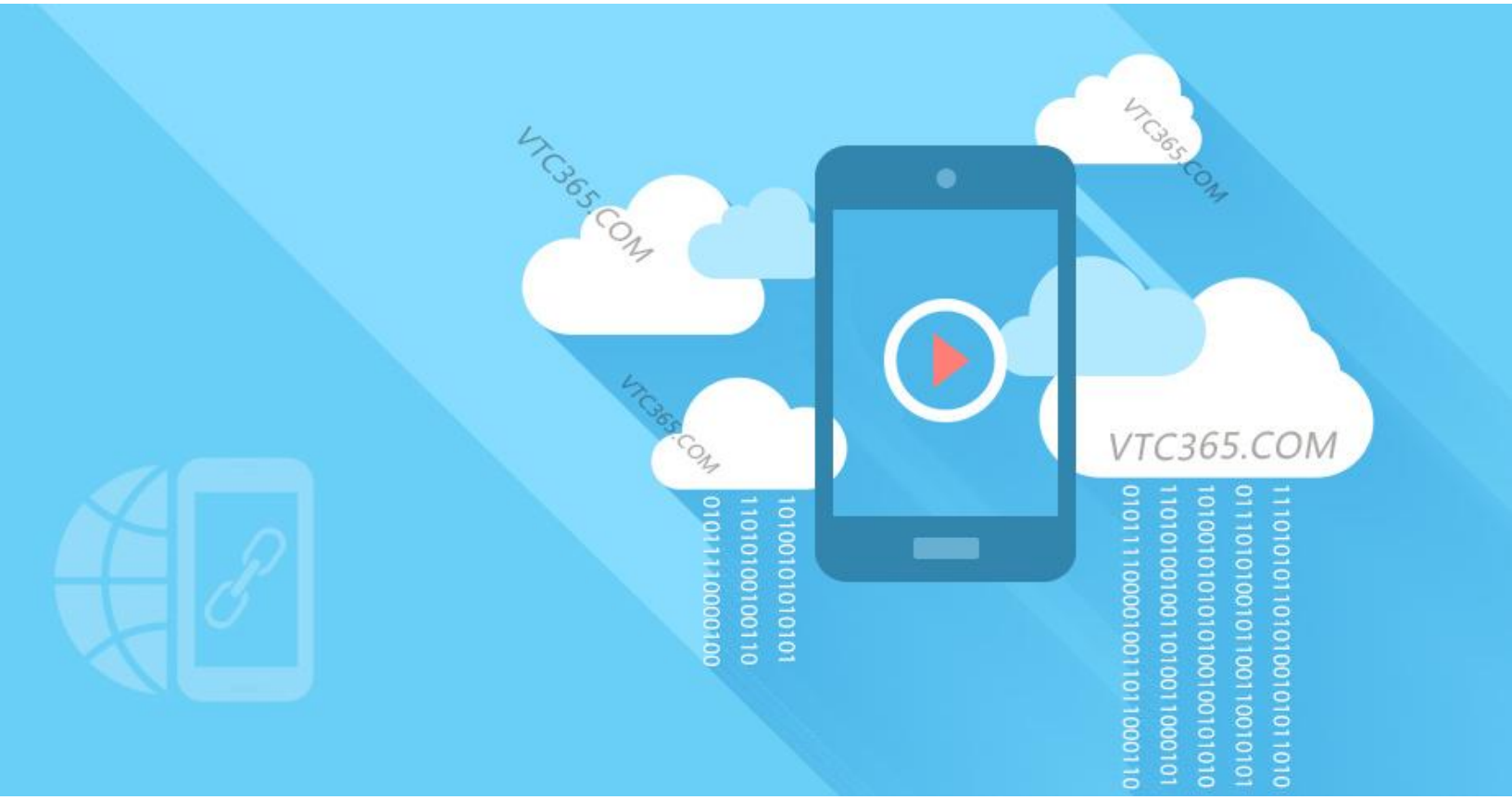


VTC 移动可视云平台及 SDK 产品和服务

ios 版本接口文档

发布时间：2015 年 8 月

软件版本：V2.5

南京微特喜网络科技有限公司

VTC 移动可视云平台及 SDK 技术
iOS 版本接口文档

版本号：V2.5

文档变更记录			
版本号	完成日期	变更描述	作者
0.1	2014-12-1	Initial version	
0.2	2015-2-30	Reworked version	
1.0	2015-3-30	First Edition version	
2.0	2015-7-22	Second version	
2.1	2015-10-30	新增获取直播视频 URL 的接口（3.2.1.2）	zhangbo@vtc365.com
2.2	2016-3-1	基于拆分的 SDK 重新整理文档	zhangbo@vtc365.com
2.3	2016-4-29	添加 APPid 和 APPtoken 的使用说明	zhangbo@vtc365.com
2.3.1	2016-6-15	新增 HLS 直播接口，支持获取网页直播播放的 URL(3.2.1.2 第 3 小节)	zhangbo@vtc365.com
2.4	2016-8-16	即时通讯 SDK 增加重连接口(3.1.7)	zhangbo@vtc365.com
2.5	2016-8-17	直播 SDK 增加设置帧率和码率接口(3.2.1.4)	zhangbo@vtc365.com

目录

1 引言

1.1 概要

VTC 移动可视云平台是一个提供 PaaS 层移动流媒体通信服务的云计算平台。平台提供 SDK 关键技术，支持视频分享、IP 电话、视频通信、视频直播等平台级移动流媒体基础功能。用户通过 SDK 技术可在其终端(APP, iPad, TV)快速实现类似微信的基础功能，并基于该基础功能开发自身产品和业务。

