微吼直播SDK for Android

http://cnstatic01.e.vhall.com/static/img/logo_20160907.png

中国领先的商务视频直播平台

目录

[一、修订记录 4](#_Toc471302974)

[二、简介 5](#_Toc471302975)

[1、支持的产品特性如下: 5](#_Toc471302976)

[2、API & SDK的使用框架流程图 6](#_Toc471302977)

[3、SDK主要流程图 6](#_Toc471302978)

[三、权限开通申请 7](#_Toc471302979)

[1、申请Key 7](#_Toc471302980)

[2、绑定应用签名信息 7](#_Toc471302981)

[3、生成当前签名AccessToken 8](#_Toc471302982)

[4、获取包名还有安全码SHA1值 8](#_Toc471302983)

[(1) 包名获取： 8](#_Toc471302984)

[(2) 发行版本安全码SHA1: 8](#_Toc471302985)

[5、客户Server端需提供给APP的信息 10](#_Toc471302986)

[四、SDK集成前必要的准备工作 10](#_Toc471302987)

[1、下载SDK与DEMO 10](#_Toc471302988)

[2、开发环境要求 10](#_Toc471302989)

[3、需要导入的AAR (之前导入的Jar) 10](#_Toc471302990)

[4、添加依赖 10](#_Toc471302991)

[5、权限及配置 10](#_Toc471302992)

[五、流媒体快速接入介绍 11](#_Toc471302993)

[1、初始化配置 11](#_Toc471302994)

[(1) 初始化VhallSDK 11](#_Toc471302995)

[(2)基础参数说明 11](#_Toc471302996)

[2、用户标识 12](#_Toc471302997)

[(1) 创建用户 12](#_Toc471302998)

[(2) 登陆 12](#_Toc471302999)

[(3)登陆参数描述 12](#_Toc471303000)

[3、发直播 13](#_Toc471303001)

[(1) 准备工作： 13](#_Toc471303002)

[(2) 发直播: 14](#_Toc471303003)

[(3) 直播事件回调 15](#_Toc471303004)

[(4) 结束直播 16](#_Toc471303005)

[4、看直播 17](#_Toc471303006)

[(1) 看直播 17](#_Toc471303007)

[(2) 观看事件回调 18](#_Toc471303008)

[(3) 停止观看 19](#_Toc471303009)

[5、看回放 19](#_Toc471303010)

[(1) 看回放 19](#_Toc471303011)

[(2) 观看回放事件回调 20](#_Toc471303012)

[(3) 播放器方法 21](#_Toc471303013)

[(4) 发送/获取评论信息 22](#_Toc471303014)

[6 滤镜直播: 23](#_Toc471303015)

[六、功能接入介绍 24](#_Toc471303016)

[1、聊天服务器相关功能 24](#_Toc471303017)

[(1)、上下线消息通知 26](#_Toc471303018)

[(2)、聊天消息 26](#_Toc471303019)

[(3)、聊天记录 27](#_Toc471303020)

[(4)、问答消息 28](#_Toc471303021)

[2、消息服务器相关功能 29](#_Toc471303022)

[(1)、PPT翻页消息 30](#_Toc471303023)

[(2)、活动结束消息 30](#_Toc471303024)

[(3)、抽奖消息 30](#_Toc471303025)

[3、分辨率切换/切换到单音频 32](#_Toc471303026)

[4、设置观看布局 32](#_Toc471303027)

[1、 文档演示功能 33](#_Toc471303028)

[第三方K值认证 33](#_Toc471303029)

[2、 认证流程 33](#_Toc471303030)

[3、 开启设置 34](#_Toc471303031)

[4、 K值使用 34](#_Toc471303032)

[二、 版本迁移重点说明 35](#_Toc471303033)

[1、 2.3.2迁移到2.4.0 35](#_Toc471303034)

