
ServiceKey	String：注册是增加消息头，用于特殊项目，如增加一个消息头格式：“Token：123213213213213”； 如果增加多个消息头中间使用“|||”分割 比如：“AAA：123213213213213||| BBB：1232132132132”	
RouteKey	String：业务主键，特殊业务使用，由平台提供，只有 set 属性	
TransMode	枚举 emTransferMode： TransferMode_Udp, TransferMode_WebSocket, TransferMode_Tcp（暂不支持）， TransferMode_Tls（暂不支持）	
XPath	String：业务路径，wss	

【示例代码】

```
SipPhoneCtrl.Instance().SetServer("192.168.2.136");
```

3.2 方法

3.2.1 init

【功能】

初始化 SipCtrl 类

【预置条件】

1）、引用 framework 成功；

【函数】

```
id Initial()
```

【参数】

【返回值】

SipCtrl 对象

3.2.2 dealloc

【功能】

释放 SipCtrl 类

【预置条件】

1）、init 成功；

【函数】

```
void dealloc()
```

【参数】

【返回值】

无

3.2.3 getInstance（静态）

【功能】

得到 SipCtrl 对象

【预置条件】

1）、引用 framework 成功；

【函数】

```
id getInstance()
```

【参数】

【返回值】

SipCtrl 对象

3.2.4 GetVersion

【功能】

得到版本号码

【预置条件】

1）、Initial 成功；

【函数】

```
NSString* GetVersion()
```

【参数】

无

【返回值】

类似 10.20160607

3.2.5 Initial

【功能】

初始化电话条对象

【预置条件】

1）、已经设置了电话的各种参数(主要是 Server，ServerPort，Domain 等参数)；

【函数】

```
int Initial:(OneIntParamCallBack)initcb
```

【参数】

Block：void (^OneIntParamCallBack)(int code)，code 表示初始化网络连接结果码，200 表示成功，　其他表示失败。

【返回值】

0:成功

其他：失败

【后处理事件】

触发事件 3.3.7 OnInitialStatusChange；

3.2.6 UnInitial

【功能】

析构电话条对象

【预置条件】

1）、Initial 成功；

【函数】

```
int UnInitial()
```

【参数】

无

【返回值】

0:成功

其它：失败

3.2.7 Register

【功能】

注册

【预置条件】

1）、Initial 成功；

2）、已经设置了电话的各种参数(主要是 Number，PassWord，AuthName，AuthType 等参数)；

【函数】

```
int Register()
```

【参数】

无

【返回值】

0:成功

其它：失败

【后处理事件】

触发事件 3.3.1 onRegistrationStateChanged；

3.2.8 UnRegister

【功能】

注销

【预置条件】

1）、Initial 成功；

2）、Register 成功；

【函数】

```
int UnRegister()
```

【参数】

无

【返回值】

0:成功

其它：失败

【后处理事件】

触发事件 3.3.1 onRegistrationStateChanged;

3.2.9 DoCall

【功能】

外呼

【预置条件】

1）、Initial 成功;

【函数】

```
int DoCall: (NSString*) strNum extraHeader: (NSString*) extra
```

【参数】

参数	说明
strNum	String：号码 特别说明：如果是没有注册可以使用 1342622@211.103.220.86 呼到某个服务器
extraHeader	String，外呼时增加了自定义的扩展协议头,便于扩展业务，多个协议头之间使用“|||”作为分隔，格式为： “AAAA:899889|||BBBB:34534534”

【返回值】

0:成功

其它：失败

【后处理事件】

触发事件 3.3.2 onCallStateChanged;

3.2.10 Answer

【功能】

来电应答

【预置条件】

1）、电话初始化成功;

2）、有电话振铃;

【函数】

```
int Answer()
```

【参数】

【返回值】

0:成功

其它：失败

【后处理事件】

触发事件 3.3.2 onCallStateChanged;

3.2.11 Disconnect

【功能】

挂断电话

【预置条件】

1）、电话初始化成功;

2）、电话接通;

【函数】

```
int Disconnect()
```

【参数】

无

【返回值】

0:成功

其它：失败

【后处理事件】

触发事件 3.3.2 onCallStateChanged;

3.2.12 Reject

【功能】

唤醒拒接

【预置条件】

1）、Initial 成功;

【函数】

```
int Reject: (NSString*) destNum
```

【参数】

参数	说明
destNum	String：拒接标示 格式：”APP 呼叫 callID”,由唤醒消息传回

【返回值】

0:成功

其它：失败

【后处理事件】

触发事件 3.3.2 onCallStateChanged；

3.2.13 SendDtmf

【功能】

二次拨号

【预置条件】

1）、Initial 成功；

2）、有呼叫接通；

【函数】

```
-(int) SendDtmf:(int)tapKey DtmType:(int)dtmfType
```

【参数】

参数	说明
tapKey	int：二次拨号键 键值： 1(1),2(2),3(3),4(4),5(5),6(6), 7(7),8(8),9(9),10(*),0(0),11(#), 12(A),13(B),14(C),15(D)
dtmfType	Int 二次播放方式 0：Info 消息，application/dtmf-relay 1：Info 消息，application/dtmf 2：带内

