

微吼直播 SDK for iOS

微吼直播 中国领先的商务视频直播平台

目录

一、	修订记录	5
二、	简介	5
三、	主要流程图	6
四、	权限开通申请	6
五、	SDK 使用准备	7
1、	下载 SDK&DEMO	7
2、	开发环境要求	7
3、	添加库	7
4、	添加 framework	8
六、	获取当前 SDK 版本号	9
七、	AppKey 和 AppSecretKey 设置	9
八、	发起直播	9
1、	初始化直播引擎	10
2、	设置采集设备	10
3、	初始化音频设备	10
4、	开始视频采集	10
5、	开始发起直播	10
6、	停止直播	11
7、	切换摄像头	11
8、	手动对焦	11
9、	变焦	11
10、	设置闪光灯的模式	11
11、	重连流	12
12、	断开推流的连接，停止直播	12
13、	销毁直播引擎	12
14、	发直播事件响应	12
九、	观看直播	12
1、	初始化 VHMoviewPlayer 对象	12

2、	观看直播（RTMP）	13
3、	观看直播视频 （仅 HLS 可用）	13
4、	观看回放视频 （仅 HLS 可用）	13
5、	设置静音	14
6、	设置系统声音大小	14
7、	获取系统声音大小	14
8、	停止播放	14
9、	清晰度切换/切换到单音频.....	15
10、	销毁播放器	15
11、	看直播事件响应	15
十、	用户标识功能.....	16
1、	用户标识	16
2、	退出当前帐号	17
3、	获取当前用户标识状态	17
4、	获取当前用户状态	17
十一、	聊天功能	17
1、	创建聊天实例	17
2、	设置代理	18
3、	发送聊天内容	18
4、	接收上下线消息（代理方法）	18
5、	接收聊天消息（代理方法）	18
十二、	问答功能	18
1、	创建问答实例	18
2、	设置代理	18
3、	发送问题	19
4、	接收问答消息（代理方法）	19
十三、	文档演示功能	19
十四、	常量定义及错误码定义	21
十五、	DEMO 简介	23
十六、	第三方 K 值认证	24
1、	认证流程	24

2、	开启设置	24
3、	K 值使用	25
十七、	版本迁移重点说明	26
1、	2.3.0 迁移到 2.4.0.....	26

一、 修订记录

日期	版本号	描述	修订者
2016-04-21	V2.1.2	初稿	xy
2016-05-06	V2.2.0	新增文档演示	xy
2016-07-26	V2.3.0	新增清晰度切换	lkl
2016-09-27	V2.4.0	新增用户标识 新增聊天 新增问答 集成应用签名机制 观看直播支持音视频切换	xy