# 一、修订记录

|  |  |  |  |
| --- | --- | --- | --- |
| 日期 | 版本号 | 描述 | 修订者 |
| 2016-04-21 | V2.1.1 | 初稿 | xy |
| 2016-05-06 | V2.2.0 | 新增文档演示 | xy |
| 2016-08-01 | V2.3.1 | 多分辨率方案/防盗链/多展示方案 | xy |
| 2016-09-27 | V2.4.0 | 新增用户标识/聊天/问答/应用签名/观看语音直播 | xy |
| 2016-11-01 | V2.5.0 | 新增子账号/聊天记录/抽奖 |  |
| 2016-11-30 | V2.5.1 | 新增滤镜 |  |
| 2016-12-30 | V2.5.4 | 播放器支持MP4播放，增加观看回放评论 |  |
| 2017-03-01 | V2.6.0 | 签到，公告，弹幕，表情 |  |
| 2017-03-15 | V2.7.0 | 问卷 |  |
| 2017-03-31 | V2.7.1 | 白板、画笔 |  |
| 2017-04-15 | V2.8.0 | VR 陀螺仪 |  |
| 2017-06-12 | V2.9.0 | 添加推流地址调度，添加用户信息数据统计 添加自定义推流数据、自定义渲染 |  |
| 2017-08-22 | V3.0 | 1、丰富SDK数据统计 2、SDK降噪与增益 3、更改SDK设置滤镜方案及API 4、性能优化及bug修复 |  |
| 2017-10-24 | V3.0.1 | 1、新增投屏功能(体验) |  |

# 二、简介

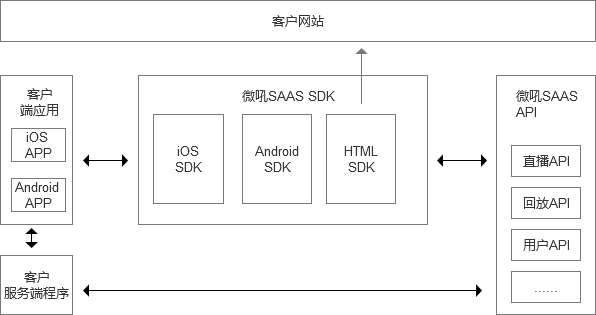
本文档为了指导开发者更快使用Android系统上的 “自助式网络直播服务SDK”，默认读者已经熟悉IDE的基本使用方法（本文以Android Studio为例），以及具有一定的编程知识基础等。

## 新版本重点说明

## 2、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 | 640\*480 |
| 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 | 使用软编，码率固定在300-400之间 |
| 目标帧率 | 帧率默认为10帧 最大可修改到30帧 |
| 支持环境 | Android 4.0以上， |
| 观看直播 | 支持播放协议 | RTMP |
| 延时 | RTMP: 2-5秒 |
| 支持解码 | H.264 |
| 文档演示 | 支持文档演示 | 文档可与视频同步演示 |
| 观看回放 | 支持协议 | HLS |
| 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） | 支持发起观看直播聊天 | 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 |

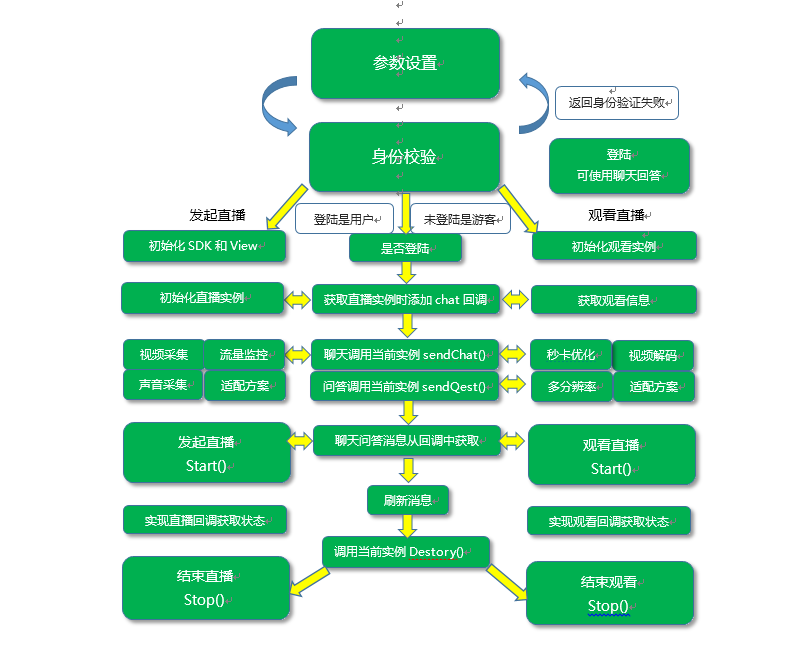
## 3、API & SDK的使用框架流程图



## 4、SDK主要流程图

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。

主要流程设计如下：



# 三、权限开通申请

## 1、申请Key

请点击 [API & SDK权限申请](http://webim.qiao.baidu.com/im/index?siteid=113762&ucid=2052738) 或致电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](http://e.vhall.com/home/vhallapi/authinfo)。

