

微吼直播 SDK for iOS

微吼直播 中国领先的商务视频直播平台

| | | |
|----|---------------------------------------|-----------|
| 一、 | 修订记录..... | 7 |
| 二、 | 简介 | 7 |
| 三、 | 主要流程图 | 8 |
| 四、 | 权限开通申请 | 9 |
| 五、 | SDK 使用准备 | 10 |
| 1、 | 下载 SDK&DEMO | 10 |
| 2、 | 开发环境要求 | 10 |
| 3、 | ATS 支持情况 | 10 |
| 4、 | 添加库 | 10 |
| 5、 | Demo 结构调整 | 11 |
| 6、 | 添加 framework | 14 |
| 六、 | 获取当前 SDK 版本号 | 14 |
| 七、 | 打印日志..... | 15 |
| 八、 | APPKEY 和 APPSECRETKEY 设置 | 15 |
| 九、 | 发起直播..... | 15 |
| 1、 | 初始化直播引擎..... | 15 |
| 2、 | 设置采集设备 | 15 |
| 3、 | 初始化音频设备 | 16 |
| 4、 | 开始视频采集 | 16 |
| 5、 | 开始发起直播 | 16 |
| 6、 | 停止直播..... | 16 |
| 7、 | 切换摄像头 | 16 |
| 8、 | 手动对焦..... | 17 |

| | | |
|-----|--|----|
| 9、 | 变焦 | 17 |
| 10、 | 设置闪关灯的模式 | 17 |
| 11、 | 重连流 | 17 |
| 12、 | 断开推流的连接，停止直播 | 17 |
| 13、 | 销毁直播引擎 | 18 |
| 14、 | 发直播事件响应 | 18 |
| 十、 | 观看直播 | 19 |
| 1、 | 初始化 VHMoviePlayer 对象 | 19 |
| 2、 | 观看直播（RTMP） | 20 |
| 3、 | 观看直播视频 （仅 HLS 可用,2.3.4 以后版本不维护此功能） | 20 |
| 4、 | 观看回放视频 （仅 HLS 可用） | 20 |
| 5、 | 设置静音 | 21 |
| 6、 | 设置系统声音大小 | 21 |
| 7、 | 获取系统声音大小 | 21 |
| 8、 | 停止播放 | 21 |
| 9、 | 清晰度切换/切换到单音频 | 21 |
| 10、 | 销毁播放器 | 22 |
| 11、 | 看直播事件响应 | 22 |
| 十一、 | 登录功能（VHALLAPI.H） | 24 |
| 1、 | 登录 | 24 |
| 2、 | 退出当前帐号 | 24 |
| 3、 | 获取当前登录状态 | 25 |
| 4、 | 获取当前登录用户账号 | 25 |
| 十二、 | 聊天功能 | 25 |

| | | |
|-----|--|----|
| 1、 | 创建聊天实例 | 25 |
| 2、 | 设置代理..... | 25 |
| 3、 | 发送聊天内容 | 25 |
| 4、 | 接收上下线消息（代理方法） | 26 |
| 5、 | 接收聊天消息（代理方法） | 26 |
| 十三、 | 问答功能..... | 26 |
| 1、 | 创建问答实例 | 26 |
| 2、 | 设置代理..... | 26 |
| 3、 | 发送问题..... | 26 |
| 4、 | 接收问答消息（代理方法） | 27 |
| 十四、 | 抽奖功能..... | 27 |
| 1、 | 创建抽奖实例 | 27 |
| 2、 | 设置代理..... | 27 |
| 3、 | 提交个人中奖信息 | 27 |
| 4、 | 接收抽奖开始消息（代理方法） | 27 |
| 5、 | 接收抽奖结果消息（代理方法） | 28 |
| 十五、 | 文档演示功能 | 28 |
| 1、 | 新增枚举 | 28 |
| 2、 | 新增代理方法..... | 28 |
| 3、 | 代理方法使用： | 29 |
| 十六、 | 美颜功能..... | 30 |
| 1、 | 配置是否启用美颜功能(如果启用美颜功能必须包含美颜功能滤镜库) | 30 |
| 2、 | 初始化 VHallLivePublishFilter 类，初始化方法请参考 VhallLivePublish | 30 |
| 3、 | 开启美颜功能所需额外配置..... | 30 |

| | |
|---|----|
| 4、id <VHallLivePublishFilterDelegate> GPUFilterDelegate | 31 |
| 十七、 回放评论功能 | 31 |
| 1、 发表评论 | 31 |
| 2、 获取历史评论 | 32 |
| 十七、公告功能 | 32 |
| 1、公告代理方法(VHallMoviePlayerDelegate)..... | 32 |
| 2、实例化公告展示 UI..... | 33 |
| 十八、 签到功能..... | 33 |
| 1、开始签到代理方法 | 33 |
| 2、距离签到结束剩余时间代理方法 | 33 |
| 3、签到结束代理方法 | 34 |
| 4、签到 | 34 |
| 5、取消签到 | 35 |
| 十九、问卷功能 | 35 |
| 1、获取问卷消息 （需实现 VhallSurveyDelegate） | 35 |
| 2、根据问卷 ID 获取问卷详情 | 35 |
| 3、发送问卷结果..... | 36 |
| 二十、画笔白板功能..... | 36 |
| 1、初始化 VHDocumentView..... | 36 |
| 2、接收画笔以及白板绘制消息 | 37 |
| 二十一、观看 VR 视频功能..... | 37 |
| 1、实现 VHALLMOVIEPLAYERDELEGATE 代理方法..... | 37 |
| 2、陀螺仪设置是否开启（参见 VHMoviewPlayer.h） | 37 |

| | |
|--|----|
| 3、直播流格式（默认 RTMP） | 38 |
| 二十二、常量定义及错误码定义 | 38 |
| 二十三、DEMO 简介 | 39 |
| 二十四、第三方 K 值认证 | 40 |
| a) 认证流程 | 40 |
| b) 开启设置 | 41 |
| c) K 值使用 | 41 |
| 二十五、版本迁移重点说明 | 43 |
| 1、2.7.0 迁移到 2.8.0 2.9.0 注意事项 | 43 |
| libVinnylive.a -> libVhallLiveApi.a | 43 |
| 需要删除原来的 libVinnylive.a 库 | 43 |
| 添加 libVhallLiveApi.a | 43 |
| 2、2.3.0 迁移到 2.4.0 | 43 |
| 二十六、CAMERAENGINE_RTMP.H 文件说明 | 44 |
| 1、属性注释 | 44 |
| 二十七、VHMOVIEPLAYER.H 文件说明 | 48 |
| 1、属性注释 | 48 |
| 二十八、OPENCONSTS.H 文件说明 | 50 |
| 1、OpenCONSTS.h 文件定义 VHallSDK 使用中所需的枚举值 | 50 |

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 | 新增登录 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 |
| 2016-11-10 | V2.5.0 | 新增抽奖功能 新增获取 20 条最近聊天记录功能 | wxx |
| 2016-12-20 | V2.5.3 | 新增美颜功能， 评论相关功能 支持 MP4 回放 支持 Https 协议 | wxx |
| 2017-3-1 | V2.6.0 | 新增公告功能 新增签到功能 | wxx |
| 2017-3-13 | V2.7.0 | V2.7.0 新增问卷功能 | cy |
| 2017-3-30 | V2.7.1 | 新增画笔白板功能 | cy |
| 2017-4-13 | V2.8.0 | 观看 VR 视频和陀螺仪功能 | cy |
| 2017-5-27 | V2.9.0 | 发起错误码调整 性能优化 发直播初始化接口修正 | wxx |