二、 简介

本文档为了指导开发者更快使用 iOS 系统上的 “自助式网络直播服务 SDK”，默认读者已经熟悉 XCode 的基本使用方法，以及具有一定的编程知识基础等。

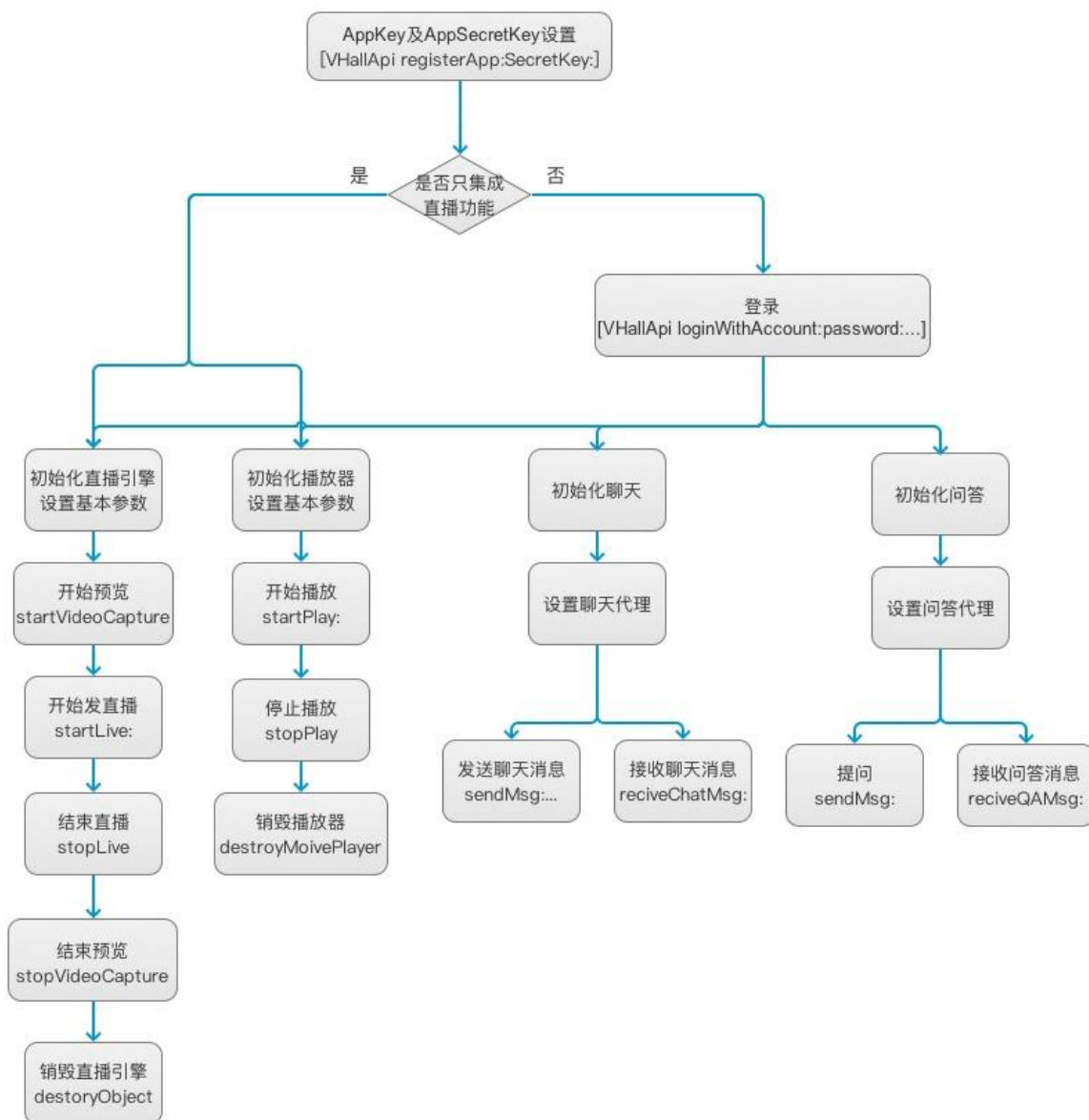
支持的产品特性如下：

分类	特性名称	描述
发起直播	支持编码类型	音频编码：AAC，视频编码：H.264
	支持推流协议	RTMP
	视频分辨率	352*288/640*480/960*540/1280*720
	屏幕朝向	横屏、竖屏
	闪光灯	开/关
	静音	开/关
	切换摄像头	前、后置摄像头
	目标码率	ios 8.0 及以上系统（支持使用硬编）， iphone 5s（全部使用软编）
	支持环境	iOS7.0 及以上
观看直播	支持播放协议	RTMP
	延时	RTMP: 2-4 秒
	支持解码	H.264
观看回放	支持协议	HLS
文档演示	支持文档演示	文档可与视频同步演示
权限	第三方 K 值认证	支持客户自己的权限验证机制来控制观看直播、观看回放的权限
用户标识（new）	支持用户标识	主要用于聊天、问答等用户互动模块
聊天（new）	支持发直播聊天	用户标识后可聊天
	支持看直播聊天	用户标识后可聊天
问答（new）	支持看直播提问	用户标识后可提问
单音频切换（new）	观看对音频转换	视频转音频，视频+文档转音频
应用签名（new）	应用签名	保证应用安全防护

三、 主要流程图

发直播、观看直播、聊天、问答，这 4 个模块集成的流程逻辑如下。

如果需要集成聊天或问答，需要提前服务器端创建用户标识，用户标识后才可正常使用。



四、 权限开通申请

1. 申请 Key

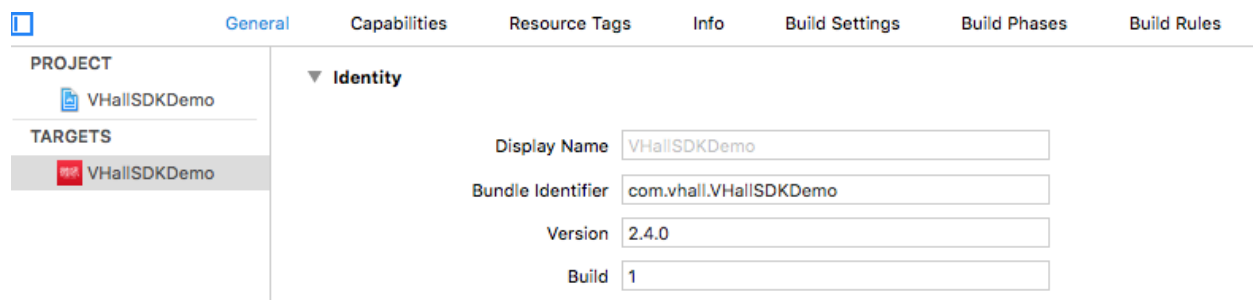
请点击 [API&SDK 权限申请](#) 或 4006826882 电话立即沟通申请，申请后客户经理会在线上与您直接联系。

审核通过后，可以获取开发应用的权限信息：App_Key、App Secret_Key，[立即查看](#)。

2. 绑定应用签名信息

使用 SDK 前集成前，务必先配置好此签名信息，否则使用时会出现“身份验证失败”提示信息。

- 进入 <http://e.vhall.com/home/vhallapi/authlist>，API/SDK 使用权限信息页面。
- 选择已开通的应用进行编辑操作。
- 点下一步进入应用绑定页面。
- 选择 IOS-SDK 切页后输入安全码 BundleID 项。（Bundle Identifier 在项目 Targets 的 General 中找到，如下图）