【返回值】

0:成功

其它：失败

3.2.14 SendMessage

【功能】

呼叫中给对方发送文本消息

【预置条件】

呼叫成功，通话中

【函数】

```
int SendMessage:(NSString*)msg
```

【参数】

参数	说明
msg	文本信息

【返回值】

0:成功

其它：失败

3.2.15 SendInfo

【功能】

呼叫中使用 Info 方法给服务端发送 SIP 文本消息

【预置条件】

呼叫成功，通话中

【函数】

```
int SendInfo:(NSString*)conType Msg:(NSString*)msg
```

【参数】

参数	说明
conType	发送消息的格式，需要与服务端商定。例如：text/plain
msg	文本信息

【返回值】

0:成功

其它：失败

3.2.16 SetSpeakMode

【功能】

通话接通后，在“免提”和“听筒”两种模式下切换

【预置条件】

1）、电话初始化成功；

2）、有电话接通；

【函数】

```
boolean SetSpeakMode()
```

【参数】

无

【返回值】

true:设置为免提模式；

false：设置为听筒模式；

注意：每次新通话默认是听筒模式

3.2.17 Mute

【功能】

设置呼叫中本端静音

【预置条件】

1）、呼叫成功，通话中

【函数】

```
boolean Mute(bool bValue)
```

【参数】

参数	说明
bValue	bool，是否本端静音。true：静音，false：不静音

【返回值】

true:成功

false：失败

注意：每次新通话默认是双方呼通状态

3.2.18 GetSpKey

【功能】

得到呼叫中关键字

【预置条件】

1）、呼叫成功，通话中

【函数】

```
String GetSpKey (String key)
```

【参数】

参数	说明
key	String，关键字，特殊业务商定

【返回值】

String，得到关键字的值；

3.2.19 SetVedioView

【功能】

设置视频本地和远程显示窗口

【预置条件】

1）、初始化成功

【函数】

```
(void) SetVedioView:(RTCEAGLVideoView*)localView VvRemote:(RTCEAGLVideoView*)re
moteView
```

【参数】

参数	说明
localView	RTCEAGLVideoView，本地视频窗口
remoteView	RTCEAGLVideoView，远程视频窗口

【返回值】

无

3.2.20 GetSipCall

【功能】

得到当前呼叫信息，包括注册信息

【预置条件】

无

【函数】

```
SipCoreCall GetSipCall()
```

【参数】

无

【返回值】

SipCoreCall 对象

说明：

属性	说明	备注
callState	类型：CallState 见 3.3.2	呼叫状态
sNum	类型：String	对方号码
sCurID	类型：String	会话 ID
nCode	类型：int	错误码
sDescription	类型：String	错误描述
regState	类型：RegistrationState 见 3.3.1	注册状态

属性	说明	备注
regCode	类型：int 401：Unauthorized 402：Payment Required 403： Forbidden 404 ： Not Found 405 ： Method Not Allowed 406 ： Not Acceptable 407 ： Proxy Authentication Required 408 ： Request Timeout	注册码
aCodec	类型：RTPCodec	通话时使用的音频编辑码
endCause	CalEndCause 上一通呼叫结束类型： CalEndCause.NoCall -1 没有呼叫 CalEndCause.NoAnswerCall 0 无应答呼叫 CalEndCause.RejectCall 1 拒接呼叫 CalEndCause.ConnectCall 2 接通呼叫 CalEndCause.OtherFailureCall 3 其他原因呼叫	只有 callState 为 CallState.CallEnd 或者 CallState.Error 时有效。
bWakeupCall	Boolean 呼入唤醒标示： true：来电唤醒 false:没有唤醒	

3.2.21 RegisterToRemotePushSever

【功能】

向唤醒服务器上注册，用于来电唤醒使用

【预置条件】

1）、app 启动时调用；

【函数】

```
void RegisterToRemotePushSever
```

【参数】

无

注意事项：AppDelegate 中

- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions

调用：

[[SipPhoneCtrl getInstance] RegisterToRemotePushSever];

3.2.22 SetAppKey

【功能】

设置 App 的课

【预置条件】

1）、framework 引入成功；

【函数】

```
void SetAppKey:(NSString*)sValue
```

【参数】

参数	说明
sValue	String：appkey，根据 pem 证书和密码分配，用于唤醒

3.2.23 SetMediaConsultationResult

【功能】

设置 Agora 呼叫加入呼叫的结果

【预置条件】

1）、初始化成功

2）、有 Agora 呼叫呼出或者呼入

3）、有 onMediaConsultation 触发

【函数】

```
void SetMediaConsultationResult:(NSString*)strKey Param:(NSString*)strParam
```

【参数】

参数	说明
strKey	String，类型 “OnAgoraVoiceJoinSuccess”： 加入 agora 会议成功； “OnAgoraVoiceError”：加入 agora 会议失败；
strParam	String，错误原因，来自 agora 的错误码