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：

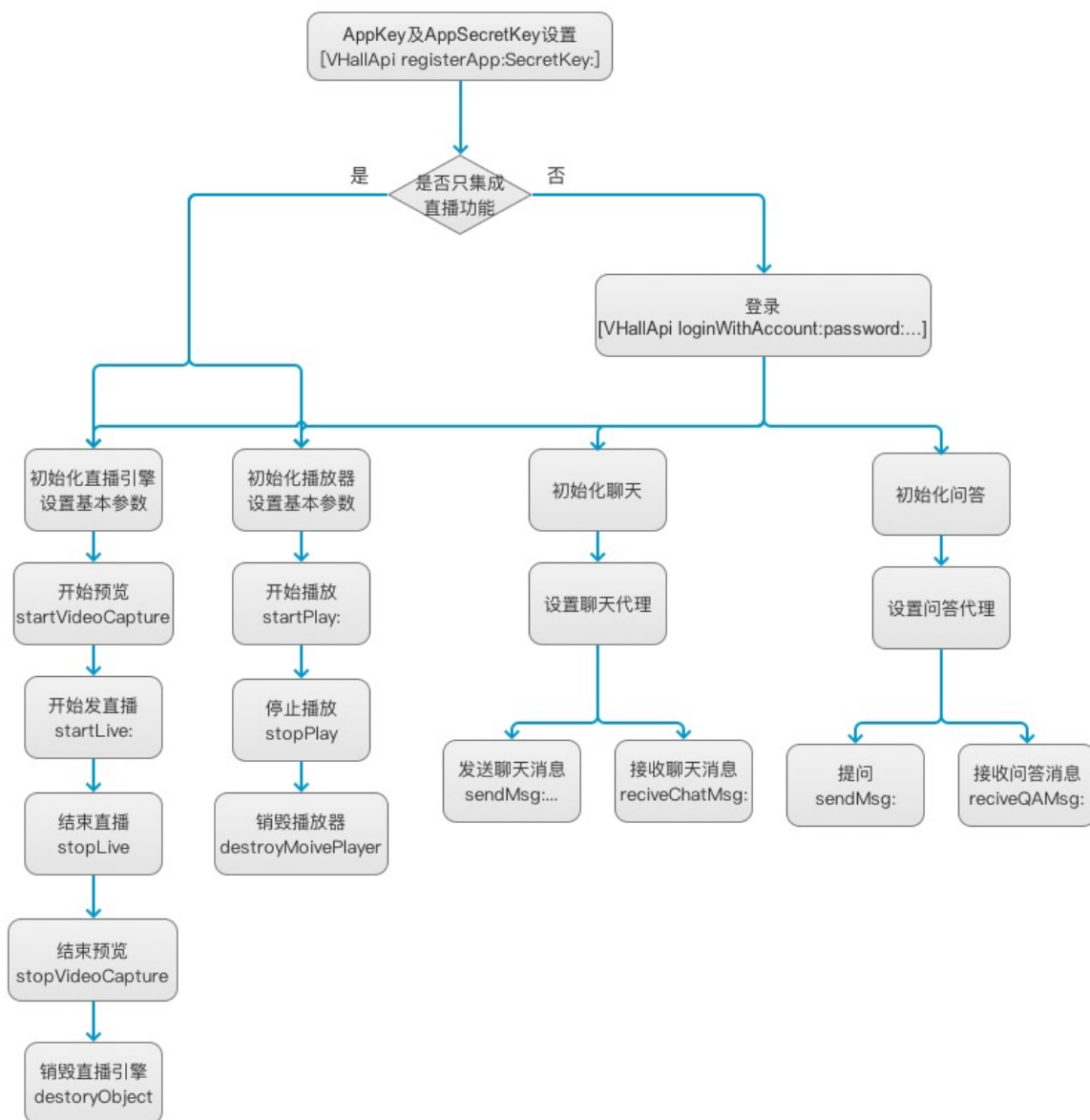
| 分类 | 特性名称 | 描述 |
|------|--------|----------------------------------|
| 发起直播 | 支持编码类型 | 音频编码：AAC，视频编码：H.264 |
| | 支持推流协议 | RTMP |
| | 视频分辨率 | 352*288/640*480/960*540/1280*720 |
| | 屏幕朝向 | 横屏、竖屏 |
| | 闪光灯 | 开/关 |
| | 静音 | 开/关 |
| | 切换摄像头 | 前、后置摄像头 |
| | 目标码率 | ios 8.0 及以上系统（支持使用硬编）， |

| | | |
|------------|-----------|------------------------------|
| | | iphone 5s 及以下设备（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 ， MP4 |
| 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 |
| 用户登录（new） | 支持微吼用户登录 | 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 | 用户登录后可聊天 |
| | 支持看直播聊天 | 用户登录后可聊天 |
| 问答（new） | 支持看直播提问 | 用户登录后可提问 |
| 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，需要提前服务器端注册用户，用户登录后才可正常使用。



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。

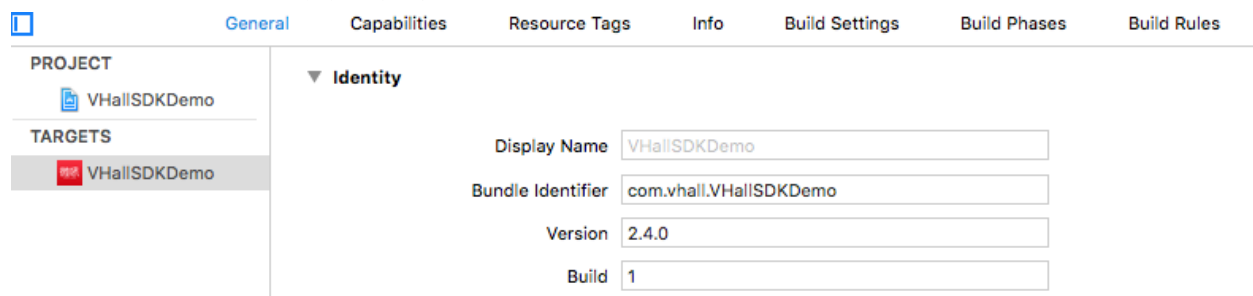
审核通过后，可以获取开发应用的权限信息：App_Key、App Secret_Key， [立即查看](#)。