## 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，配置信息流程如下。

* 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Android-SDK切页后输入以下信息：



## 3、生成当前签名AccessToken

当用户成功配置了签名信息，但是可能配置了多个签名信息，比如测试时的Debug签名或者正式的Release签名, 这是生成Token时必须传递参数App\_key 参数，用于识别当前使用哪个签名信息，这个参数在配置签名时会生成。如果只配置了一个签名可以不传，默认会取这唯一的签名，如果配置多个签名，并不传递App\_key 参数，也会出现“**身份验证失败**”提示信息

## 4、获取包名还有安全码SHA1值

### (1) 包名获取：

在Android工程目录下的AndroidManifest.xml清单文件中的package部分即是包名

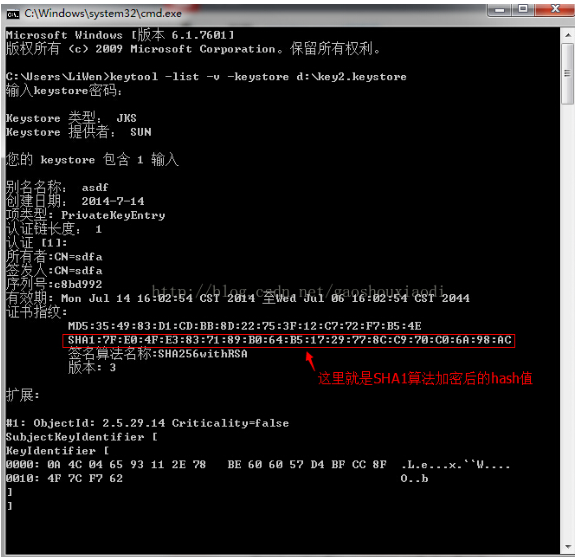
### (2) 发行版本安全码SHA1:

获取这个值有以下三种方法：

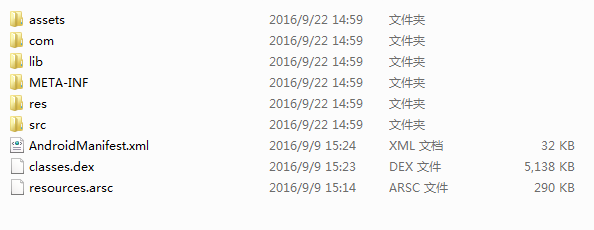
1. 通过keytool工具，在cmd命令中输入 keytool -list -v –keystore 你的签名证书文件

例如在D盘根目录，则输入：keytool -list -v -keystore d:\key2.keystore

回车后输入生成证书的密码，即可得到以下信息，其中红色框部分是SHA1值：

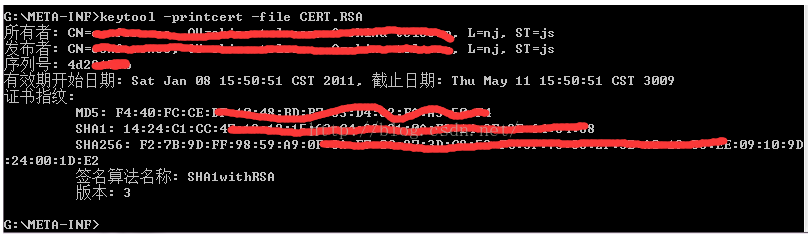


1. 对生成的正式apk安装包文件重新改扩展名字为zip，并且解压缩可以得到类似下面的文件夹，



其中有META-INF目录，双击进入目录后有MANIFEST.MF、CERT.SF和CERT.RSA三个文件，通过在cmd命令进入对应文件目录后中输入

keytool -printcert -file CERT.RSA，可以得到SHA1值，



## 5、客户Server端需提供给APP的信息

客户Server端需要提供如下信息：  
(1) Id: 通过客户Server端接口获取到，此接口需调用vhall接口[webinar/list](http://e.vhall.com/home/vhallapi/active#webinar_list_获取活动列表)获取。  
(2) AccessToken：通过客户端接口获取到，此接口需调用vhall接口[verify/access-token](http://e.vhall.com/home/vhallapi/audience#verify/access-token_获取SDK直播操作token)获取。