五、 SDK 使用准备

1、 下载 SDK&DEMO

从 github 下载：<https://github.com/vhall20/vhallsdk-ios-live/releases>

2、 开发环境要求

最低支持 iOS 版本：iOS 7.0

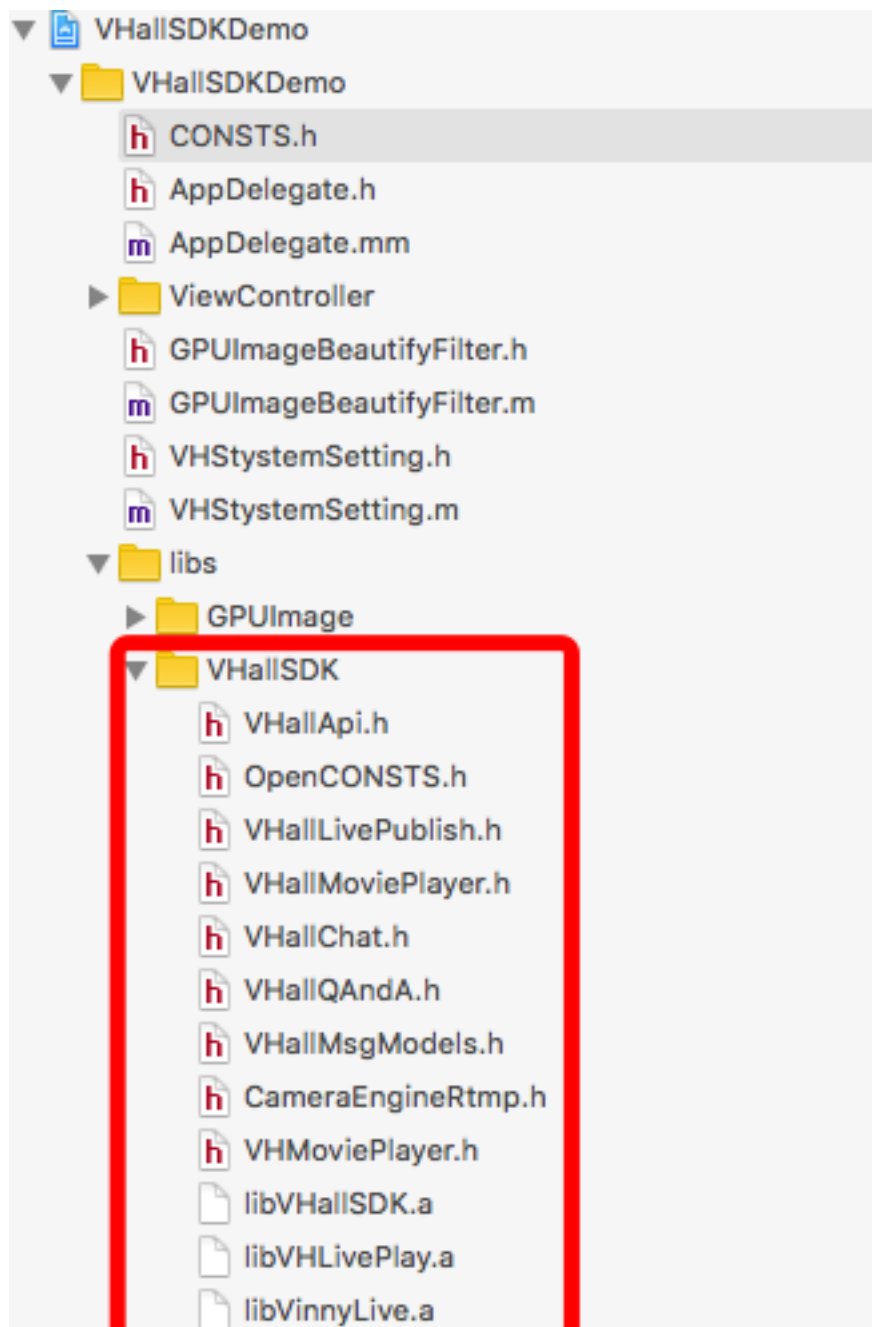
最低支持 iPhone 型号：iPhone 5

支持 CPU 架构：armv7, arm64

含有 i386 和 x86_64 模拟器版本的库文件，推流功能无法在模拟器上工作，播放功能完全支持模拟器。

















3、 添加库

将 VHallSDK 文件夹添加到工程中，添加后的效果如下图所示：



4、 添加 framework

▼ Linked Frameworks and Libraries

Name	Status
 JavaScriptCore.framework	Required ⇅
 libiconv.tbd	Required ⇅
 libbz2.1.0.tbd	Required ⇅
 libz.tbd	Required ⇅
 CoreTelephony.framework	Required ⇅
 CoreMedia.framework	Required ⇅
 QuartzCore.framework	Required ⇅
 OpenGL.framework	Required ⇅
 libVinnylive.a	Required ⇅
 OpenAL.framework	Required ⇅
 AssetsLibrary.framework	Required ⇅
 VideoToolbox.framework	Required ⇅
 AVFoundation.framework	Required ⇅
 libVHLLive.a	Required ⇅
 libVHLLivePlay.a	Required ⇅
 MediaPlayer.framework	Required ⇅

六、 获取当前 SDK 版本号

VHallApi.h

为了有效、高效定位客户问题，SDK 版本号可作为定位原因的基础信息提供给微吼。

/**

* 获取当前SDK版本号

* @return 当前SDK版本号

*/

+(NSString *) sdkVersion;

七、 AppKey 和 AppSecretKey 设置

VHallApi.h

/**

* app使用配置

* 需要在 application:(UIApplication *)application

didFinishLaunchingWithOptions:(NSDictionary *)launchOptions 中调用

* @param appKey 权限申请获得的appkey

* @param secretKey 权限申请获得的appsecretKey

*/

+ (void)registerApp:(NSString *)appKey SecretKey:(NSString *)secretKey;

八、 发起直播

VHallLivePublish.h

1、 初始化直播引擎

```
/**
 * 初始化 直播引擎
 * @param orgiation 视频拍摄方向
 *
 * @return 是否成功
 */
-(id)initWithOrgiation: (DeviceOrgiation)orgiation;
```

2、 设置采集设备

```
/**
 * 初始化 CaptureVideo
 *
 * @param captureDevicePosition
 * AVCaptureDevicePositionBack 代表后置摄像头
 * AVCaptureDevicePositionFront 代表前置摄像头
 *
 * @return 是否成功
 */
-(BOOL)initCaptureVideo: (AVCaptureDevicePosition)captureDevicePosition;
```

3、 初始化音频设备

```
-(BOOL)initAudio;
```

4、 开始视频采集

```
-(BOOL)startAudioCapture;
```

5、 开始发起直播

```
/**
 * 开始发起直播
 *
 * @param param
 * param[@"id"] = 活动Id 必传
 * param[@"access_token"] = 必传
 *
 */
-(void)startLive: (NSDictionary*)param;
```