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、 ATS 支持情况

支持

4、 添加库

添加系统库

libc++.tbd

libz2.1.0.tbd

libcucore.tbd

libz.tbd

libiconv.tbd

CoreTelephony.framework

MediaPlayer.framework

AVFoundation.framework

VideoToolbox.framework

AssetsLibrary.framework

OpenAL.framework

OpenGL ES.framework

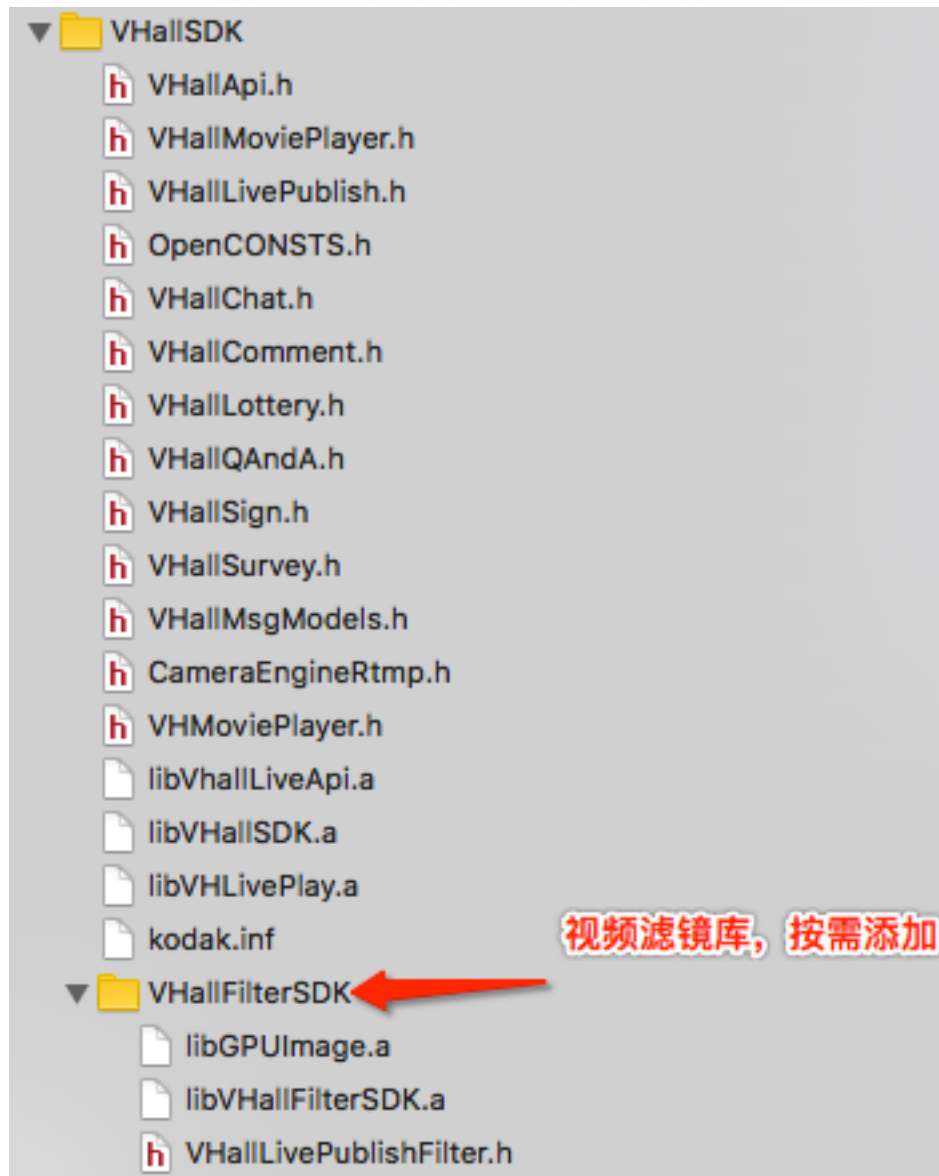
QuartzCore.framework

CoreMedia.framework

Security.framework

JavaScriptCore.framework

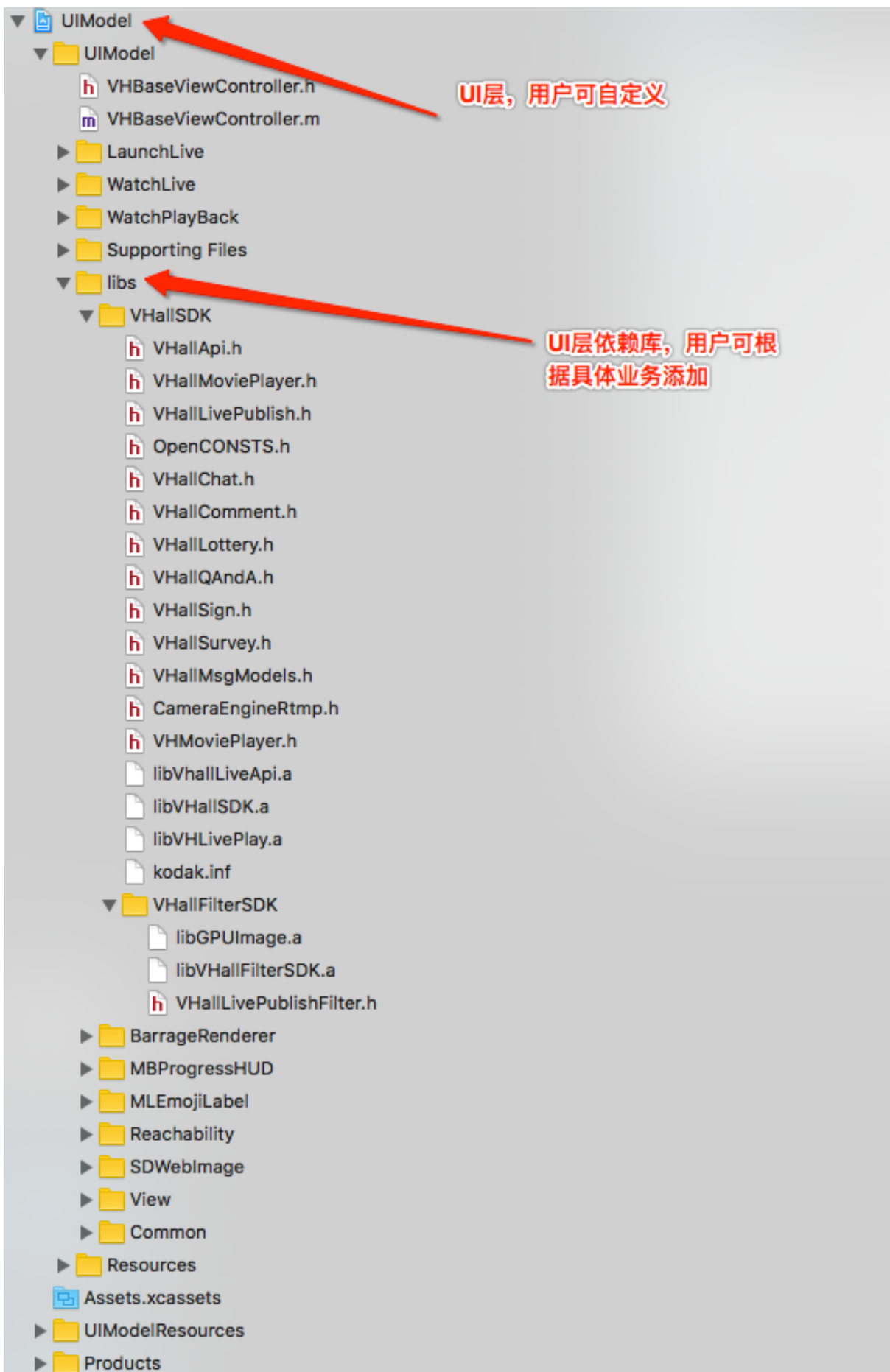
将 VHallSDK 文件夹添加到工程中，添加后的效果如下图所示：

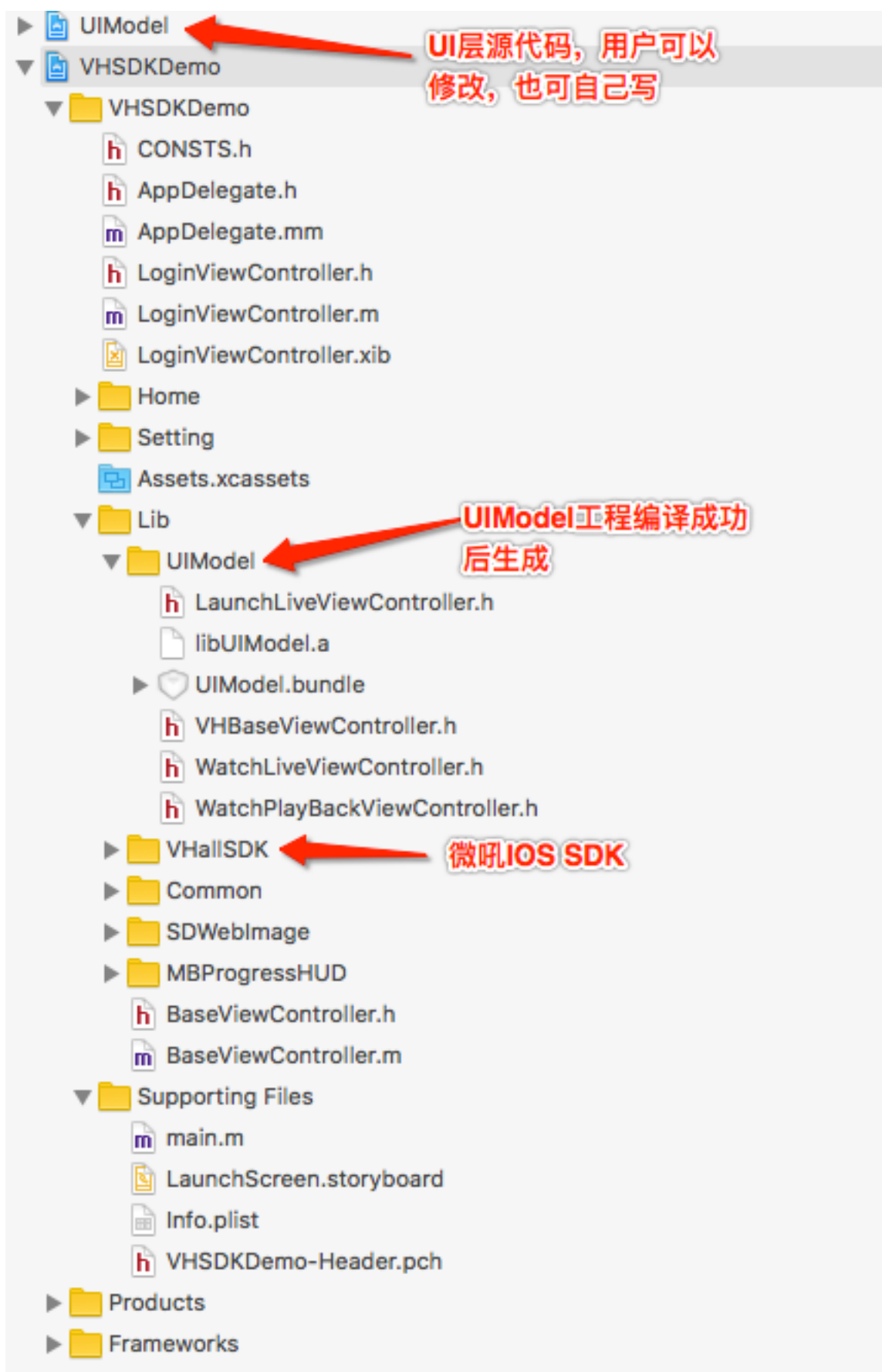


5、 Demo 结构调整

通过 VHSDKDemo.xcworkspace 打开 demo 源码 包括两部分

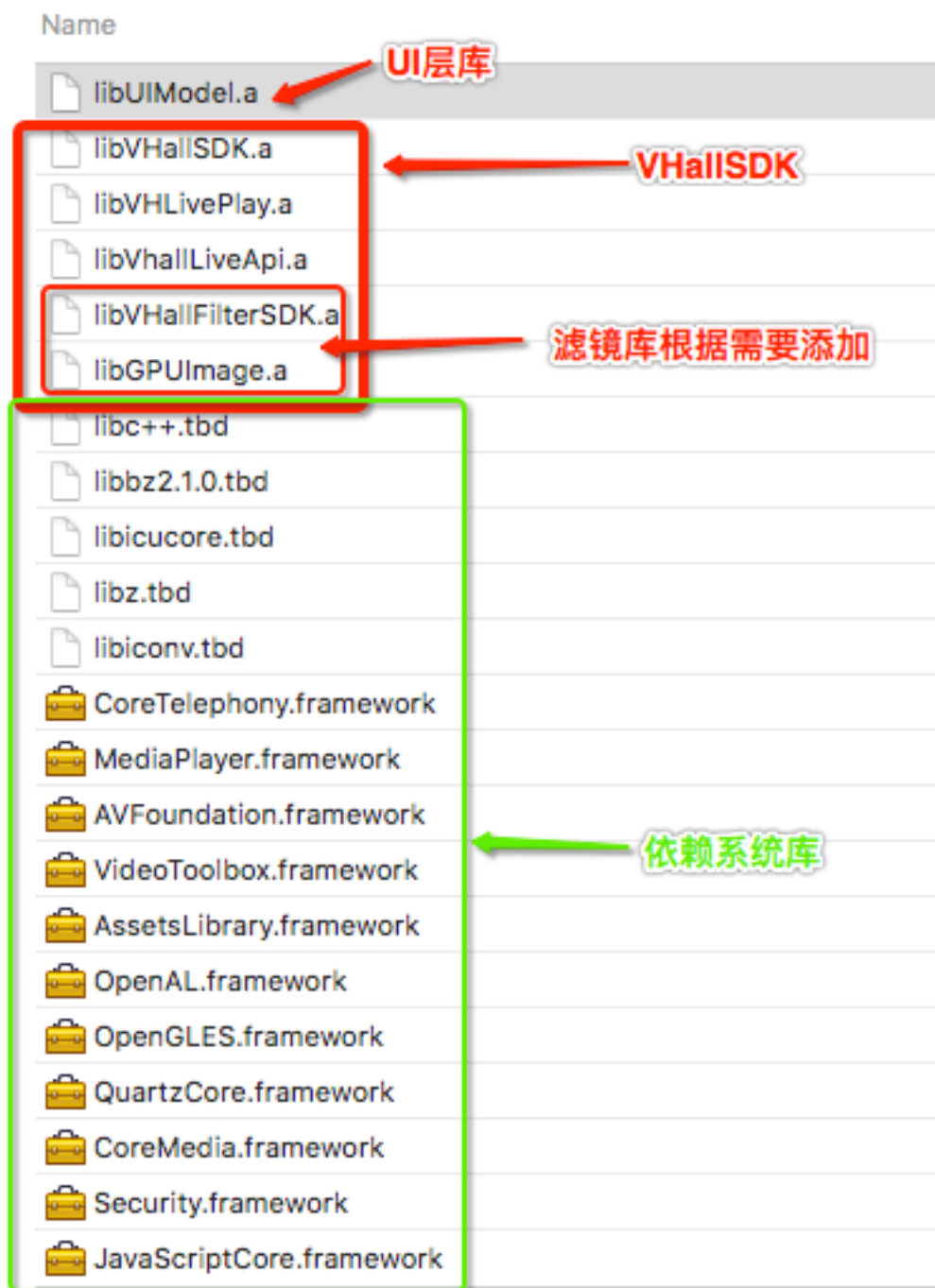
- 1、 UIModel 此部分是微吼对发起和观看端进行了简单封装，并提供源码，客户可以直接修改使用，也可作为微吼 SDK 调用 demo 做参考使用
- 2、 VHSDKDemo demo 层只有登录、设置 、直播入口，所有直播相关都在 uimodel 中 用户可以修改 UIModel 完成页面快速定制





6、 添加 framework

▼ Linked Frameworks and Libraries



六、 获取当前 SDK 版本号

VHallApi.h

为了有效、高效定位客户问题，SDK 版本号可做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、 打印日志

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。