【返回值】

无

3.2.24 UploadLogFile

【功能】

删除 Sdk 内部写的日志，上传成功后删除

【预置条件】

1）、framework 引入成功；

2）、设置了本机号码,属性 FileFlag 为 1

【函数】

```
NSString* UploadLogFile:(NSString*)sSever Port:(int)nPort
```

【参数】

参数	说明
sServer	String：平台提供
nPort	int：上传

【返回】

成功：upload [cinsip_19900000003_20190624140951.txt]success；

失败：upload failure

3.2.25 SetDelegate

【功能】

用于对象的事件回调

【预置条件】

1）、电话初始化成功；

【函数】

```
Void SetDelegate(id< SIPEventDeletgate) delegate
```

【参数】

SIPEventDeletgate，详细见 3.3 事件

3.2.26 StartScreenRecord

【功能】

开启屏幕共享

【预置条件】

呼叫成功，通话中

【函数】

```
Int StartScreenRecord
```

【返回值】

0:成功

其它：失败

3.2.27 StopScreenRecord

【功能】

关闭屏幕共享

【预置条件】

呼叫成功，通话中

【函数】

```
Int StopScreenRecord
```

【返回值】

0:成功

其它：失败

3.2.28 enableFaceDetection

【功能】

开启人脸检测

【预置条件】

1）、媒体通道选择 WebRTC

2）、开启视频通话

3）、呼叫成功，通话中

【函数】

```
void enableFaceDetection:(bool) bValue
```

【参数】

参数	说明
bValue	是否开启人脸检测功能，默认关闭,true:开启 false：关闭

【后处理事件】

触发事件 3.3.8 onFacePositionChanged

3.2.29 SetVedioPlay

【功能】

呼叫时设置视频是否发送接收视频流

【预置条件】

1）、呼叫成功，视频通话中

【函数】

```
-(Boolean) SetVedioPlay:(Boolean)bSend Recv:(Boolean)bRecv;
```

【参数】

参数	说明

bSend	Boolean，是否发送视频流。true：发送，false：不发送
bRecv	Boolean，是否接收视频流。true：接收，false：不接收

【返回值】

true:设置成功

false：失败

3.2.30 ChangeCamera

【功能】

视频通话接通后，在“前面摄像头”和“后面摄像头”两种模式下切换

【预置条件】

1）、电话初始化成功；

2）、有 webrtc 视频电话接通；

【函数】

```
boolean ChangeCamera
```

【参数】

无

【返回值】

true:设置为前面摄像头模式；

false：设置为后面摄像头模式；

注意：每次新通话默认是前面摄像头模式

3.2.31 Update

【功能】

呼叫接通后，进行影视频切换

【预置条件】

呼叫成功，通话中

【函数】

```
Int Update：(boolean) bVideo
```

【参数】

参数	说明
bVideo	boolean true 表示目标是视频；false 表示目标是音频

【返回值】

0:成功

其他：失败，

-1： 不在呼叫中；

-2：目标媒体类型和秒类型一样：

SipCall 中 mediaState 为 mediaAudioVedio（视频）不能切换到视频；

SipCall 中 mediaState 为 mediaAudio（音频）不能切换到音频

【后处理事件】

触发事件 3.3.2 onCallStateChanged；

CallState.Connected 下 mediaState 回发生变化

3.3 事件（SIPEventDeletgate）

通过委托来重载回调事件：delegate 来重载事件

3.3.1 OnRegistrationStateChanged

【功能】

电话注册事件

【预置条件】

1）、Initial 成功；

2）、调用 Register 消息；

【事件】

```
void OnRegistrationStateChanged:(RegistrationState*) state

Message:(NSString*)message
```

【参数】

参数	说明
state	RegistrationState 注册消息类型： RegistrationState.RegistrationNone 0 没有注册 RegistrationState.RegistrationProgress 1 正在注册 RegistrationState.RegistrationOk 2 注册成功 RegistrationState.RegistrationCleared 3 注销成功 RegistrationState.RegistrationFailed 4 注册失败
message	NSString:注册时描述

【触发条件】

电话的注册状态发生变化时候触发；

3.3.2 OnCallStateChanged

【功能】

电话呼叫事件

【预置条件】

1）、Initial 成功；

2）、调用 Register 消息，成功；

3）、有呼叫进入或者呼出

【事件】

```
void OnCallStateChanged:(SipCoreCall*)sipcall State:(CallState*)state

Message:(NSString*)message
```

【参数】

参数	说明
sipcall	SipCoreCall 呼叫信息，参考 3.2.9 参数说明
state	CallState 呼叫状态： CallState.IncomingReceived 来电振铃 CallState.OutgoingInit 去电振铃 CallState.CallEnd 电话结束 CallState.Error 电话异常 CallState.Connected 电话条接通 CallState.StreamsRunning 语音通话
message	NSString:描述

【触发条件】

电话呼叫状态发生变化时候触发；

3.3.3 OnDebugMessage

【功能】

底层消息报告，用于调试

【预置条件】

1）、Initial 成功；

2）、设置属性 FileFlag 为 0

【事件】

```
void OnDebugMessage: (NSString*)sMsg
```

【参数】

参数	说明
message	NSString:描述

【触发条件】

当属性 TraceFlag 不为 0 同时 FileFlag 不为 1 时底层消息上报时触发

3.3.4 OnWakeUp

【功能】

来电唤醒上报事件

【预置条件】

1）、Initial 成功；

【事件】

```
void OnWakeUp: (NSString*)callNumber BackGround:(Boolean)backGround
```

【参数】

参数	说明
callNumber	NSString:来电号码
backGround	Boolean:后台运行标志 true：后台运行 flase：前台运行

【触发条件】

有来电提醒时触发；

3.3.5 OnMediaConsultation

【功能】

Agora 或者 xylink 呼叫发起协商事件

【预置条件】

1）、Initial 成功；

2）、有呼出操作，带 Agora 标示；

【事件】

```
void OnMediaConsultation:(NSString*)sKey Param:(NSString*)sParam
```

【参数】

参数	说明
sKey	NSString 消息类型， “JoinAgoraAudio”: 加入 agora 或者 xylink 会议 “LeafAgoraAudio”: 退出 agora 或者 xylink 会议 “JoinAgoraScreen”: 加入屏幕共享 “LeafAgoraScreen”: 退出屏幕共享

Param	NSString: **sKey 为：“JoinAgoraAudio”： param 为下面** Agora: agora-AgoraAppId-AgoraChannelId-AgoraUid-AgoraCertification-AgoraTokenExpires-AgoraChannelKey-AgoraDataEncrypt-AgoraToken AgoraChannelKey: 1 aes-128-xts 2 aes-128-ecb 3 aes-256-xts XYlink: xylink-appID-MeetingID **sKey 为：“LeafAgoraAudio”: param 为空** **屏幕共享：** **sKey 为：“JoinAgoraScreen”： param 同“JoinAgoraAudio”** **sKey 为：“LeafAgoraScreen”: param 为空**

【触发条件】

agora 或者 xylink 视频呼入或者呼出，以及使用屏幕共享时触发；

3.3.6 OnMessage

【功能】

接收通话中的带内消息

【预置条件】

呼叫成功，通话中

【事件】

```
void OnMessage:(int)code Message:(NSString*)msg
```

【参数】

参数	说明
code	消息类型： 100：上传文件的结果 101：对方推送过来特殊格式图片(application/xml) 102：对方推送过来特殊格式视频(application/xml) 110：对方推送的 Alert-info; 111：对方推送过来特殊的 message(text/plain)
msg	对应的具体消息类型