6、 停止直播

```
/**
 * 停止直播
 * 与startLive成对出现，如果调用startLive，则需要调用stopLive以释放相应资源
 */
- (void)stopLive;
```

7、 切换摄像头

```
/**
 * 切换摄像头
 *
 * @param captureDevicePosition
 *
 * @return 是否切换成功
 */
- (BOOL)swapCameras: (AVCaptureDevicePosition) captureDevicePosition;
```

8、 手动对焦

```
/**
 *手动对焦
 */
- (void)setFocusPoint: (CGPoint)newPoint;
```

9、 变焦

```
/**
 * 变焦
 *
 * @param zoomSize 变焦的比例
 */
- (void)captureDeviceZoom: (CGFloat)zoomSize;
```

10、 设置闪光灯的模式

```
/**
 * 设置闪光灯的模式
 */
- (BOOL)setDeviceTorchMode: (AVCaptureTorchMode) captureTorchMode;
```

11、重连流

```
/**
 * 断流后重连
 */
-(BOOL)reconnect;
```

12、断开推流的连接，停止直播

```
/**
 * 断开推流的连接, 注意app进入后台时要手动调用此方法
 */
-(void)disconnect;
```

13、销毁直播引擎

```
/**
 * 销毁初始化数据
 */
-(void)destoryObject;
```

14、发直播事件响应

CameraEngineRtmpDelegate 在文件 OpenCONSTS.h 中

1、添加代理事件

```
/**
 * 采集到第一帧的回调
 * @param image 第一帧的图片
 */
-(void)firstCaptureImage:(UIImage*) image;
/**
 * 发起直播时的状态
 * @param liveStatus 直播状态
 */
```

九、观看直播

VHallMoviePlayer.h

1、初始化 VHMoviewPlayer 对象

```
/**
 * 初始化VHMoviewPlayer对象
```

```

*
* @param delegate
*
* @return 返回VHMoviePlayer的一个实例
*/
-(instancetype)initWithDelegate:(id <VHallMoviePlayerDelegate>)delegate;

```

2、 观看直播（RTMP）

```

/**
 * 观看直播
 *
 * @param param
 * param[@"id"] = 活动Id 必传
 * param[@"name"] = 未用户标识状态必传
 * param[@"email"] = 未用户标识状态必传且保证唯一性
 * param[@"pass"] = （活动如果有K值需要传）
 *
 */
-(BOOL)startPlay:(NSDictionary*)param;

```

3、 观看直播视频 （仅 HLS 可用）

```

/**
 * 观看直播视频 （仅HLS可用）
 *
 * @param param
 * param[@"id"] = 活动Id 必传
 * param[@"name"] = 未用户标识状态必传
 * param[@"email"] = 未用户标识状态必传且保证唯一性
 * param[@"pass"] = （活动如果有K值需要传）
 *
 * @param moviePlayerController MPMoviePlayerController 对象
 */
-(void)startPlay:(NSDictionary*)param moviePlayer:(MPMoviePlayerController*)moviePlayerController;

```

4、 观看回放视频 （仅 HLS 可用）

```

/**
 * 观看回放视频 （仅HLS可用）
 *

```

```

* @param param
* param[@"id"] = 活动Id 必传
* param[@"name"] = 如未用户标识, SDK要传
* param[@"email"] = 如未用户标识, SDK要传
* param[@"pass"] = 活动如果有K值或密码需要传
*
* @param moviePlayerController MPMoviePlayerController 对象
*/-(void)startPlayback:(NSDictionary*)param
moviePlayer:(MPMoviePlayerController *)moviePlayerController;

```

5、 设置静音

```

/**
* 设置静音
*
* @param mute 是否静音
*/
-(void)setMute:(BOOL)mute;

```

6、 设置系统声音大小

```

/**
* 设置系统声音大小
*
* @param size float [0.0~1.0]
*/
+(void)setSysVolumeSize:(float)size;

```

7、 获取系统声音大小

```

/**
* 获取系统声音大小
*/
+(float)getSysVolumeSize;

```

8、 停止播放

```

/**
* 停止播放
*/
-(void)stopPlay;

```

9、 清晰度切换/切换到单音频

```
/**
 * 设置直播视频清晰度 （只有直播有效）
 *
 * @return 返回当前视频清晰度
 * 如果和设置的不一致，设置无效保存原有清晰度。
 * 设置成功刷新直播。
 */
-(VHallMovieDefinition)setDefinition:(VHallMovieDefinition)definition;

//直播视频清晰度

typedef NS_ENUM(NSInteger, VHallMovieDefinition)
{
    VHallMovieDefinitionOrigin = 0,          //原画
    VHallMovieDefinitionUHD = 1,             //超高清
    VHallMovieDefinitionHD = 2,               //高清
    VHallMovieDefinitionSD = 3,               //标清
    VHallMovieDefinitionAudio = 4            //无视频，即单音频
};
```

10、 销毁播放器

```
/**
 * 销毁播放器
 */
-(void)destroyMoivePlayer;
```

11、 看直播事件响应

VHMoviePlayerDelegate 在文件 OpenCONSTS.h 中

```
/**
 * 播放连接成功
 */
- (void)connectSucceed:(VHMoviePlayer*)moviePlayer info:(NSDictionary*)info;

/**
 * 缓冲开始回调
 */
- (void)bufferStart:(VHMoviePlayer*)moviePlayer info:(NSDictionary*)info;

/**
 * 缓冲结束回调
 */
```

```

-(void)bufferStop: (VHMoviePlayer*)moviePlayer info: (NSDictionary*)info;
/**
 * 下载速率的回调
 * @param moviePlayer
 * @param info 下载速率信息 单位kbps
 */
- (void)downloadSpeed: (VHMoviePlayer*)moviePlayer info: (NSDictionary*)info;
/**
 * 网络状态的回调
 * @param moviePlayer
 * @param info 网络状态信息 content的值越大表示网络越好
 */
- (void)netWorkStatus: (VHMoviePlayer*)moviePlayer info: (NSDictionary*)info;
/**
 * 该直播支持的清晰度列表
 * @param definitionList 支持的清晰度列表
 */
-(void)VideoDefinitionList: (NSArray*)definitionList
/**
 * 播放时错误的回调
 * @param livePlayErrorType 直播错误类型
 */
-(void)playError: (LivePlayErrorType)livePlayErrorType info: (NSDictionary*)info;