```

VHLogType_OFF    = 0,    //关闭日志 默认设置
VHLogType_ON     = 1,    //开启日志
VHLogType_ALL    = 2,    //开启全部日志
+ (void)setLogType:(VHLogType)type;

```

八、 AppKey 和 AppSecretKey 设置

VHallApi.h

```

/**
 * app注册
 * 需要在 application:(UIApplication *)application
didFinishLaunchingWithOptions:(NSDictionary *)launchOptions 中调用
 * @param appKey    vhall后台注册生成的appkey
 * @param appKey    vhall后台注册生成的appsecretKey
 */
+ (void)registerApp:(NSString *)appKey SecretKey:(NSString *)secretKey;

```

九、 发起直播

VHallLivePublish.h 发直播其他属性查看 CameraEngineRtmp.h

1、 初始化直播引擎

```

/**
 * 初始化 直播引擎
 * @param orientation视频拍摄方向
 *
 * @return 是否成功
 */
- (id)initWithOrientation:(DeviceOrientation)orientation;

```

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 查看 文档中常量定义和错误码定义
 * @param info 直播状态对应信息，字典格式 {@"code":xx,@"content": xxx}
 * code: 状态码/错误码，如枚举值 (0,1, ...) content: 对应的状态或错误说
 * 明,如播放缓冲开始、播放缓冲结束等
 */

-(void)publishStatus:(LiveStatus)liveStatus withInfo:(NSDictionary*)info;
```

//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
    };