# 四、SDK集成前必要的准备工作

## 1、下载SDK与DEMO

Github ：<https://github.com/vhall/vhallsdk_live_android/releases>

## 2、开发环境要求

Pc操作系统：64window系统

JDK: 1.7以上

Android studio : 建议使用Android studio 2.0以上

Android: 4.0以上

备注： Android设备操作系统需要4.0以上, 需要访问手机硬件,暂不支持模拟器开发

## 3、添加依赖

工程直接依赖Module uilibs 即可

## 4、添加依赖

compile 'com.github.bumptech.glide:glide:3.7.0' // 用于加载PPT

compile('io.socket:socket.io-client:0.8.0') {//用于SDK网络连接

exclude group: 'org.json', module: 'json'

}

## 5、权限及配置

<uses-permission android:name="android.permission.CAMERA" />

<uses-permission android:name="android.permission.WRITE\_EXTERNAL\_STORAGE" />

<uses-permission android:name="android.permission.RECORD\_AUDIO" />

<uses-permission android:name="android.permission.RECORD\_VIDEO" />

<uses-permission android:name="android.permission.INTERNET" />

<uses-permission android:name="android.permission.WRITE\_SETTINGS" />

<uses-permission android:name="android.permission.ACCESS\_WIFI\_STATE" />

<uses-permission android:name="android.permission.ACCESS\_NETWORK\_STATE" />

<uses-permission android:name="android.permission.FLASHLIGHT" />

<uses-permission android:name="android.permission.CHANGE\_WIFI\_MULTICAST\_STATE" />

<uses-feature android:name="android.hardware.camera" />

<uses-feature android:name="android.hardware.camera.autofocus" />

# 五、流媒体快速接入介绍

## 1、初始化配置

### (1) 初始化VhallSDK

在用户调用VhallSDK中的任意方法前，一定要先调用init方法，初始化VhallSDK。

*/\*\**

*\* VhallSDK初始化*

*\* @param Context*

*\* @param APP\_KEY 权限申请时获得*

*\* @param APP\_SECRET\_KEY 权限申请时获得*

*\*/*

VhallSDK.init(this , APP\_KEY , APP\_SECRET\_KEY);

获取App\_Key、App\_Secret\_Key —> <http://e.vhall.com/home/vhallapi/authlist>

### (2)基础参数说明

|  |  |
| --- | --- |
| 参数 | 描述 |
| id | 对应创建的活动ID(在官网创建) |
| token | 对应创建的访问Token (测试Token的实效是一天) |
| 码率 | 默认300 |
| 帧率 | 默认10帧 可选范围为10~30帧 超过30帧的按30帧算 |
| 初次缓冲时间 | 只用在观看直播, 默认为2秒  (这里的缓冲时间不是用于延迟播放,而是缓冲2秒的数据) |
| 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 |
| 分辨率 | 640\*480/1280\*720 |
| APP\_KEY | 权限申请时获得 |
| APP\_SECRET\_KEY | 权限申请时获得 |
| 包名 | 第三方用户App包名 |
| 签名 | 第三方用户App签名的SHA1值 |

备注 ： 当连接失败 SDK默认重新连接3次，每次重连时间约为5秒

## 2、用户标识

### (1) 创建用户

API地址 ： <http://e.vhall.com/home/vhallapi/active#user_register_第三方创建用户>

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，请参数请参照API地址，

### (2) 登陆

当用户在vhall平台创建用户标识成功之后，调用VhallSDK中的login方法，如果用户需要使用如聊天，问答等功能则必须用户标识。如果不用户标识则默认是游客模式 (Demo里即使是游客也是可以聊天的，用户可以根据自己的场景控制。问答必须创建用户)

以下是代码展示：

VhallSDK.getInstance().login(username, userpass, new VhallSDK.LoginResponseParamCallback() {

@Override

public void success(String vhall\_id, String customer\_id) {vhall\_id}

@Override

public void failed(int errorCode, String reason) {}

});

### (3)登陆参数描述

|  |  |
| --- | --- |
| 参数字段 | 描述 |
| username | 用户名 |
| userpass | 用户密码 |
| vhallSDK.RequestCallback() | 回调信息 |