```

十、 用户标识功能

如果使用聊天和问答功能，需要用户提前调用 **WebApi** 进行创建用户标识操作。详细接口说明，查看 http://e.vhall.com/home/vhallapi/active#user_register_第三方创建用户。

VHallApi.h

1、 用户标识

```

/*!
 * 用户标识(如使用聊天，问答等功能必须用户标识)
 *
 * @param aAccount 账号 需服务器调用微吼API 创建用户标识
 * @param aPassword 密码
 * @param aSuccessBlock 成功的回调
 * @param aFailureBlock 失败的回调
 *
 */
+ (void)loginWithAccount: (NSString *)aAccount

```



```
password:(NSString *)aPassword
success:(void (^)(void))aSuccessBlock
failure:(void (^)(NSError *error))aFailureBlock;
```

2、 退出当前帐号

```
/*!
 * 退出当前账号
 *
 * @param aSuccessBlock 成功的回调
 * @param aFailureBlock 失败的回调
 *
 * @result 错误信息
 */
+ (void)logout:(void (^)(void))aSuccessBlock
               failure:(void (^)(NSError *error))aFailureBlock;
```

3、 获取当前用户标识状态

```
/*!
 * 获取当前用户标识状态
 *
 * @result 当前是否已标识
 */
+ (BOOL)isLoggedIn;
```

4、 获取当前用户状态

```
/*!
 * 获取当前用户标识
 *
 * @result 获取当前用户标识
 */
+ (NSString *)currentAccount;
```

十一、 聊天功能

如果使用聊天和问答功能，需要用户提前调用 WebApi 进行创建用户标识操作。详细接口说明，查看 http://e.vhall.com/home/vhallapi/active#user_register_第三方创建用户。
VHallChat.h

1、 创建聊天实例

```
_chat = [[VHallChat alloc] init];
```

2、 设置代理

```
_chat.delegate = self;
```

3、 发送聊天内容

```
/**
 * 发送聊天内容
 * 成功回调成功Block
 * 失败回调失败Block
 * 失败Block中的字典结构如下:
 *   key:code 表示错误码
 *   value:content 表示错误信息
 */
- (void)sendMsg:(NSString *)msg success:(void (^)(void))success failed:(void (^)(NSDictionary* failedData))reslutFailedCallback;
```

4、 接收上下线消息（代理方法）

```
/**
 * 接收上下线消息
 * 代理方法先设置delegate属性
 * 接收到的VHallOnlineStateModel实例数组
 */
- (void)reciveOnlineMsg:(NSArray *)msgs;
```

5、 接收聊天消息（代理方法）

```
/**
 * 接收聊天消息
 * 代理方法先设置delegate属性
 * 接收到的VHallChatModel实例数组
 */
- (void)reciveChatMsg:(NSArray *)msgs;
```

十二、 问答功能

如果使用聊天和问答功能，需要用户提前调用 WebApi 进行创建用户标识操作。详细接口说明，查看 http://e.vhall.com/home/vhallapi/active#user_register 第三方创建用户。
VHallQAndA.h

1、 创建问答实例

```
_QA = [[VHallQAndA alloc] init];
```

2、 设置代理

```
_QA.delegate = self;
```

3、 发送问题

```
/**
 * 发送问题
 * 成功回调成功Block
 * 失败回调失败Block
 * 失败Block中的字典结构如下:
 *   key:code 表示错误码
 *   value:content 表示错误信息
 */
- (void)sendMsg:(NSString *)msg success:(void (^)(void))success failed:(void
(^)(NSDictionary* failedData))reslutFailedCallback;
```

4、 接收问答消息（代理方法）

```
/**
 * 接收问答消息
 * 代理方法先设置delegate属性
 * 接收到的VHallQAModel实例数组
 */
- (void)reciveQAMsg:(NSArray *)msgs;
```

十三、 文档演示功能

当直播活动类型为“视频+文档”或“音频+文档”时，通过以下方法可集成观看，文档会与视频或音频播放同步。

文档 VHallMoviePlayer.h 里的修改内容如下：

1、新增枚举

```

/**
 * 视频播放模式
 */
typedef NS_ENUM(NSInteger, VHALLMovieVideoPlayMode) {
    VHALLMovieVideoPlayModeNone = 0, //不存在
    VHALLMovieVideoPlayModeMedia = 1, //单视频
    VHALLMovieVideoPlayModeTextAndVoice = 2, //文档+声音
    VHALLMovieVideoPlayModeTextAndMedia = 3, //文档+视频
};

/**
 * 活动状态
 */
typedef NS_ENUM(NSInteger, VHALLMovieActiveState) {
    VHALLMovieActiveStateNone = 0,
    VHALLMovieActiveStateLive = 1, //直播
    VHALLMovieActiveStateReservation = 2, //预约
    VHALLMovieActiveStateEnd = 3, //结束
    VHALLMovieActiveStateReplay = 4, //回放
};