```

返回 info 结构{ notiType: 3, content: 错误码 }

notiType 即 LiveStatus

kLiveStatusGetUrlError 发直播错误码

10030 身份验证出错

10401 活动开始失败

10402 当前活动 ID 错误

10403 活动不属于自己

10409 第三方用户对象不存在 【新】

10410 第三方用户对象不存在 【新】

10411 用户套餐余额不足 【新】

当 liveStatus == kLiveStatusPushConnectError 时，content 代表出错原因

```

4001    握手失败
4002    链接 vhost/app 失败
4003    网络断开 （预留，暂时未使用）
4004    无效 token
4005    不再白名单中
4006    在黑名单中
4007    流已经存在
4008    流被禁掉 （预留，暂时未使用）
4009    不支持的视频分辨率（预留，暂时未使用）
4010    不支持的音频采样率（预留，暂时未使用）
4011    欠费

```

十、 观看直播

VHallMoviePlayer.h

1、 初始化 VHMoviewPlayer 对象

```

/**
 * 初始化VHMoviewPlayer对象
 *
 * @param delegate
 *
 * @return 返回VHMoviewPlayer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"] = 未登录用户必须传值，用户昵称，已登录用户可传也可以不传
 * param[@"email"] = 未登录用户必须传值，用户邮箱，如果无邮箱需填写设备唯一
标识保证唯一性，已登录用户可传也可以不传
 * param[@"pass"] = K值，已创建的活动如果有K值需要传，没有可不传
 *
 */
-(BOOL)startPlay:(NSDictionary*)param;
```

3、 观看直播视频 （仅 HLS 可用,2.3.4 以后版本不维护此功能）

```
/**
 * 观看直播视频 （仅HLS可用）
 *
 * @param param
 * param[@"id"] = 活动Id 必传
 * param[@"name"] = 未登录用户必须传值，用户昵称，已登录用户可传也可以不传
 * param[@"email"] = 未登录用户必须传值，用户邮箱，如果无邮箱需填写设备唯一标
识保证唯一性，已登录用户可传也可以不传
 * param[@"pass"] = K值，已创建的活动如果有K值需要传，没有可不传
 *
 * @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"] = 未登录用户必须传值，用户昵称，已登录用户可传也可以不传
 * param[@"email"] = 未登录用户必须传值，用户邮箱，如果无邮箱需填写设备唯一
标识保证唯一性，已登录用户可传也可以不传
```

```

* param[@"pass"] = K值，已创建的活动如果有K值需要传，没有可不传

*
* @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) destroyMoviePlayer;

```

11、看直播事件响应

VHMoviePlayerDelegate 在文件 OpenCONSTS.h 中

```

/**
 * 播放连接成功
 * @param moviePlayer 微吼播放器实例
 * @param info 直播状态对应信息，字典格式 @{@"code":xx, @"content": xxx}
 * code: 状态码，如枚举值 (0, 1, ...) content: 对应的状态说明
 */
- (void) connectSucceed: (VHMoviePlayer*) moviePlayer info: (NSDictionary*) info;

/**
 * 缓冲开始回调
 * @param moviePlayer 微吼播放器实例
 * @param info 直播状态对应信息，字典格式 @{@"code":xx, @"content": xxx}
 * code: 状态码，如枚举值 (0, 1, ...) content: 对应的状态说明
 */
- (void) bufferStart: (VHMoviePlayer*) moviePlayer info: (NSDictionary*) info;

```

```

    * 缓冲结束回调
    * @param moviePlayer 微吼播放器实例
    * @param info 直播状态对应信息，字典格式{@"code":xx, @"content": xxx}
    * code: 状态码，如枚举值 (0,1, ...) content: 对应的状态说明
    */
-(void)bufferStop:(VHMoviePlayer*)moviePlayer info:(NSDictionary*)info;
/**
    * 下载速率的回调
    * @param moviePlayer 微吼播放器实例
    * @param info 下载速率信息 单位kbps，字典格式{@"code":xx, @"content": xxx}
    * code: 状态码，如枚举值 (0,1, ...) content: 对应的下载速率
    */
-(void)downloadSpeed:(VHMoviePlayer*)moviePlayer info:(NSDictionary*)info;
/**
    * 网络状态的回调
    * @param moviePlayer 微吼播放器实例
    * @param info 网络状态信息，字典格式{@"code":xx, @"content": xxx}
    * code: 状态码，如枚举值 (0,1, ...) content: 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
    * @param info 播放器错误信息，字典格式{@"code":xx, @"content": xxx}
    * code: 错误码，如枚举值 (0,1, ...) content: 错误信息说明
    */
-(void)playError:(LivePlayErrorType)livePlayErrorType info:(NSDictionary*)info;

//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
    };
    kLivePlayGetUrlError 播放网络请求错误码

```

10030 身份验证出错
 10402 当前活动 ID 错误
 10405 录播不存在
 10049 访客数据信息不全
 10404 KEY 值验证出错
 10046 当前活动已结束
 10405 微吼用户 ID 错误
 10047 您已被踢出，请联系活动组织者
 10048 活动现场太火爆，已超过人数上限
 10410 用户信息不存在

十一、 登录功能（VHallApi.h）

如果使用聊天和问答功能，需要用户提前调用 WebApi 进行注册用户操作。否则提示用户不存在。详细接口说明，查看 http://e.vhall.com/home/vhallapi/active#user_register_第三方创建用户。

1、 登录

```

//*!
 * 登录（如使用聊天，问答等功能必须登录）
 *
 * @param aAccount      账号 需服务器调用微吼注册API 注册该用户账号密
码
 * @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 (^)())aSuccessBlock
                failure:(void (^)(NSError *error))aFailureBlock;

```

3、 获取当前登录状态

```

/*!
* 获取当前登录状态
*
* @result 当前是否已登录
*/
+ (BOOL)isLoggedIn;

```

4、 获取当前登录用户账号

```

/*!
* 获取当前登录用户账号
*
* @result 前登录用户账号
*/
+ (NSString *)currentAccount;

```

十二、 聊天功能

如果使用聊天和问答功能，需要用户提前调用 WebApi 进行注册用户操作。否则提示用户不存在。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(^)())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;

```

十三、 问答功能

如果使用聊天和问答功能，需要用户提前调用 WebApi 进行注册用户操作。否则提示用户不存在。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))resultFailedCallback;

```

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)recvQAMsg:(NSArray *)msgs;

```

十四、 抽奖功能

请详见 VHallLottery.h 文件的说明

1、 创建抽奖实例

```
_lottery= [[VHallLotteryalloc] init];
```

2、 设置代理

```
_lottery.delegate = self;
```

3、 提交个人中奖信息

```

/**
* 提交个人中奖信息
* @param info 个人信息
*      key:user_name 用户名
*      key:phone 电话
* 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)submitLotteryInfo:(NSDictionary *)info success:(void (^)(void))success
failed:(void (^)(NSDictionary *failedData))resultFailedCallback;

```

4、 接收抽奖开始消息（代理方法）

```

/**
 * 接收抽奖开始消息
 * 代理方法先设置delegate属性
 * 接收到的VHallStartLotteryModel实例
 */
- (void)startLottery:(VHallStartLotteryModel *)msg;

```

5、 接收抽奖结果消息（代理方法）

```

/**
 * 接收抽奖结果消息
 * 代理方法先设置delegate属性
 * 接收到的VHallEndLotteryModel实例
 */
- (void)endLottery:(VHallEndLotteryModel *)msg;

```

十五、 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
 * 观看端获取直播端的播放模式枚举
 */
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
 * 观看端获取直播端活动播放状态枚举
 */
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
 *
 * 视频播放中，如包含PPT文档，获取文档最新页面Path；空文档Path是空值
 * @param changeImage 图片更新
 */
- (void)PPTScrollNextPagechangeImagePath:(NSString*)changeImagePath;
/**
 * 获取视频播放模式
 *
 * 播放端，返回直播端播放布局类型（播放布局类型：纯视频 / 文档+视频 / 文档+语音）
 * @param playMode 视频播放模式
 */
- (void)VideoPlayMode:(VHallMovieVideoPlayMode)playMode;
/**
 * 获取视频活动状态
 *
 * 获取直播端播放状态
 * @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