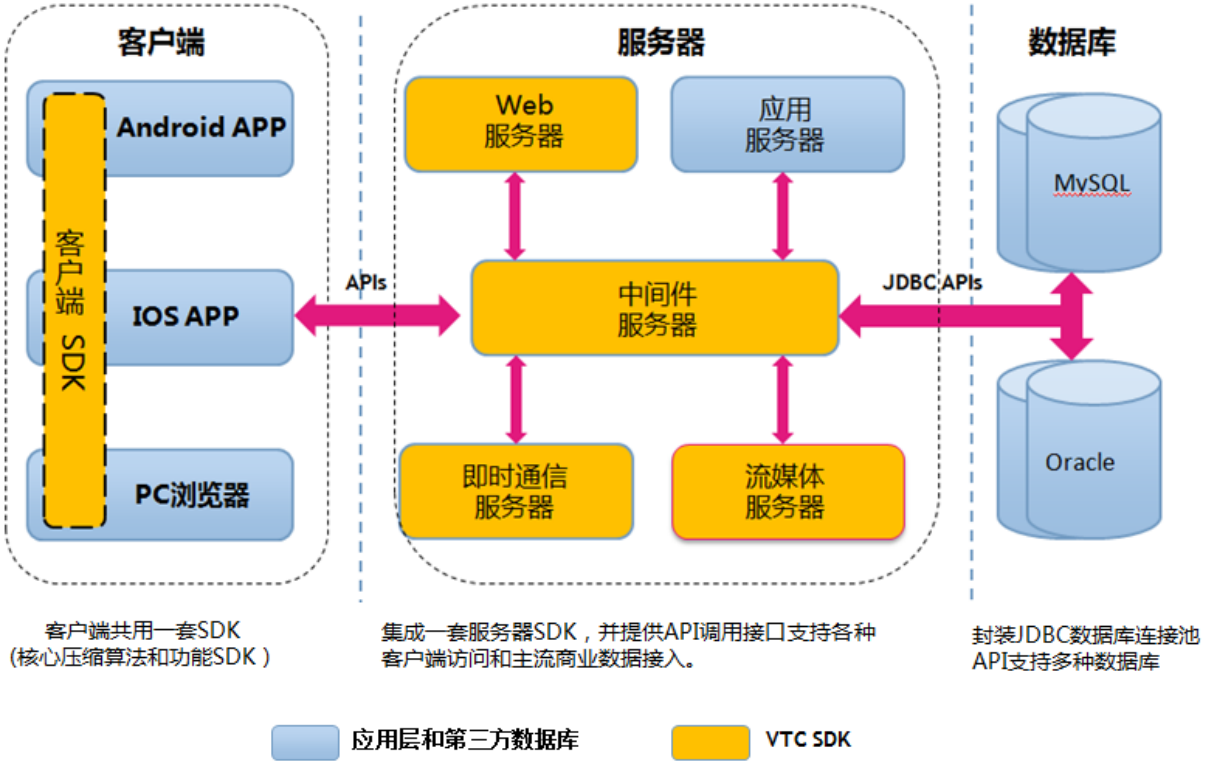
1.2 重要假设条件

- 用户具有基础的 iOS 应用层开发能力
- 用户具备基本的 iOS 代码调试、跟踪能力
- 由于 demo 里面涉及音视频，请使用真机编译运行

2 运行环境

	硬件	软件
终端	Android 手机	OS：Android 4.0 系统以上。 HW：推荐 2012 年以后的双核以上手机。 带支持 NEON 多媒体加速指令集（伪硬件模块），"ARMv7 with NEON" or "Cortex A8/A9 "及以上
	iPhone 手机	OS: iOS-7, iOS-8 HW: iPhone4S, iPhone5，iPhone5S, iPhone6, iPhone6P（推荐 iPhone5s）
	PC 机	OS: Windows7, WindowsXP, 浏览器：要求支持 IE8, Safari(on IOS), Chrome 安装 VTC 插件
服务器	x86CPU/4 核/ 8G RAM / 720G Disk 以上配置	Linux OS: CentOS 6.2 Database：MySQL, Oracle10g Web Server: Apache/Tomcat7
网络	Wi-Fi: 电信 DSL+WLAN	流畅直播：上、下行最小带宽 〉256 kbps 标清直播：上、下行最小带宽〉1.2Mbps
	4G: TD-LTE	
	3G: CDMA-EVDO/WCDMA/	

2.1 系统架构



2.2 用户接入管理

用户进入 www.vtc365.com 主页，点击页面右上角的“注册”进入注册页面。选择“开发者用户”进行注册，按照流程注册成功并登录后，即可看到如下页面，

点击“查看”获得 appId 和 appToken（用于在你的程序启动后，向微特喜服务器注册你的 APP，如下图

```
- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {
    // Override point for customization after application launch.
    //设置在微特喜网站注册获得的APPID和APPToken
    [VTCCommonClient sharedInstance].appId = 10004;
    [VTCCommonClient sharedInstance].appToken = @"elZ2k2GXfwJEayY3EGRh";