【触发条件】

当服务器推送 message 消息时触发

3.3.7 OnInfo

【功能】

接收通话中的带内消息

【预置条件】

呼叫成功，通话中

【事件】

```
void OnInfo:(NSString*)conType Message:(NSString*)msg
```

【参数】

参数	说明
conType	接收文本格式，例如：text/plain
msg	接收的文本消息

【触发条件】

当服务器推送 info 消息时触发

3.3.8 OnInitialStatusChange

【功能】

初始化的网络连接状态

【预置条件】

调用 Initial 方法

【事件】

```
void OnInitialStatusChange:(int)code Des:(NSString*)des
```

【参数】

参数	说明
code	状态码。200：连接成功，0：表示关闭，其他：连接错误
msg	连接信息描述

3.3.9 onFacePositionChanged

【功能】

WebRTC 人脸检测结果回调

【预置条件】

1、WebRTC 媒体通话；

2、enableFaceDetection 开启了人脸检测功能

【事件】

```
void onFacePositionChanged:(int) faces Des:(NSArray*)rects
```

【参数】

参数	说明
faces	检测的人脸个数
rects	人脸的位置数组

3.3.10 OnScreenRecord

【功能】

获取屏幕共享协商结果获取

【预置条件】

呼叫成功，通话中

【事件】

```
void OnScreenRecord:(NSString*)key ;
```

【参数】

参数	说明
key	屏幕共享的协商结果，枚举值： “begin”：屏幕共享开始； “end”：屏幕共享结束

【触发条件】

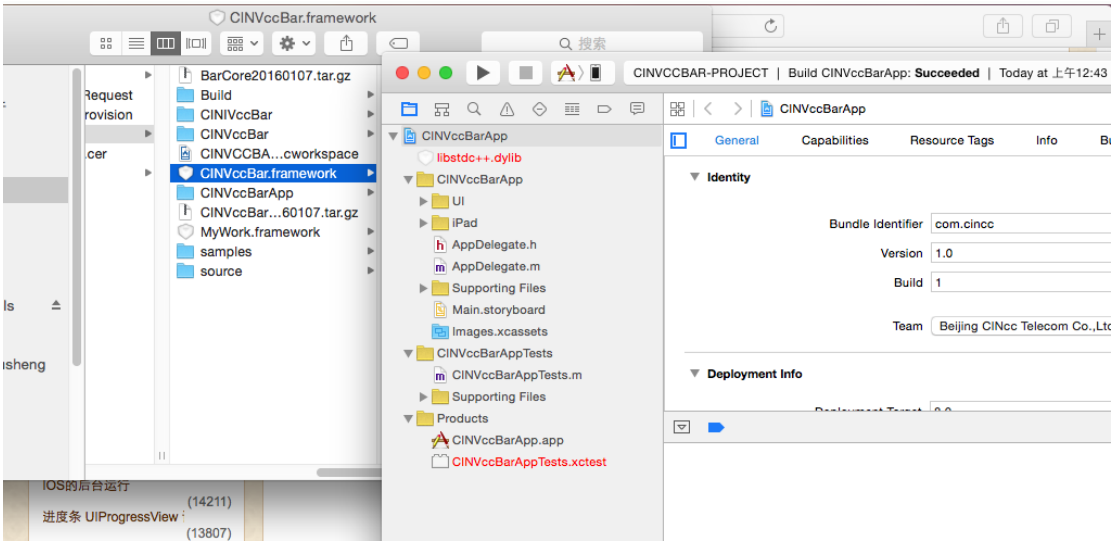
调用 StartScreenRecord 方法后接收回调

第4章 开发指南

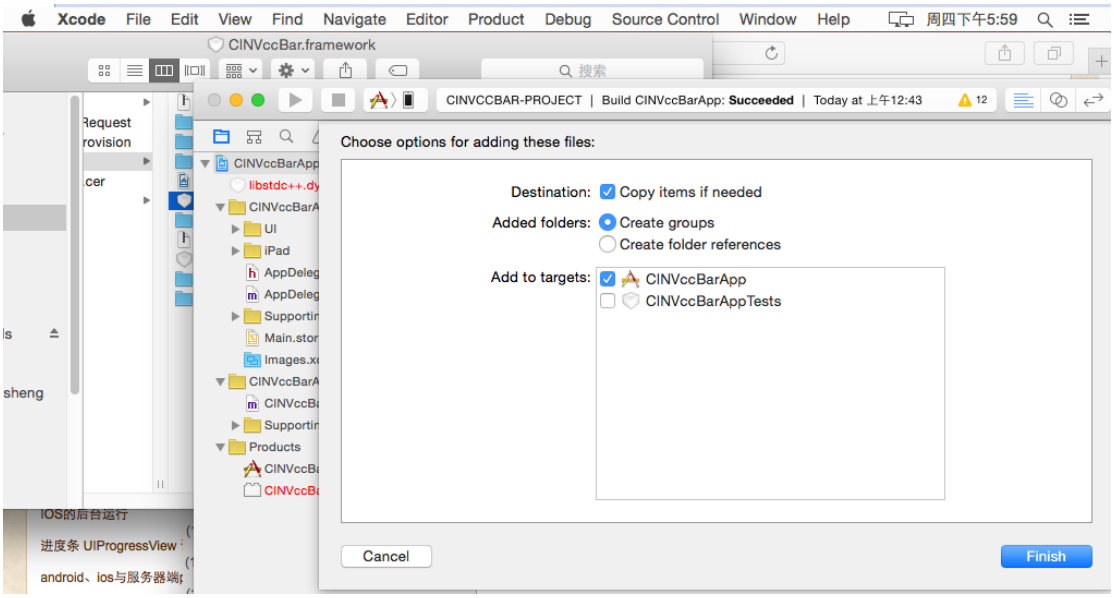
4.1 开发步骤

4.1.1 framework 加入工程

把 CINRTC. framework,CINSIP.framework,CINResource.bundle 加入工程，以 CINSIP.framework 为例，在 file in finder 中找到 CINSIP.framework