```

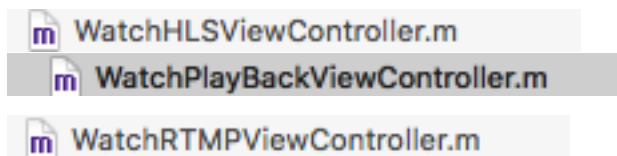
/**
 * 观看直播视频
 *
 * @param param
 * param[@"id"] = 活动Id 必传
 * param[@"app_key"] = 必传
 * param[@"name"] = 必传
 * param[@"email"] = 必传
 * param[@"pass"] = (活动如果有K值或密码需要传)
 * param[@"app_secret_key"] = 必传
 */
- (BOOL)startPlay:(NSDictionary*)param;
/**
 * 观看直播视频 (仅HLS可用)
 *
 * @param param
 * param[@"id"]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



方法实现:

注释:

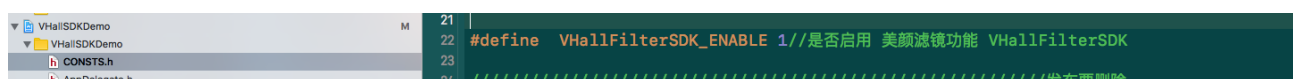
-(void)PPTScrollNextPagechangeImagePath:(NSString*)changeImagePath,
此方法中, 图片缓存处理请开发者根据自己需求处理图片缓存

```
-(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);
}
```

十六、 美颜功能

请详见 VHallLivePublishFilter.h(此类继承 VHallLivePublish)具有 VHallLivePublish 所有功能(使用美颜功能工程中已集成 CPUImage, 如有冲突不要加载 VhallFilterSDK/libImage.a) 使用此功能建议设置码率 1000k 以上

1、配置是否启用美颜功能(如果启用美颜功能必须包含美颜功能滤镜库)



2、初始化 VHallLivePublishFilter 类, 初始化方法请参考 VhallLivePublish

```
self.engine = [[VHallLivePublishFilter alloc] initWithOrgiation:deviceOrgiation];
BOOL ret = [_engine initCaptureVideo:AVCaptureDevicePositionFront];
```

3、开启美颜功能所需额外配置

```
_engine.openFilter = YES;
[_engine setBeautifyFilterWithBilateral:10.0f Brightness:1.0f Saturation:1.0f];
```

注释: openFilter=YES 为开启美颜

/**

* setBeautifyFilterWithBilateral: Brightness: Saturation: 设置 VHall 美颜滤镜参数
GPUFilterDelegate == nil 时有效

```

*   @param distanceNormalizationFactor // A normalization factor for the distance
between central color and sample color.
*   @param brightness // The brightness adjustment is in the range [0.0,
2.0] with 1.0 being no-change.
*   @param saturation // The saturation adjustment is in the range [0.0,
2.0] with 1.0 being no-change.
*   return BOOL YES 设置成功 NO 设置失败
*/

```

4、id <VHallLivePublishFilterDelegate> GPUFilterDelegate

GPUFilterDelegate 滤镜代理，在代理方法中添加您自己的滤镜

注释：

(1) 默认为 nil,只有使用自己的滤镜的情况下设置此代理

(2) 必须发直播前设置

(3) 如果此属性不为 nil 时，SDK 自带美颜功能失效，使用代理中设置的滤镜发起直播

十七、 回放评论功能

请详见 VHallComment.h 文件的说明

创建评论实例

```
_comment = [[VHallComment alloc] init];
```

1、 发表评论

```

/**
 * 发表评论内容
 * 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
 */
- (BOOL)sendComment:(NSString *)comment success:(void(^)( ))success failed:(void
(^)(NSDictionary* failedData))reslutFailedCallback;

```

2、 获取历史评论

/** 注意（当 A 用户发完一条评论，B 立即拉取评论内容，此时数据库会返回一条重复数据，第三方用户可以根据返回模型里面的 **commentId** 进行去重处理）

*获取历史评论记录 在进入回放活动后调用

*@param limit 每次拉取数据条数，默认每次 20 条，最多 50 条

*@param pos 从第几条数据开始获取，默认 0

* 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)getHistoryCommentPageCountLimit:(NSInteger)limit offset:(NSInteger) pos
success:(void(^)(NSArray *msgs))success failed:(void(^)(NSDictionary *
faileData))reslutFailedCallback
```

十七、公告功能

1、公告代理方法(VHallMoviePlayerDelegate)

/**

* 播主发布公告

*

* 播主发布公告消息

*/

```
-(void)Announcement:(NSString*)content publishTime:(NSString*)time;
```


2、实例化公告展示 UI

```
5 #pragma mark - Announcement
7 - (void)Announcement:(NSString*)content publishTime:(NSString*)time
8 {
9     NSLog(@"公告:%@",content);
10
11     if (!announcementContentDic)
12     {
13         announcementContentDic = [[NSMutableDictionary alloc] init];
14     }
15     [announcementContentDic setObject:content forKey:@"announceContent"];
16     [announcementContentDic setObject:time forKey:@"announceTime"];
17
18     if(!announcementView)
19     {
20         announcementView = [[AnnouncementView alloc] initWithFrame:CGRectMake(0, 0, _showView.width,
21                                     35) content:content time:nil];
22         announcementView.content = [content stringByAppendingString:time];
23         [_showView addSubview:announcementView];
24     }
25 }
```

十八、 签到功能

请详见 VHallSign.h 文件的说明需实现（VHallSignDelegate）

1、开始签到代理方法

```
/**
 * 开始签到
 *
 * 开始签到消息
 */
- (void)startSign;
```

2、距离签到结束剩余时间代理方法

```
/**
 * 距签到结束剩余时间
 *
 * 距签到结束剩余时间
 */
```

- (void)signRemainingTime:(NSTimeInterval)remainingTime;

3、签到结束代理方法

```
/**
 * 签到结束
 *
 * 签到结束消息
 */
- (void)stopSign;
```

4、签到

```
/**
 * 签到
 * 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
 * 10010 活动不存在
 * 10011 不是该平台下的活动
 * 10017 活动id 不能为空
 * 10807 用户id不能为空
 * 10813 签名ID不能为空
 * 10814 用户名称不能为空
 * 10815 当前用户已签到
 */
-(BOOL)signSuccess:(void(^)(void))success failed:(void (^)(NSDictionary*
failedData))reslutFailedCallback;
```

```

331 #pragma mark - VHallSignDelegate
332 - (void)startSign
333 {
334     NSLog(@"开始签到");
335     __weak typeof(self) weakSelf = self;
336     [SignView showSignBtnClickedBlock:^(BOOL{
337         [weakSelf SignBtnClicked];
338         return NO;
339     }]);
340 }
341
342 - (void)SignBtnClicked
343 {
344     __weak typeof(self) weakSelf = self;
345     [_sign signSuccess:^(
346         [SignView close];
347         [weakSelf showMsg:@"签到成功" afterDelay:2];
348     ) failed:^(NSDictionary *failedData) {
349         [weakSelf showMsg:[NSString stringWithFormat:@"%@", 错误码%@", failedData[@"content"], failedData[@"code"]] afterDelay:2];
350         [_sign cancelSign];
351         [SignView close];
352     }];
353 }

```

5、取消签到

```

/**
 * 取消签到
 *
 * 取消签到
 */
- (void)cancelSign;

```

十九、问卷功能

详情参见 VHallSurvey.h

1、获取问卷消息（需实现 VHallSurveyDelegate）

```

/**
 * 接收问卷消息
 *
 *
 */
- (void)receiveSurveyMsgs: (NSArray*)msg;

```

2、根据问卷 ID 获取问卷详情

```

/**

```

```

* 获取问卷内容
*
* @param surveyId          调查问卷Id
* @param webId             当前活动Id
* @param success           成功回调成功Block 返回问卷内容
* @param reslutFailedCallback 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)getSurveyContentWithSurveyId:(NSString*)surveyId
webInarId:(NSString*)webId success:(void(^)(VHallSurvey* msgs))survey failed:(void
(^)(NSDictionary* failedData))reslutFailedCallback;

```

3、发送问卷结果