```

)

2.3 SDK 模块名称

音视频通话 sdk：Webrtc.framework

即时通讯 sdk：VTCXmpp.framework

直播 sdk：VTCLive.framework

基础 sdk：VTCCommon.framework

3 SDK 接口规范

SDK 接口分为客户端接口和服务器接口。客户端接口分别为 APP 上层应用之间的接口（外部接口）和 APP 与服务器间接口（内部接口），目前支持 Android，IOS,和 Web JS 接口，本文档为 iOS SDK 接口。服务器接口为与客户端和与客户业务服务器之间的接口，主要体现为 HTTP 接口，将在单独的服务器文档中描述。

3.1 SDK 文件介绍

音视频通话 sdk 外部接口主要在 RTCClient.h 文件中

即时通讯 sdk 外部接口主要在：XmppClient.h 文件中

直播 sdk 外部接口主要在：LiveClient.h 文件中

基础 sdk,包含注册登录等接口主要在 VTCCommonClient.h 文件中

3.1.1 注册

```
- (void)userRegisterWithUserId:(NSString *)userId
                      password:(NSString *)password
                          nick:(NSString *)nick
                        result:(VTC_CommonHandler)result;
```

3.1.2 登录

```
- (void)loginWithUserId:(NSString *)userId password:(NSString *)pwd
appId:(NSInteger)appId resultHandle:(VTC_CommonHandler)result;
```

3.1.3 XMPP 登录

Xmpp 登陆之前先要做下面的初始化工作

```
//初始化xmpp
[[XmppClient sharedInstance] setupStream];
//初始化聊天服务器url
[XmppClient sharedInstance].serverUrl = [UserDefault objectForKey:@"ServerUrl"];
```