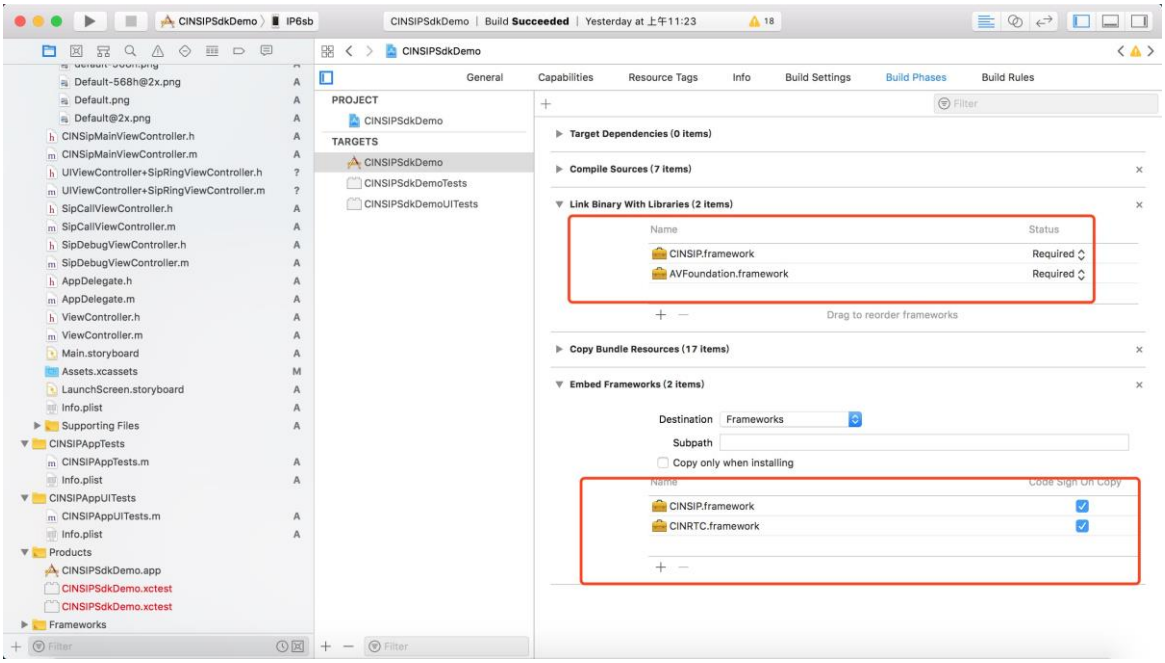


拖入工程：

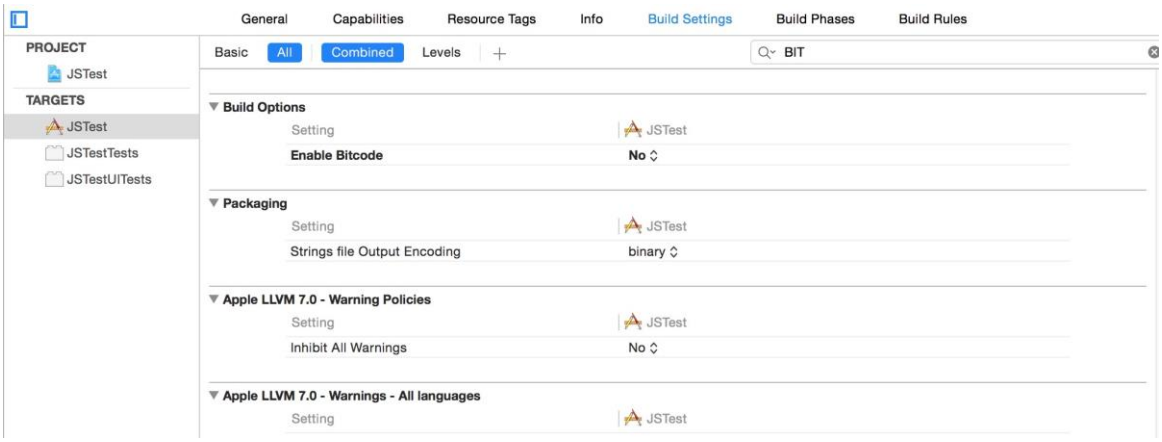


4.1.2 XCode 设置

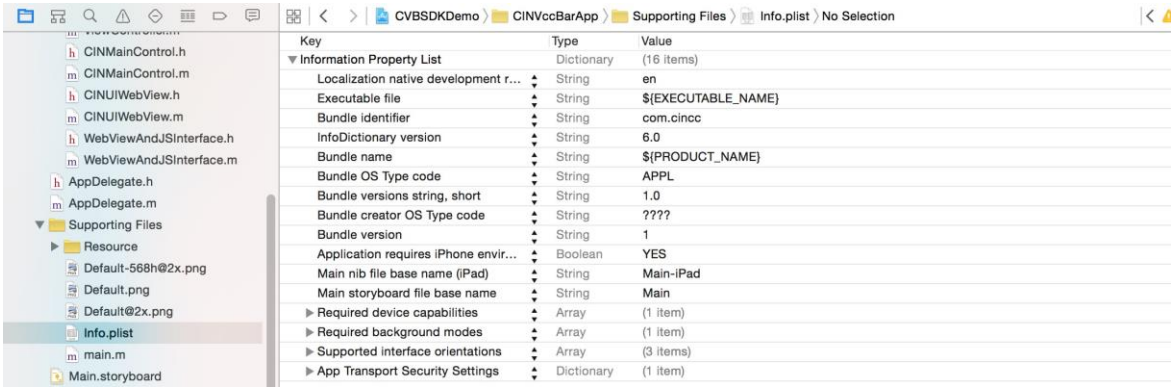
1）、Build Phases 或者 General 中的 Embedded Binaries 中加入 frameworks：