```

/**
* 发送完成问卷
*
* 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:(NSArray *)msg success:(void(^)( ))success failed:(void
(^)(NSDictionary* failedData))reslutFailedCallback;

```

二十、画笔白板功能

1、初始化 VHDocumentView

VHDocumentView 负责展示 PPT、白板以及画笔渲染，使用之前需要初始化。初始化位置为收到 PPT 翻页消息代理方法中（ - (void) PPTScrollNextPagechangeImagePath:(NSString*)changeImagePath（具体看 demo 实例）

2、接收画笔以及白板绘制消息

需要实现 `VHallMoviePlayerDelegate` 中的协议（`docList` 为文档画笔数据, `boardList` 为白板画笔数据）

```
/**
 * 画笔
 *
 *
 */
- (void)docHandList:(NSArray*)docList whiteBoardHandList:(NSArray*)boardList
```

二十一、观看 VR 视频功能

1、实现 `VHallMoviePlayerDelegate` 代理方法

```
/**
 * 获取当前视频播放模式
 *
 * @param playMode 视频播放模式
 */
-(void)VideoPlayMode:(VHallMovieVideoPlayMode)playMode
isVrVideo:(BOOL)isVrVideo;
根据 isVideo 返回值选择正确渲染视图模式（具体代码参见 WatchLiveViewController）
```

2、陀螺仪设置是否开启（参见 `VHMoviePlayer.h`）

```
/**
 * 是否使用陀螺仪，仅VR播放时可用
 */
- (void)setUsingGyro:(BOOL)usingGyro;
```

具体代码参见 `WatchLiveViewController`

3、直播流格式（默认 RTMP）

```
//直播流格式
typedef NS_ENUM(int, LiveFormat)
{
    kLiveFormatNone = 0,
    kLiveFormatRtmp,
    kLiveFormatFlv
}
```

具体参见 Demo(WatchLiveViewController.m)

二十二、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
};

```

二十三、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
#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
```

用户登录：

有些功能模块需要用户登录后才可正常使用，比如聊天、问答。

帐号和密码：微吼直播官网的帐号名称和登录密码，可通过以下方式获得

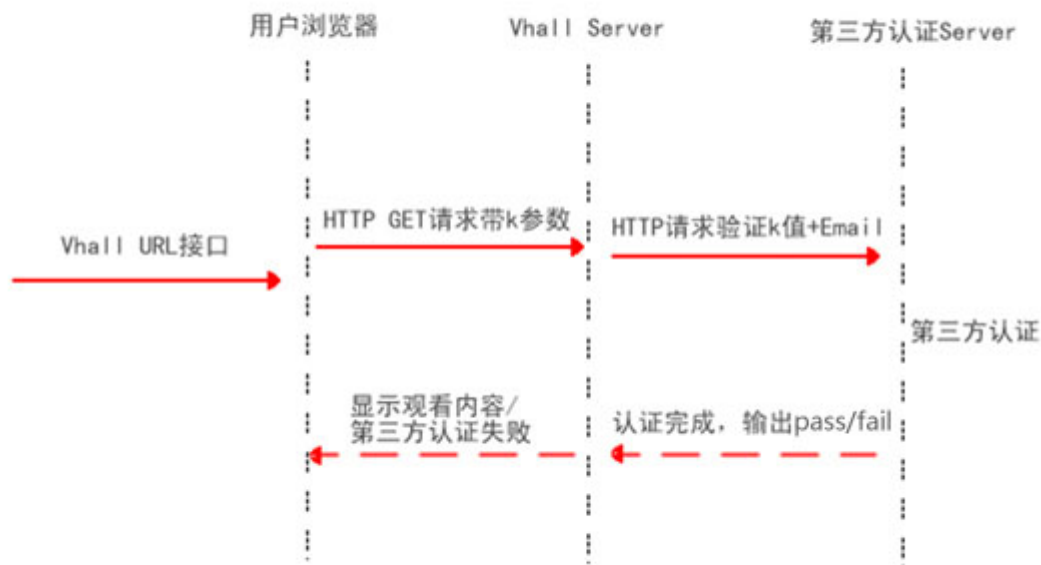
- 1) 通过微吼直播官网注册: <http://e.vhall.com/auth/register>
- 2) 通过接口调用注册: http://e.vhall.com/home/vhallapi/active#user_register 第三方创建用户

二十四、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>

a) 认证流程



b) 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'

c) 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"] = 如未登录SDK要传

* param[@"email"] = 如未登录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"] = 如未登录SDK要传

* param[@"email"] = 如未登录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

二十五、版本迁移重点说明

1、2.7.0 迁移到 2.8.0 2.9.0 注意事项

libVinnyLive.a -> libVhallLiveApi.a

需要删除原来的 libVinnyLive.a 库

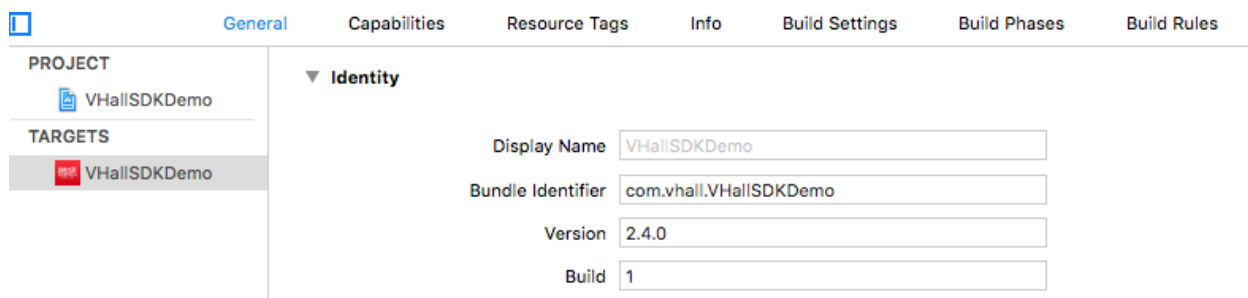
添加 libVhallLiveApi.a

2、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

```

二十六、CameraEngineRtmp.h 文件说明

CameraEngineRtmp.h 为 VHallSDK 所依赖文件，VHallLivePublish 直接继承此文件，不能直接使用 CameraEngineRtmp.h

1、属性注释

```

/**
 * 推流连接的超时时间，单位为毫秒 默认 5000
 */
@property (nonatomic,assign)int publishConnectTimeout;
/**
 * 推流断开重连的次数 默认为 5
 */
@property (nonatomic,assign)int publishConnectTimes;
/**
 * 用来显示摄像头拍摄内容的 View
 */

```

```

@property(nonatomic,strong,readonly)UIView * displayView;

/**
 * 视频采集的帧率 范围 [10~30]
 */
@property(nonatomic,assign) int videoCaptureFPS;

/**
 * 代理
 */
@property(nonatomic,assign)id <CameraEngineRtmpDelegate> delegate;
/**
 * 视频分辨率 默认值是 kGeneralViodeResolution 960*540
 */
@property(nonatomic,assign)VideoResolution videoResolution;
/**
 * 视频码率设置
 */
@property(nonatomic,assign)NSInteger videoBitRate;
/**
 * 音频码率设置
 */
@property(nonatomic,assign)NSInteger audioBitRate;
/**
 * 设置静音
 */
@property(assign,nonatomic)BOOL isMute;
/**
 * 判断用户使用是前置还是后置摄像头
 */
@property(nonatomic,assign,readonly)AVCaptureDevicePosition
captureDevicePosition;
/**
 * 滤镜的回调，在此回调中做滤镜处理
 */
@property (nonatomic, strong) void (^captureVideoBuf)(CMSampleBufferRef
sampleBuffer);

/**
 * 当前推流状态
 */