APP 通过调用以下接口来完成 XMPP 登录：

```
- (void)xmppConnectWithParam:(NSDictionary *)dic result:(xmppLoginHandler)result
```

3.1.4 XMPP 登录冲突回调

```
- (void)apiClient:(XMPPStream *)sender didReceiveDuplicateLogin:(id)error;
```

3.1.5 获取好友列表

```
- (void)fetchRosterResult:(VTC_FriendHandler)result;
```

3.1.6 好友列表更新回调

```
-   (void)apiClient:(XMPPStream *)sender didFriendListUpdated:(NSArray *)friendList;
```

3.1.7 即时通讯重连回调

在即时通讯 SDK 和服务器的连接中断需要重新连接时，SDK 内部会抛出一个通知询问应用层是否进行重连，当用户注册并实现下面的通知后

```
[NSNotificationCenter defaultCenter] addObserver:self
selector:@selector(customMethods) name:"XMPPAUTOCONNECT" object:nil];
```

在实现的 customMethods 方法里面通过设置 SDK 里面的 autoConnectFlag 属性值，可以控制重连的行为：

```
@property (assign, nonatomic) BOOL autoConnectFlag;
```

当设置 autoConnectFlag 为 YES 时候，允许重连，否则 SDK 将中断重连。用户可以重新登出再登录来恢复。此接口可以用来实现断线登录前，先判断用户是否允许在此设备登录，如是否已经被其它设备踢出登录。

3.1.8 好友操作

3.1.8.1 添加好友

```
- (void)addFriendFromId:(NSString *)fromId
              toIdArr:(NSArray *)toIdArr
               reason:(NSString *)reason      result:(VTC_FriendHandler)result;
```

3.1.8.2 收到添加好友的请求

```
- (void)apiClient:(XMPPStream *)sender didReceiveFriendRequest:(NSDictionary *)dic
```

3.1.8.3 同意对方的好友请求

```
- (void)agreeAddFriend:(NSString *)friendId reason:(NSString *)reason
result:(VTC_FriendHandler)result;
```

3.1.8.4 拒绝对方的好友请求

```
- (void)refuseAddFriend:(NSString *)friendId reason:(NSString *)reason
result:(VTC_FriendHandler)result;
```

3.1.8.5 收到别人对自己添加好友请求的处理

```
- (void)apiClient:(XMPPStream *)sender
didReceiveFriendHandleAddRequest:(NSDictionary *)dic;
```

3.1.8.6 删除好友

```
- (void)removeUser:(NSString *)jidStr result:(VTC_FriendHandler)result;
```

3.1.8.7 收到被好友删除的通知

```
- (void)apiClient:(XMPPStream *)sender didReceiveBeDeleted:(NSDictionary *)dic;
```

3.1.9 群操作

3.1.9.1 创建群

```
//创建群
- (void)createGroup:(NSString *)groupName
      groupIntro:(NSString *)groupIntro
```

```
        groupType:(NSString *)groupType
         headIcon:(NSString *)headIcon
           result:(VTC_GroupBaseHandler)result;
```

3.1.9.2 删除群

```
//删除群
- (void)deleteGroup:(NSString *)groupId result:(VTC_GroupBaseHandler)result;
```

3.1.9.3 查询群

```
- (void)searchGroupWithUserId:(NSString *)fromJid
result:(VTC_GroupBaseHandler)result;
```

3.1.9.4 查询群成员

```
- (void)searchGroupMember:groupId
                   result:(VTC_GroupBaseHandler)result;
```

3.1.9.5 加入群成员

```
- (void)addGroupMember:(NSArray *)arr
                result:(VTC_GroupBaseHandler)result;
```

3.1.9.6 删除群成员

```
- (void)deleteGroupMember:(NSArray *)arr
                   result:(VTC_GroupBaseHandler)result;
```

3.1.9.7 申请退出群

```
- (void)applyQuitGroup:(NSString *)groupId
                result:(VTC_GroupBaseHandler)result;
```

3.1.9.8 邀请好友加入群

```
- (void)inviteFriends:(NSArray *)friendsArr
          invitorName:(NSString *)invitorNick
          toJoinGroup:(NSString *)groupId
            groupName:(NSString *)groupName
              result:(VTC_GroupBaseHandler)result;
```