2）、使用时把 Enable Bitcode 修改成 NO：



3）、设置写日志的权限：



或者切换到 Source code 下增加：

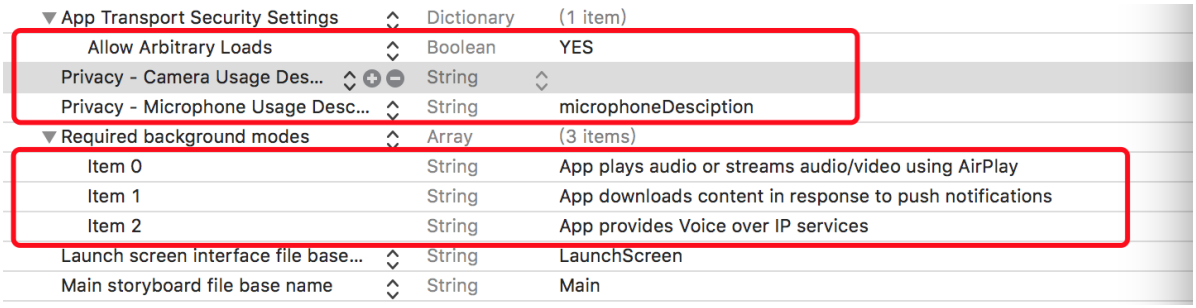
<key>NSAppTransportSecurity</key>

<dict>

```
<key>NSAllowsArbitraryLoads</key>
<true/>
</dict>
```

4.1.3 通话设置

就 SIP 电话而言，需要程序在 1）后台运行，2）需要支持 Voip，后台播放流媒体功能，3）后台来电消息通知。



1)、需要有 micphone 的使用权限：

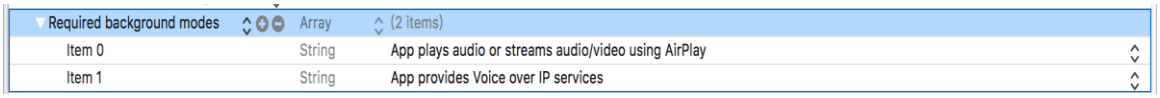
在工程的 info.plist 中增加

Privacy－Microphone Usage Description //麦克风权限

Privacy-Camera Usage Description //摄像头权限

2）、需要支持 Voip，后台播放流媒体功能：

在工程的 info.plist 中增加



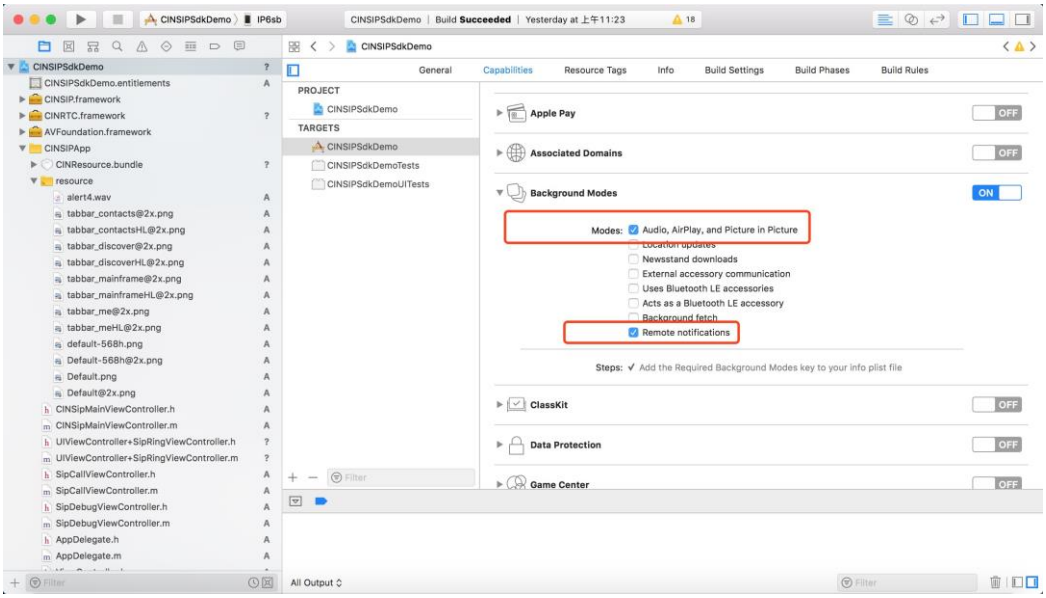
文本方式打开：

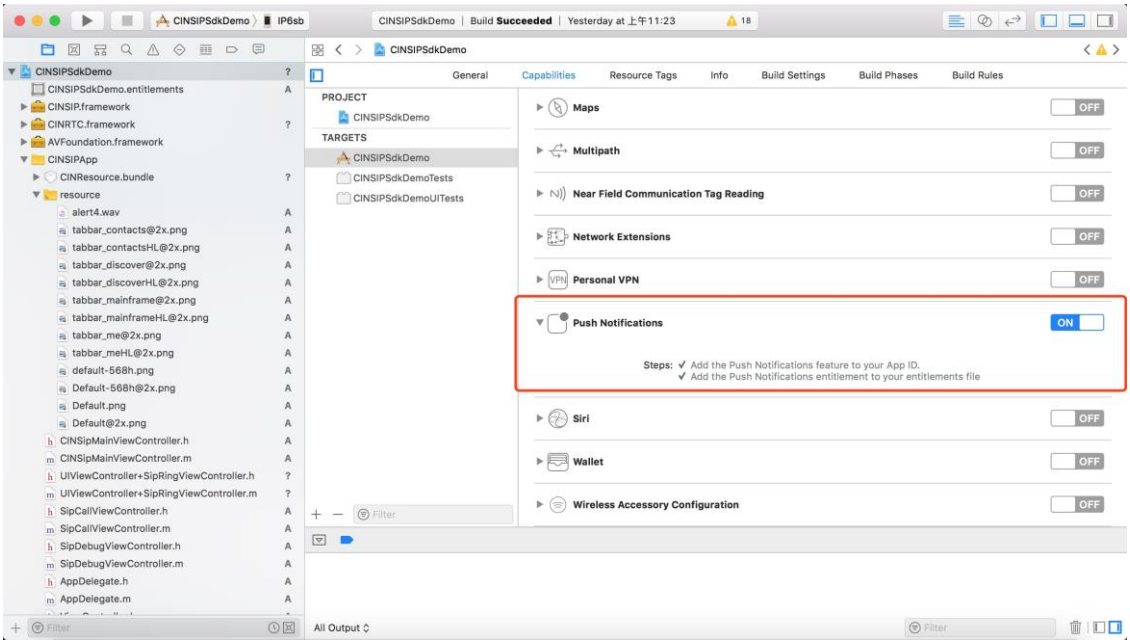
```
<key>UIBackgroundModes</key>
<array>
<string>audio</string>
<string>voip</string>
</array>
```

3）、需要远程唤醒

4.1.4 来电唤醒

1）、支持唤醒的设置：





2）、app 启动第一时间调用：

```
[[SipPhoneCtrl getInstance] RegisterToRemotePushSever];
```

3）、重载 OnWakeUp 回调事件

```
- (void)OnWakeUp:(NSString*)callNumber BackGround:(Boolean)backGround
{
  if(backGround)
  {//后台运行，弹出唤醒窗口并注册 sip 终端
    [self onCallRing:CallNumber];
  }
  else{
     //注册 sip 终端  }
}
```