```

2、新增代理方法

```

/**
 * 包含文档 获取翻页图片路径
 *
 * @param changeImage 图片更新
 */
- (void)PPTScrollNextPagechangeImagePath:(NSString*)changeImagePath;


/**
 * 获取视频播放模式
 *
 * @param playMode 视频播放模式
 */
- (void)VideoPlayMode:(VHALLMovieVideoPlayMode)playMode;

/**
 * 获取视频活动状态
 *
 * @param playMode 视频活动状态
 */
- (void)ActiveState : (VHALLMovieActiveState)activeState;

```

3、代理方法使用：

注释：新增代理方法，只在视频直播和回放请求成功后会返回对应返回值；请求失败，返回值为 nil；遵从的方法如下图：

文件位置：  VHallMoviePlayer.h

M

```

/**
 * 观看直播视频
 *
 * @param param
 * param[@"id"] = 活动Id 必传
 * param[@"app_key"] = 必传
 * param[@"name"] = 必传
 * param[@"email"] = 必传
 * param[@"pass"] = (活动如果有K值或密码需要传)
 * param[@"app_secret_key"] = 必传
 */
-(BOOL)startPlay:(NSDictionary*)param;

/**
 * 观看直播视频 (仅HLS可用)
 *
 * @param param
 * param[@"id"] = 活动Id 必传
 * param[@"app_key"] = 必传
 * param[@"name"] = 必传
 * param[@"email"] = 必传
 * param[@"pass"] = (活动如果有K值或密码需要传)
 * param[@"app_secret_key"] = 必传
 *
 * @param moviePlayerController MPMoviePlayerController 对象
 */
-(void)startPlay:(NSDictionary*)param moviePlayer:(MPMoviePlayerController *)moviePlayerController;

/**
 * 观看回放视频 (仅HLS可用)
 *
 * @param param
 * param[@"id"] = 活动Id 必传
 * param[@"app_key"] = 必传
 * param[@"name"] = 必传
 * param[@"email"] = 必传
 * param[@"pass"] = (活动如果有K值或密码需要传)
 * param[@"app_secret_key"] = 必传
 *
 * @param moviePlayerController MPMoviePlayerController 对象
 */
-(void)startPlayback:(NSDictionary*)param moviePlayer:(MPMoviePlayerController *)moviePlayerController;

```

代理方法实现:

请参考 SDK demo 文件:

以下文件均可



WatchHLSViewController.m



WatchPlayBackViewController.m



WatchRTMPViewController.m

方法实现:

注释:

-(void)PPTScrollNextPagechangeImagePath:(NSString*)changeImagePath,

此方法中, 图片缓存处理请开发者根据自己需求处理图片缓存

```

-(void)PPTScrollNextPagechangeImagePath:(NSString *)changeImagePath
{
    self.textImageView.image = [UIImage imageWithData:[NSData dataWithContentsOfURL:[NSURL URLWithString:changeImagePath]]];
}
-(void)VideoPlayMode:(VHallMovieVideoPlayMode)playMode
{
    VHLog(@"---%ld", (long)playMode);
}
-(void)ActiveState:(VHallMovieActiveState)activeState
{
    VHLog(@"activeState-%ld", (long)activeState);
}

```

十四、 常量定义及错误码定义

//设置摄像头取景方向

typedef NS_ENUM(int, DeviceOrgiation)

{

```

    kDevicePortrait,
    kDeviceLandSpaceRight,
    kDeviceLandSpaceLeft
};

//发直播分辨率
typedef NS_ENUM(int, VideoResolution)
{
    kLowVideoResolution = 0,           //低分边率      352*288
    kGeneralVideoResolution,          //普通分辨率    640*480
    kHVideoResolution,                //高分辨率      960*540
    kHDVideoResolution                 //超高分辨率    1280*720
};

//发直播/看直播时直播状态
typedef NS_ENUM(int, LiveStatus)
{
    kLiveStatusBufferingStart = 0,     //播放缓冲开始
    kLiveStatusBufferingStop  = 1,     //播放缓冲结束
    kLiveStatusPushConnectSucceed =2,  //直播连接成功
    kLiveStatusPushConnectError =3,    //直播连接失败
    kLiveStatusCDNConnectSucceed =4,   //播放CDN连接成功
    kLiveStatusCDNConnectError =5,     //播放CDN连接失败
    kLiveStatusParamError =6,          //参数错误
    kLiveStatusRecvError =7,           //播放接受数据错误
    kLiveStatusSendError =8,           //直播发送数据错误
    kLiveStatusDownloadSpeed =9,       //播放下载速率
    kLiveStatusUploadSpeed =10,        //直播上传速率
    kLiveStatusNetworkStatus =11,      //当前的网络状态, >= 0为正常, < 0 代表卡顿
    kLiveStatusGetUrlError =12,        //获取推流地址失败
    kLiveStatusWidthAndHeight =13,     //返回播放视频的宽和高
    kLiveStatusAudioInfo  =14          //音频流的信息
};

//观看直播错误类型
typedef NS_ENUM(int, LivePlayErrorType)
{
    kLivePlayGetUrlError = kLiveStatusGetUrlError, //获取服务器rtmpUrl错误
    kLivePlayParamError = kLiveStatusParamError,  //参数错误
    kLivePlayRecvError  = kLiveStatusRecvError,    //接受数据错误
    kLivePlayCDNConnectError = kLiveStatusCDNConnectError, //CDN链接失败
    kLivePlayJsonFormalError = 15                  //返回json格式错误
};

```

```
};

//RTMP 播放器View的缩放状态
typedef NS_ENUM(int, RTMPMovieScalingMode)
{
    kRTMPMovieScalingModeNone,          // No scaling
    kRTMPMovieScalingModeAspectFit,     // Uniform scale until one dimension fits
    kRTMPMovieScalingModeAspectFill,    // Uniform scale until the movie fills the
    visible bounds. One dimension may have clipped contents
};
```