```

```

@property(assign, nonatomic, readonly) BOOL isPublishing;

/**
 * 是否使用专用蓝牙麦克风 默认 NO 使用系统默认麦克风
 */
@property (nonatomic, assign, readonly) BOOL isUseBluetoothMic;

//采集设备初始化
- (id)initWithOrgiation:(DeviceOrgiation)orgiation;

/**
 * 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;

//初始化音频
- (BOOL)initAudio;

//开始视频采集
- (BOOL)startVideoCapture;

//停止视频采集
- (BOOL)stopVideoCapture;

//开启音频采集;
- (BOOL)startAudioCapture;

//暂停音频采集
- (BOOL)pauseAudioCapture;

//停止音频采集,此方法调用了，就要在调用 initAudio 方法初始化
- (BOOL)stopAudioCapture;

/**
 * 开始直播
 *
 * @param liveId 活动 id

```

```

    * @param token    token
    * @param rtmpUrl 推流 host
    */
- (void)setRoomId:(NSString *)liveId
           token:(NSString *)token
      rtmpUrl:(NSString *)rtmpUrl
pushType:(VHallStreamType)streamType;
/**
 * 切换摄像头
 *
 * @param captureDevicePosition
 *
 * @return 是否切换成功
 */
- (BOOL)swapCameras:(AVCaptureDevicePosition)captureDevicePosition;

//手动对焦
-(void)setFoucsFoint:(CGPoint)newPoint;

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

/**
 * 设置闪关灯的模式
 */
- (BOOL)setDeviceTorchModel:(AVCaptureTorchMode)captureTorchMode;

/**
 * 断网后重练
 */
-(BOOL)reconnect;

/**
 * 销毁初始化数据，同步销毁，如果感觉销毁太慢，可以开线程去销毁
 */
- (void)destoryObject;

/**
 * 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;

/**
 * 推送视频数据
 *
 * @param sampleBuffer YUV420sp 数据
 */
- (void)pushVideoData:(CMSampleBufferRef)sampleBuffer;

//直播状态
-(void)liveStatus:(NSNotification*)notification;

```

二十七、VHMoviePlayer.h 文件说明

1、属性注释

```

@property(nonatomic,assign)id <VHMoviePlayerDelegate> delegate;
@property(nonatomic,strong,readonly)UIView * moviePlayerView;
@property(nonatomic,assign)int timeout; //RTMP 链接的超时时间 默认 5 秒,
单位为毫秒
@property(nonatomic,assign)int reConnectTimes; //RTMP 断开后的重连次数 默认
2 次
@property(nonatomic,assign)int bufferTime; //RTMP 的缓冲时间 默认 2 秒 单位为
秒 必须>0 值越小延时越小,卡顿增加
@property(assign,readonly)int realityBufferTime; //获取 RTMP 播放实际的缓冲时间
/**
 * 推流格式
 */
@property(assign,nonatomic)LiveFormat liveFormat;
/**
 * 视频 View 的缩放比例 默认是自适应模式
 */
@property(nonatomic,assign)RTMPMovieScalingMode movieScalingMode;

/**
 * 初始化 VHMoviePlayer 对象

```



```

*
* @param delegate
*
* @return 返回 VHMoviePlayer 的一个实例
*/
- (instancetype)initWithDelegate:(id <VHMoviePlayerDelegate>)delegate;
/**
 * 设置渲染视图 在 startPlayWithURL:之前设置，之后设置无效
 */
- (void)setRenderViewModel:(VHallRenderModel)renderModel;
/**
 * 设置监控日志的参数 注意参数是 json string,开始直播前设置，之后设置无效
 * param:{
 * "vid":"直播发起者账号",
 * "vfid":"直播发起者父账号",
 * "guid":"观众唯一标识",
 * "vtype":"直播视频类别",
 * "topic":"活动话题"
 * }
 * return 0 设置成功，-1 是 json 解析失败
 */
- (int)setMonitorLogParam:(NSString*)param;
/**
 * 开始观看
 *
 * @param urls rtmpUrls(注意数组中的地址是完整地址)
 */
- (void)startPlayWithURL:(NSArray *)urls;

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

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

```

```

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

/**
 * 停止播放
 */
- (void)stopPlay;

/**
 * 清空视频剩余的最后一帧画面
 */
- (void)cleanLastFrame;

/**
 * 是否使用陀螺仪，仅 VR 播放时可用
 */
- (void)setUsingGyro:(BOOL)usingGyro;

/**
 * 设置视频布局的方向，仅 VR 模式可用,切要开启陀螺仪
 */
- (void)setUILayoutOrientation:(DeviceOrientation)orientation;

/**
 * 销毁播放器，异步销毁的
 */
- (void)destroyMoivePlayer;

//直播状态的通知
- (void)liveStatues:(NSNotification*)notification;

```

二十八、OpenCONSTS.h 文件说明

1、OpenCONSTS.h 文件定义 VHallSDK 使用中所需的枚举值

```

/**
 * 打开 VHall Debug 模式
 *
 * @param enable true 打开 false 关闭
 */
extern void EnableVHallDebugModel(BOOL enable);

//设置摄像头取景方向
typedef NS_ENUM(int,DeviceOrientation)
{
    kDevicePortrait,
    kDeviceLandSpaceRight,
    kDeviceLandSpaceLeft
};

//直播流格式
typedef NS_ENUM(int,LiveFormat)
{
    kLiveFormatNone = 0,
    kLiveFormatRtmp,
    kLiveFormatFLV
};

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
};

typedef NS_ENUM(int,LiveStatus)
{
    kLiveStatusNone = -1,
    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,      //保留字段，暂时无用
    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
    kLiveStatusAudioRecorderError  =15, //音频采集失败，提示用户查看权限或者
    重新推流，切记此事件会回调多次，直到音频采集正常为止
    kLiveStatusUploadNetworkException=16, //发起端网络环境差
    kLiveStatusUploadNetworkOK = 17,   //发起端网络环境恢复正常
    kLiveStatusCDNStartSwitch = 18,    //CDN 切换
    kLiveStatusRecvStreamType = 19     //接受流的类型
};

```

```

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
};

```

//流类型

```

typedef NS_ENUM(int,VHallStreamType)
{
    kVHallStreamTypeNone = 0,
    kVHallStreamTypeVideoAndAudio,
    kVHallStreamTypeOnlyVideo,
    kVHallStreamTypeOnlyAudio,
};

```

```

typedef NS_ENUM(int,VHallRenderModel){
    kVHallRenderModelNone = 0,
    kVHallRenderModelOrigin, //普通视图的渲染
    kVHallRenderModelDewarpVR, //VR 视图的渲染
};

@protocol CameraEngineRtmpDelegate <NSObject>
/**
 * 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
 */
-(void)publishStatus:(LiveStatus)liveStatus withInfo:(NSDictionary*)info;
/**
 * code 含义
 * 10030 身份验证出错
 * 10401 活动开始失败
 * 10402 当前活动 ID 错误
 * 10403 活动不属于自己编辑
 */
@end

@class VHMMoviePlayer;

@protocol VHMMoviePlayerDelegate <NSObject>

@optional
/**
 * 播放连接成功
 */
- (void)connectSucceed:(VHMMoviePlayer*)moviePlayer info:(NSDictionary*)info;
/**
 * 缓冲开始回调
 */
- (void)bufferStart:(VHMMoviePlayer*)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;

/**
 * cdn 发生切换时的回调
 *
 * @param moviePlayer
 * @param info
 */
-(void)cdnSwitch:(VHMoviePlayer*)moviePlayer info:(NSDictionary*)info;

/**
 * Streamtype
 *
 * @param moviePlayer moviePlayer
 * @param info info
 */
-(void)recStreamtype:(VHMoviePlayer*)moviePlayer info:(NSDictionary*)info;

/**
 * 播放时错误的回调
 *
 * @param livePlayErrorType 直播错误类型
 */
-(void)playError:(LivePlayErrorType)livePlayErrorType info:(NSDictionary*)info;

/**
 * code 含义
 * 10030 身份验证出错
 * 10402 当前活动 ID 错误
 * 10404 KEY 值验证出错

```

- * 10046 当前活动已结束
- * 10047 您已被踢出，请联系活动组织者
- * 10048 活动现场太火爆，已超过人数上限
- */