3.1.9.9 同意加入群

```
- (void)agreeJoinGroup:(NSString *)groupId
                userId:(NSString *)userId
                result:(VTC_GroupBaseHandler)result;
```

3.1.9.10 更新群昵称

```
- (void)updateGroup:(NSString *)groupName
          newNick:(NSString *)newNickname
          result:(VTC_GroupBaseHandler)result;
```

3.1.9.11 更新群头像

```
- (void)updateGroup:(NSString *)groupName
          newHead:(NSString *)newHeadicon
                result:(VTC_GroupBaseHandler)result;
```

3.1.9.12 更新我的群昵称

```
- (void)updateMyGroupNick:(NSString *)nick groupId:(NSString *)groupId
result:(VTC_GroupBaseHandler)result;
```

3.1.9.13 更新我在群的开关

```
- (void)updateMyGroupSetting:(NSString *)settingStr groupId:(NSString *)groupId
result:(VTC_GroupBaseHandler)result;
```

3.1.10 发送消息

```
- (void)sendMesWithId:(NSString *)messageId
           chatType:(NSInteger)type
            msgBody:(NSString *)msgBody
                to:(NSString *)toJid
             result:(VTC_MsgBaseHandler)result;
```

3.1.10.1 接收到消息

```
- (void)apiClient:(XMPPStream *)sender didReceiveMessage:(XMPPMessage
*)message;
```

3.1.11 接收到系统通知

```
- (void)apiClient:(XMPPStream *)sender didReceiveNotice:(NSDictionary *)dic;
```

3.2 音视频分享和直播功能

3.2.1 调用 VTC 高效音视频录制或直播

应用通过调用[LiveClient sharedInstance]获取对音视频操作的单例，录制和直播的 API 封装在该单例对象中

接着通过调用下面的API获取录制和直播的界面，然后将它添加到自己的界面中：

```
/*! @brief 获得录制视频的预览界面
 */
- (AVCaptureVideoPreviewLayer *)previewLayer;
```

```
typedef void(^VTC_LiveHandler) (BOOL success, id obj, NSString *errorMsg);
```

VTC_LiveHandler 是录制或者直播开始结束操作的回调函数，其中参数 success 标记操作是否成功，参数 obj 返回操作成功后直播或者录播的视频信息，参数 errorMsg 表示操作失败后的错误信息。

3.2.1.1 录制

1. 准备录制，设置录制视频的标题，描述以及是否保留 ts 文件在本地，缺省是保留

```
- (void)prepareLocalRecordTitle:(NSString *)title
                          desc:(NSString *)desc
                   saveAvLocal:(BOOL)saveLocalAv;
```

2. 然后调用如下接口开始录制：

```
- (void)startRecord:(VTC_LiveHandler)result;
```

3. 结束录制，如果录制成功，回调函数result中的obj返回的是一个字典，字典里面存放录制视频的路径，封面名称以及视频名称（视频存放在ios沙盒目录下的Documents文件夹下面）

```
- (void)stopRecordingResult:(VTC_LiveHandler)result;
```

4. 退出录制需要调用以下接口来释放 AVCapturesession 资源：

```
- (void)endCaptureSession;
```

3.2.1.2 直播

1. 直播对象的设置：

直播对象，数组里面放置用户的 userid，如果不需要通知，此处可以不初始化该数组

```
@property (strong, nonatomic) NSMutableArray *directList;
```

2. 准备直播

```
@interface LiveClient :NSObject
/*! @brief 准备视频直播
 *
 * @param videoTitle  视频标题.
 * @param videoDescription  视频描述.
 * @param isPtv  是否定向直播.
 * @param shareToPlatform  是否分享到VTC之外的平台.

 * @param saveAVLocal  直播、录播:是否需要保留AV file ( .ts文件.
 */
- (void)prepareLiveRecordTitle:(NSString *)title
                          desc:(NSString *)desc
                         isPtv:(BOOL)isPtv
        shareToThirdPlatform:(BOOL)shareToPlatform
                saveAvLocal:(BOOL)saveLocalAv
                     result:(VTC_LiveHandler)result;
```