十五、 DEMO 简介

DEMO 只针对核心功能进行演示，不包括 UI 界面设计。

主要测试参数说明：

- 1) 活动 ID: 指的是客户创建的一个直播活动的唯一标识，Demo 测试时可从 e.vhall.com 的控制台页面上获取到
- 2) Token: Demo 测试时可从 <http://e.vhall.com/api/test> 页面，调用接口 [verify/access-token](#) 获取到，有效期为 24 小时
- 3) 码率设置: 主要用于视频编码设置，码率与视频的质量成正比，默认值 300，单位 Kbps
- 4) 缓冲时间: 延时观看时间
- 5) 分辨率: 352*288/640*480/960*540/1280*720
- 6) K 值: 默认为空，指的是控制直播观看权限的参数，具体使用说明参考[第三方 K 值验证](#)

客户 Server 端需要提供如下信息：

- 1) 活动 Id: 通过客户 Server 端接口获取到，此接口需调用 VHALL 接口 [webinar/list](#) 获取
- 2) AccessToken: 通过客户 Server 端接口获取到，此接口需调用 VHALL 接口 [verify/access-token](#) 获取。

基础参数配置填写：

找到 Demo 中 CONSTS.h 文件，找到以下代码进行每一项的填写。

```
//接口文档说明: http://e.vhall.com/home/vhallapi
#define DEMO_AppKey          @"替换成您自己的 AppKey"          //AppKey 详见:  ▪
API&SDK 权限申请
#define DEMO_AppSecretKey    @"替换成您自己的 AppSecretKey"    //AppSecretKey
#define DEMO_ActivityId      @"" //活动 id 详见:  ▪ 自助式网络直播 API -> 活动管理
#define DEMO_AccessToken     @"" //直播 Token 详见:  ▪ 自助式网络直播 API -> 观众管理
->verify/access-token

#define DEMO_account         @"" //账号 详见:  ▪ 自助式网络直播 API -> 活动管理
->user/register 创建用户标识
```

#define DEMO_password @"" //密码 详见：· 自助式网络直播 API -> 活动管理
->user/register 创建用户标识

用户标识：

有些功能模块需要用户标识后才可正常使用，比如聊天、问答。

帐号和密码：可通过以下方式获得

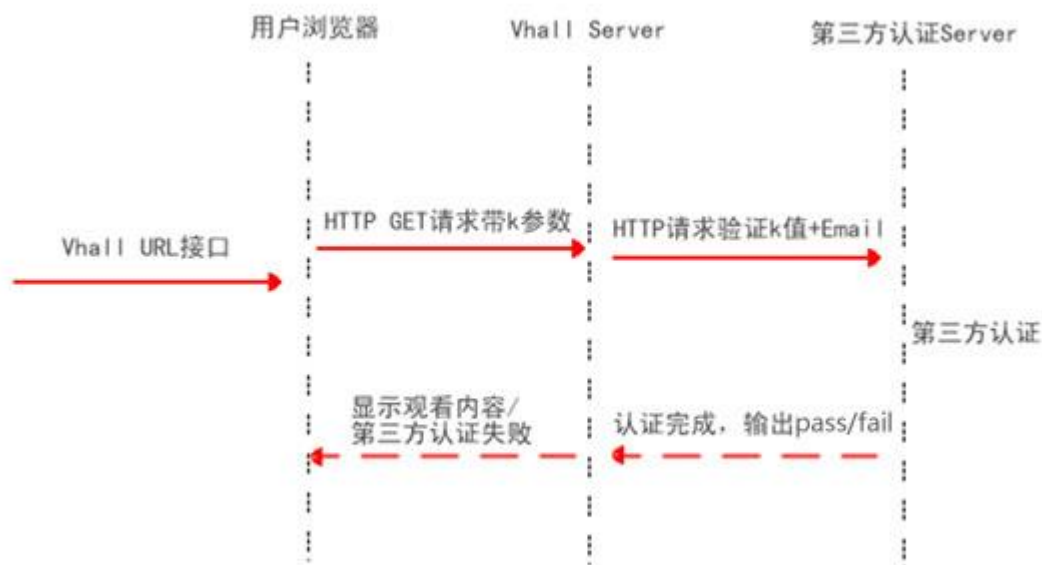
1) 通过接口调用创建用户标识：http://e.vhall.com/home/vhallapi/active#user_register [第三方创建用户](#)

十六、 第三方 K 值认证

观看直播、观看回放的权限控制，支持使用客户的权限验证逻辑。

具体可参考：<http://e.vhall.com/home/vhallapi/embed>

1、 认证流程



2、 开启设置

2.1 第三方回调接口设置

1) 全局设置： 针对所有的活动配置生效，如果针对单个活动再做配置，以单个活动配置为最终配置。接口调用设置接口：[webinar/whole-auth-url](#) 全局配置第三方 K 值验证 URL

2) 针对某个活动的配置方式一：通过页面配置 <http://e.vhall.com/webinar/auth/123456789>，数字表示自己帐号下的活动 id

3) 针对某个活动的配置方式二：通过接口([webinar/create](#) 或 [webinar/update](#))设置

4) 接口参数: `use_global_k` , 默认为 0 不开启, 1 为开启, 是否针对此活动开启全局 K 值配置; 当设置为 0 后, 则以单个活动的配置为最终配置。

2.2 Vhall 接口 URL 中请务必带上 `k` 参数, 如果这个参数为空或者没有这个参数, 则视为认证失败

2.3 Vhall 系统收到用户的接口访问请求后, 会向第三方认证 URL(`auth_url`)发送 HTTP POST 请求, 同时将 `email` 和 `k` 值作为 POST 数据提交 给第三方认证。由第三方系统验证 `k` 值的合法性。如果认证通过, 第三方认证 URL(`auth_url`)返回字符串 `pass`, 否则的返回 `fail`