4)、请提供唤醒证书和 password；

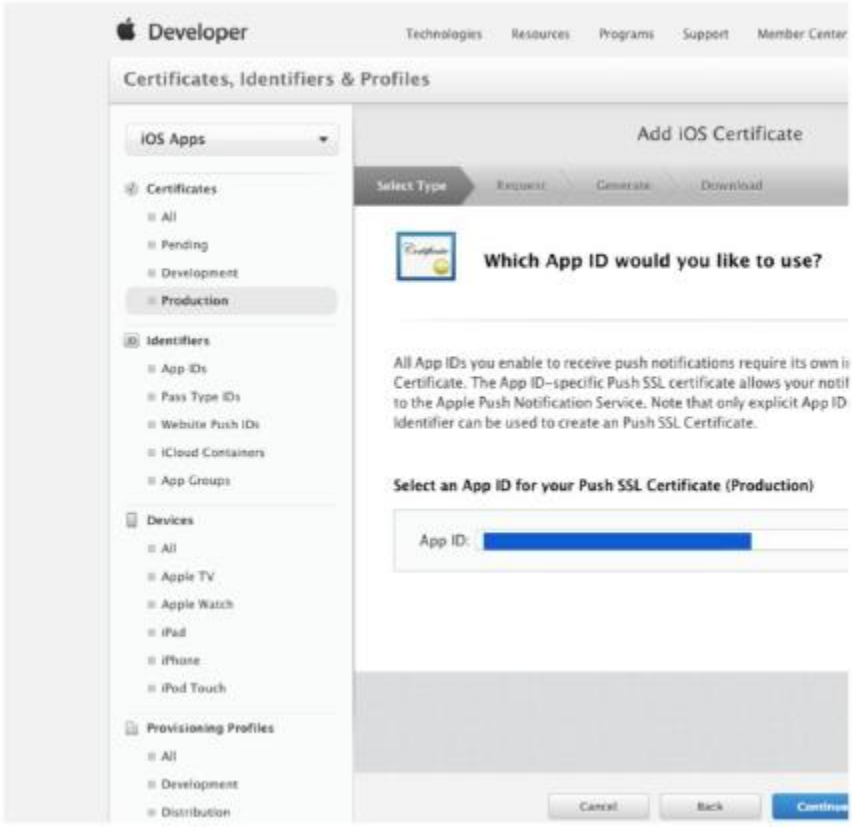
4.1.5 Voip Service 证书申请

1. 打开苹果开发者网站：https://developer.apple.com/

2. 从 Member Center 进入 Certificates, Identifiers & Profiles:

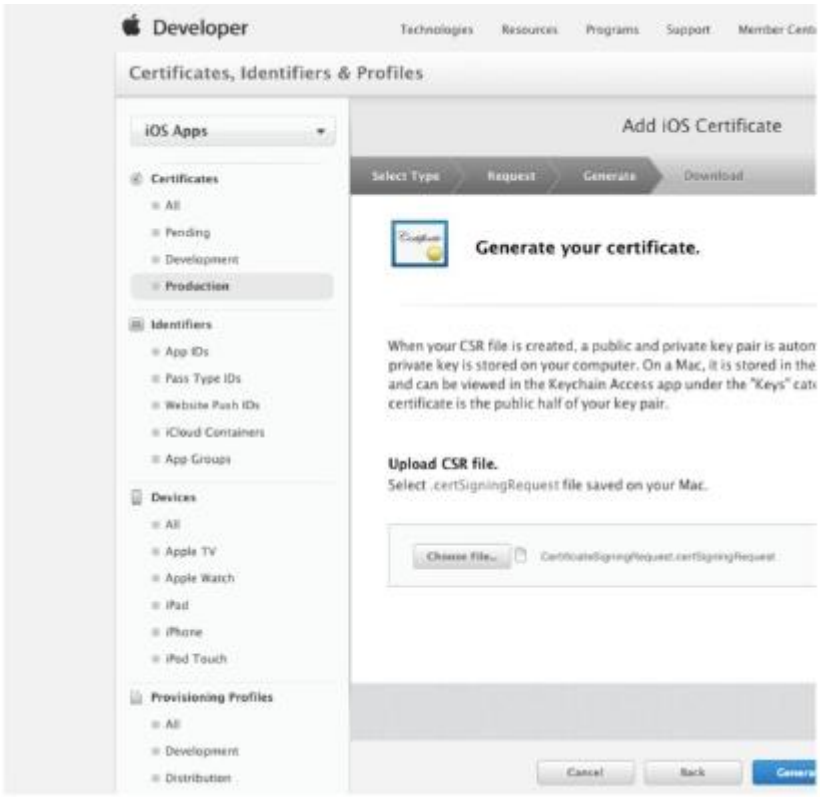
3. 选择要制作的推送证书

4. 按照提示进行操作,选择对应的 App ID：



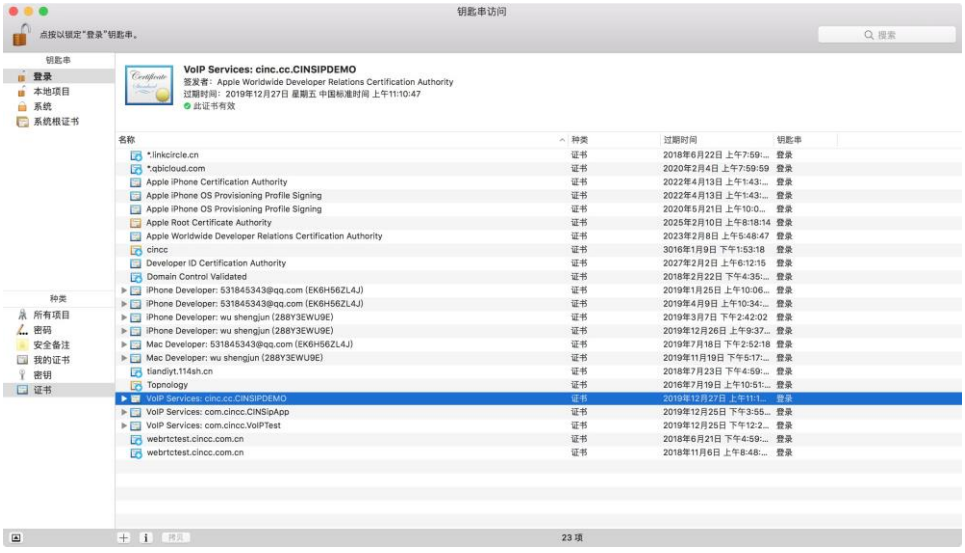
5. 根据 Certificate Assistant 的提示,创建 Certificate Request:

6. 上传 生成的 Request 文件,点击 Generate 生成证书:

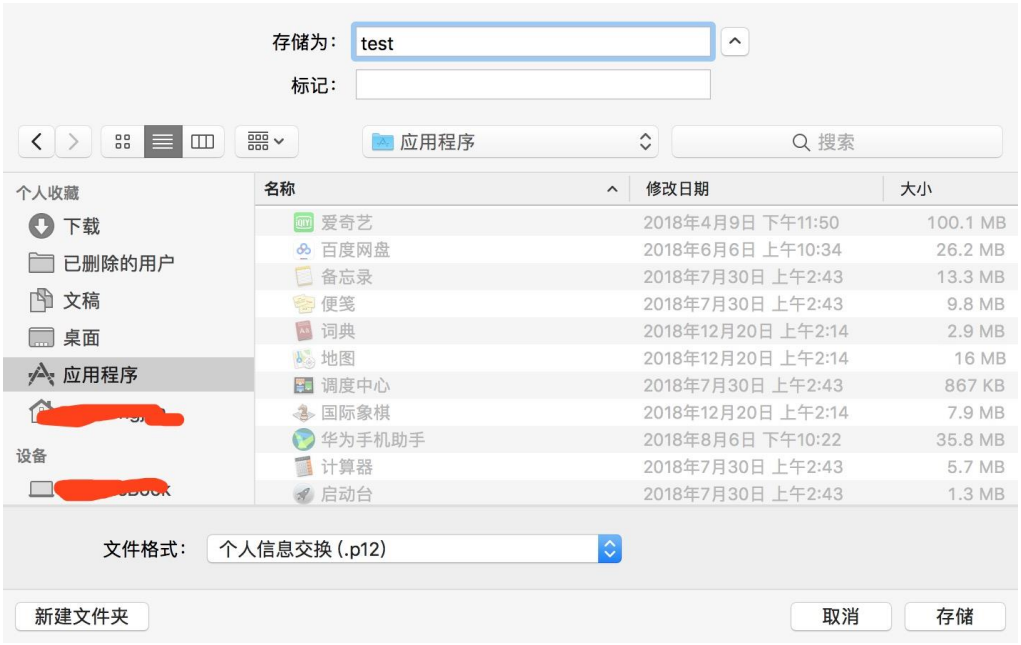


7. 此时可以在 Certificates 中查找到此证书,下载证书到本地:

8. 双击证书,将其加入到 Keychain 中,创建成功的话,证书下方会有私钥。



9. 在 Keychain 中,右击证书,导出证书为 Cer.p12;右击证书下的私钥,导出私钥为 Key.p12，并填写密码，记录下来。



10.使用下属命令生产pem格式的证书，并把证书和密码提供给厂家。

openssl x509 -in ****.cer -inform der -out VoiPCert.pem

openssl pkcs12 -nocerts -out VoIPKey.pem -in ***.p12

cat VoiPCert.pem VoIPKey.pem > @@@.pem

***为输出的证书名，@@@为提供给建设商的证书名

4.2 Demo

1）、XXX.h 中

```
#import <CINSIP/SIPPhoneCtrl.h>

@class XXXViewController;

@interface XXXViewController: NSObject<SIPEventDelegate>
{
    SIPPhoneCtrl *_oSIPCtrl;
}
```

2)、XXX.m 中、

```
//////////////////////////////
////  init
//////////////////////////////
-(id)init
{
    if(self=[super init])
```

```
        {
            _oSIPCtrl = [[SIPPhoneCtrl alloc] init];
           或者
            _oSIPCtrl = [SIPPhoneCtrl getInstance];
            _oSIPCtrl.delegate = self;
        }
        return self;
    }

    ////////////////////////////
    ////  event reload
    ////////////////////////////

    -  (void)OnRegistrationStateChanged:(RegistrationState*)state
    Message:(NSString  *)message
    {
        NSLog(@"**************OnRegistrationStateChanged \n");
    }
    - (void)OnCallStateChanged:(SipCoreCall*)sipcall State:
    (CallState*)state Message:(NSString *)message
    {
        NSLog(@"**************OnCallStateChanged \n");
    }
```

3）、初始化 SIP 电话

```
SipPhoneCtrl*  oCINSip  =  [SIPPhoneCtrl getInstance];

    [oCINSip  SetServer:_txtServerIP.text];
    [oCINSip  SetDomain:_txtServerIP.text];
    [oCINSip  SetServerPort:[_txtSIPPort.text  intValue]];
    [oCINSip  SetPassWord:_txtSIPPassWd.text];
    [oCINSip  SetNumber:_txtNumber.text];
    [oCINSip  SetAuthName:_txtNumber.text];
    [oCINSip  SetAuthType:1];
    [oCINSip  SetDefaultCodec:8];
    if([oCINSip  Initial]  ==  0){
            [oCINSip  Register];
    }
```

4）、事件重载函数中处理呼叫或者注册的消息；

修改历史

版本	2.0.0.5		
修订人		**创建时间**	2022-06-08
主要修订项目	1）、增加 Update 方法支持音视频切换		
版本	1.0.0.4		
创建人		**创建时间**	2021-07-30
主要修订项目	1）、增加屏幕共享双录方法		
版本	1.0.0.3		
创建人		**创建时间**	2021-01-29
主要修订项目	1）、增加 SetVedioView 方法； 2）、增加 UploadLogFile 方法； 3）、增加 SetMediaConsultationResult 方法； 4）、增加 OnDebugMessage 事件； 5）、增加 OnMediaConsultation 事件；		
版本	1.0.0.2		
创建人		**创建时间**	2020-05-14
主要修订项目	1）、增加 mute 方法		
版本	1.0.0.1		
创建人		**创建时间**	2019-01-8
主要修订项目	1）、修改属性代理为函数设置； 2）、增加唤醒注册函数 RegisterToRemotePushServer； 3）、增加了 SetAppKey 函数，用于唤醒； 4）、增加了 RouteKey 属性，支持业务主键功能； 3) 、增加事件 OnWakeUp 事件；		
版本	1.0.0.0		
创建人		**创建时间**	2017-12-26
主要修订项目	新建		