3. 然后调用如下接口开始直播：

```
- (void)startRecord:(VTC_LiveHandler)result;
```

在开始直播回调函数返回成功后，可以通过如下 API 获得该视频的观看地址，应用可以自己传递给接收者

```
- (NSString *)getRtspUrl; 获得 RTSP 观看地址
- (NSString *)getHttpUrl; 获得浏览器观看地址
```

4. 然后调用如下接口结束直播：

```
- (void)stopRecordingResult:(VTC_LiveHandler)result;
-
```

5. 接收方收到直播通知：

直播方开始直播，将直播对象传递给服务端，服务端可以通过 xmpp 等协议通知对方，这里我们使用 xmpp 协议作为信令传输

接收方收到直播通知（section 3.2.4.4）之后通过实现以下 delegate/protocol 来播放视频，调用者可自行选择播放器，播放存放在返回值 streamUrl 中的视频。

```
/*! @brief 接收到直播消息
 *
 * @param sender XMPPStream
 * @param dic 接收到的直播消息
 */

- (void)apiClient:(id)sender didReceiveLiveMsg:(NSDictionary *)dic;
```

3.2.1.3 直播/录制中的其他接口

1. 直播或录制中开启闪光灯：

```
- (void)swapTorch;
```

2. 直播或录制中切换摄像头：

```
-(void)swapCamera;
```

3.2.1.4 设置帧率和码率

```
- (void) setRecordParas:(int)fps bitRate:(int)bitrate;
```

参数fps表示用户可以设置采集图像的帧率，建议不要设置的太大或者太小，可以设置在10-30之间。值越大则画面越连贯，但是会增大数据量和CPU使用率。

参数bitrate表示用户可以设置视频码率(单位Kbps)，码率越大，则视频越清晰，但是数据量会越大。可以在500到1200之间设置一个适当的值。

3.2.2 调用系统默认音视频录制接口

请参考iOS开发文档，自行实现。

[注]：调用系统默认音视频录制接口生成的音视频文件size较大，在聊天session中分享和上传耗时和占用带宽较大，但清晰度稍好。

3.3 IP 电话

3.3.1 对方信息存放

```
/**
 * 通话对象
 */
@property (strong, nonatomic) NSString *chatWithUserId;
```

3.3.2 主叫

1. 在主叫方拨打 IP 电话之前，调用者需要调用以下接口完成一些初始化工作：

```
- (void)prepareForIPCall;
```

2. 初始化工作完成之后进入 IP 电话拨打界面，同时调用以下接口来分配音频资源，开始一通 IP 电话：

```
-(void)startIPCall:(BOOL)isCallerchatWithUser:(NSString *)user;
```

3. 挂断电话：

```
//IP 电话挂断
- (void)stopIPCall;
```

3.3.3 被叫

应用在收到 IP 电话通知时，可以在界面上显示 IP 电话通知，当用户点击接听按钮时建立呼叫。

1. 应用通过实现以下 delegate/protocol 来显示收到 IP 电话通知：

```
@protocol rcvIPVideoCallDelegate <NSObject>

@optional
(void)receiveIPCallFromUser:(NSString *)userId;
@end
```

2. 准备接听之前，请调用以下接口完成一些准备工作（相关变量的初始化）：

```
//IP 电话准备接听：被叫
- (void)prepareForIPCallAnswer;
```

3. 准备工作完成之后，可以进入 IP 电话界面，同时调用以下接口来开始这通 IP 电话：

```
//IP 电话开始
//isCaller:  NO --被叫
- (void)startIPCall:(BOOL)isCaller      chatWithUser:(NSString *)user;
```

4. 挂断电话：

```
//IP 电话挂断
- (void)stopIPCall;
```

3.3.4 通话中的状态

无论主叫还是被叫，IP 电话开始之后会经历以下几种状态，可以通过实现对应的各个 delegate/protocol 来完成相应界面的实现：

```
@protocol IPVideoCallDelegate <NSObject>
@optional
/**
 * IP电话开始，但是未进入talking状态。
 *
 */
- (void)IPCallDidStart;
```

```
/**
 * IP电话进入talking状态。
 *
 */
- (void)IPCallinTalk:(RTCMediaStream *)stream;

/**
 * IP电话停止。
 *
 */
- (void)IPCallDidStop:(BOOL)rlsFlag;

@end
```