注: 需要确保您的回调地址支持 `multipart/form-data` 方式接收 `post` 数据。

2.4 Vhall 系统根据第三方认证 URL 返回值判断认证是否成功。只有收到 `pass`, 才能认定为验证成功, 否则一律跳转到指定的认证失败 URL, 或者提示'非法访问'

3、 K 值使用

1) 网页嵌入或SDK里的调用方法, 请务必带上`k`参数, 如果这个参数为空或者没有这个参数, 则视为认证失败

●网页嵌入地址类似:

`http://e.vhall.com/webinar/inituser/123456789?email=test@vhall.com&name=visitor&k=随机字符串`

●SDK里的调用方法, 需要传递3个参数`name`, `email`, `pass`

email: 可选参数, 如果不填写系统会随机生成邮箱地址。由于`email`自身的唯一性, 我们推荐使用`email`来作为唯一标识有效用户的字段。对于第三方自有用户数据的系统, 也可以使用一些特征ID作为此标识, 请以`email`的格式组织, 比如在第三方系统中, 用户ID为123456, 可在其后添加一个`@domain.com`, 组成`123456@domain.com`形式的`email`地址。

name: 可选参数, 如果不填写系统会随机生成。此字段表示用户昵称、姓名或其他有意义的字符串。可以为中文, 但必须为UTF-8, 且经过URL编码(`urlencode`)。

k: 可选参数, 此字段为了提供给第三方可以根据自己的权限系统, 验证客户是否可访问直播地址。

/**

* 观看直播视频

*

* @param param

* param["@id"] = 活动Id 必传

* param["@name"] = 如未用户标识, SDK要传

* param["@email"] = 如未用户标识, SDK要传

* param["@pass"] = 活动如果有K值或密码需要传

*

/-(BOOL)startPlay:(NSDictionary)param;

/**

* 观看回放视频 (仅HLS可用)

*

* @param param

```

* param[@"id"] = 活动Id 必传
* param[@"name"] = 如未用户标识, SDK要传
* param[@"email"] = 如未用户标识, SDK要传
* param[@"pass"] = 活动如果有K值或密码需要传
*
* @param moviePlayerController MPMoviePlayerController 对象
*/-(void)startPlayback:(NSDictionary*)param moviePlayer:(MPMoviePlayerController*)moviePlayerController;

```

2) Vhall系统收到用户的接口访问请求后, 会向第三方认证URL(auth_url)发送HTTP POST请求, 同时将email和k值作为POST数据提交 给第三方认证。由第三方系统验证k值的合法性。如果认证通过, 第三方认证URL(auth_url)返回字符串pass, 否则的返回fail

注: 需要确保您的回调地址支持 multipart/form-data 方式接收 post 数据。

3) Vhall 系统根据第三方认证URL返回值判断认证是否成功。只有收到pass, 才能认定为验证成功, 否则一律跳转到指定的认证失败 URL, 或者提示'非法访问'

4) 参数特征

URL请求很容易被探测截获, 这就要求第三方系统生成的K值必须有以下特征:

- 唯一性: 每次调用接口必须产生不同的K值
- 时效性: 设定一个时间范围, 超时的K值即失效。
- 如果包含有第三方系统内部信息, 必须加密和混淆过。

5) 建议的K值实现

第三方系统可以考虑K值元素包括: 用户ID、Vhall直播ID、时间戳(1970-01-01至今的秒数)元素组合后加密后, 使用Base64或者hex 匹配成URL可识别编码。K值在第三方系统中持久化或放在Cache中

回调验证时, 根据时间戳判断是否在设定时间内有效

验证结束, 若认证通过, 则从DB或Cache中移除K值

DB或Cache建议有时效性控制, 自动失效或定期清理过期数据

十七、 版本迁移重点说明

1、 2.3.0 迁移到 2.4.0

1) 绑定应用签名信息

使用 SDK 前集成前, 务必先配置好此签名信息, 否则使用时会出现“身份验证失败”提示信息。

- 进入 <http://e.vhall.com/home/vhallapi/authlist> , API/SDK 使用权限信息页面。
- 选择已开通的应用进行编辑操作。
- 点下一步进入应用绑定页面。
- 选择 IOS-SDK 切页后输入安全码 BundleID 项。(Bundle Identifier 在项目 Targets 的 General 中找到, 如下图)



2) AppDelegate.m

```
#import "VHallApi.h"

- (BOOL)application:(UIApplication *)application
didFinishLaunchingWithOptions:(NSDictionary *)launchOptions
{
    [VHallApi registerApp:AppKey SecretKey:AppSecretKey]; //新增
}
```

3) VHallLivePublish.h

```
- (void)startLive:(NSDictionary*)param; // 参数发生变化，去掉 AppKey 和 AppSecretKey
- (void)stopLive; // 结束直播 用于替换原来 disconnect 方法 与 startLive 成对出现，
如果调用 startLive，则需要调用 stopLive 以释放相应资源
- (void)disconnect; // 方法不再用于结束直播，只用于手动断开直播流，断开推流
的连接，注意 app 进入后台时要手动调用此方法、切回到前台需 reconnect 重新推流。
(注：特别需要使用 disconnect 的地方都改成 stopLive)
bitRate -> videoBitRate // 比特率属性变为 videoBitRate
```

4) VHallMoviePlayer.h

```
-(BOOL)startPlay:(NSDictionary*)param; // 参数发生变化，去掉 AppKey 和 AppSecretKey
-(void)startPlayback:(NSDictionary*)param moviePlayer:(MPMoviePlayerController
*)moviePlayerController; // 参数发生变化，去掉 AppKey 和 AppSecretKey
```