3.4 视频电话

3.4.1 对方信息存放

```
/**
 * 通话对象
 */
@property (strong, nonatomic) NSString *chatWithUserId;
```

3.4.2 主叫

1. 在主叫方拨打视频电话之前，调用者需要调用以下接口完成一些初始化工作：
   (void)prepareForVideoCall:(NSString*)chatWithUserJidStr;

2. 初始化工作完成之后进入视频电话拨打界面，同时调用以下接口来分配音视频资源，开始一通视频电话：
   - (void)startVideoCall:(BOOL)isCaller chatWithUser:(NSString *)user;

3. 挂断电话：
   - (void)stopVideoCall;

3.4.3 被叫

应用在收到视频电话通知时，可以在界面上显示视频电话通知，当用户点击接听按钮时建立呼叫。

1，收到视频电话

- (void)receiveIPCallFromUser:(NSString *)userId;

2，准备接听之前，请调用以下接口完成一些准备工作（相关变量的初始化）：

- (void)prepareForVideoCall:(NSString *)chatWithUserJidStr;

3，准备工作完成之后，可以进入 IP 电话界面，同时调用以下接口来开始接通 IP 电话：

- (void)startVideoCall:(BOOL)isCaller chatWithUser:(NSString *)user;

4，挂断电话

- (void)stopVideoCall;

3.4.4 通话中的状态

无论主叫还是被叫，视频电话开始之后会经历以下几种状态，可以通过实现对应的各个 delegate/protocol 来完成相应界面的实现：

```
@protocol IPVideoCallDelegate <NSObject>
@optional

/**
 * 视频电话开始，但是未进入talking状态。
 *
 */
- (void)videoCallDidStart;


/**
 * 视频电话进入talking状态。
 *
 */
- (void)videoCallinTalk:(RTCMediaStream *)stream;


/**
 * 视频电话停止。
 *
 */
- (void)videoCallDidStop:(BOOL)rlsFlag;


@end
```

3.4.5 视频 View

1. 视频通话过程中界面需要呈现出本地视频和对方视频，调用者需要先定义好各个视频窗口大小，然后在 videoCallinTalk：方法中调用以下两个接口可以分别获得本地视频 view 和对方视频 view：

```
//视频电话:添加本地视频窗口
```

```
-(RTCEAGLVideoView *) localVideoView:(CGRect)localView;
//视频电话:添加对端视频窗口
-(RTCEAGLVideoView *) remoteVideoView:(CGRect)remoteView mediaStream:(RTCMediaStream *)stream;
```

2. 视频通话结束的时候调用以下接口移除这两个 view，释放资源：

```
//视频电话结束，移除 localview 和 remote view
-(void) removeVideoView;
```

4 数据库对接

详见服务器 SDK 接口文档

5 版本升级

5.1 兼容性

本 SDK 目前支持 iOS 6.0 以上版本。

当客户端操作系统版本升级后，客户端 SDK 有时可能需要针对少数 API 的升级优化。公有云独立节点运营客户和私有云用户将优先保证及时升级，公有云用户 SDK 将在统一时间通知发布更新版本。

当服务器版本经过 2 个大版本(1.x.y -> 2.x.y -> 3.x.y)升级后，客户端 SDK 需要同步升级以匹配服务器功能。对相差两个大版本的客户端 SDK 本系不再保证 100%支持。

5.2 互联互通

Android 与 IOS SDK 之间跨平台通信已通过服务器标准 XMPP 和 SIP 信令协议支持。

6 Glossary 词汇表

6.1 Acronyms 首字母缩写

Acronym	Expansion
VTC	Video Technologies Communications
SDK	Software Development Kit
API	Application Interface
XMPP	The Extensible Messaging and Presence Protocol
SIP	Session Initiation Protocol
HLS	HTTP Live Streaming protocol
RTP	Real Time Protocol
RTSP	Real Time Streaming Protocol
SVN	Subversion management tool