返回参数描述

|  |  |
| --- | --- |
| 参数字段 | 描述 |
| vhall\_id | vhall平台生成的ID 后续看直播会用到 |
| customer\_id | 用户平台生成的ID |

错误码

|  |  |
| --- | --- |
| 错误码 | 描述 |
| 10501 | 用户不存在 |
| 10502 | 登陆密码不正确 |

## 3、发直播

### (1) 准备工作：

屏幕保持常亮。

getWindow().setFlags(WindowManager.LayoutParams.FLAG\_KEEP\_SCREEN\_ON,

WindowManager.LayoutParams.FLAG\_KEEP\_SCREEN\_ON);

横竖屏发起视频

*/\*\**

*\* 如果竖屏发起设置ActivityInfo.SCREEN\_ORIENTATION\_REVERSE\_PORTRAIT*

*\* 如果横屏发起设置ActivityInfo.SCREEN\_ORIENTATION\_REVERSE\_LANDSCAPE*

*\*/*

setRequestedOrientation(ActivityInfo.SCREEN\_ORIENTATION\_REVERSE\_PORTRAIT);

设置发起布局

<com.vhall.business.VhallCameraView

android:id="@+id/cameraview"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent" />

VhallCameraView

Activity 被创建，首先初始化 VhallCameraView . 当前自定义View会处理包括采集,自动聚焦等关于Camera的操作 , VhallCameraView 需要初始化获得一些信息，调用 init()方法

*/\*\**

*\* pixel\_type 发起的分辨率*

*\*/*

getCameraView().init(pixel\_type, Activity(), new RelativeLayout.LayoutParams(0, 0));

### (2) 发直播:

Broadcast实例：

Broadcast实例 这里需要将之前设置的信息传入Broadcast中 列如自定义view、帧率、码率 、发起事件回调、聊天，此处完整代码可以参考Demo

Broadcast.Builder builder = new Broadcast.Builder()

.cameraView(mView.getCameraView()).frameRate(param.frameRate)

.chatCallback(new ChatCallback()) *//如需要使用聊天 加上这个回调*

.videoBitrate(param.videoBitrate)

.callback(new BroadcastEventCallback()); // *直播事件回调*

broadcast = builder.build();

一键发起直播，调用SDK initBroadcast方法，在这之前要先初始化观看实例。

发起参数描述：

|  |  |
| --- | --- |
| 参数字段 | 描述 |
| id | 活动ID |
| accessToken | 访问token |
| vhallID | 是否使用子账号发直播（新加） |
| Broadcast | 发起实例 |
| RequestCallback | 回调信息 |

备注： 子账号需要先创建，创建后会获取vhallID，当vhallID这个参数不为空时，使用子账号发起直播，使用的Token也需要用子账号重新生成，否则会返回身份验证失败。当vhallID这个参数为空时，默认使用主账号。

以下是代码展示

VhallSDK.getInstance().initBroadcast(param.id, param.token, getBroadcast(),

new VhallSDK.RequestCallback(){

@Override

public void success() {} // 发起成功

@Override

public void failed(int errorCode, String reason) {}

});

### (3) 直播事件回调

private class BroadcastEventCallback implements Broadcast.BroadcastEventCallback {

@Override

public void onError(int errorCode, String reason) {}

@Override

public void onStateChanged(int stateCode) {

switch (stateCode) {

case Broadcast.STATE\_CONNECTED: */\*\* 连接成功\*/*

break;

case Broadcast.STATE\_NETWORK\_OK: */\*\* 网络通畅\*/*

break;

case Broadcast.STATE\_NETWORK\_EXCEPTION: */\*\* 网络异常\*/*

break;

case Broadcast.STATE\_STOP:*/\*\* 直播停止\*/*

break;

}

}

@Override

public void uploadSpeed(String kbps) {*/\*\* 下载速度\*/*}

}

状态码

|  |  |  |
| --- | --- | --- |
| 状态码 | 描述 | Broadcast常量 |
| 20151 | 连接成功 | Broadcast.STATE\_CONNECTED |
| 20152 | 网络通畅 | Broadcast.STATE\_NETWORK\_OK |
| 20153 | 网络异常 | Broadcast.STATE\_NETWORK\_EXCEPTION |
| 20154 | 直播停止 | Broadcast.STATE\_STOP |

错误码

|  |  |
| --- | --- |
| 错误码 | 描述 |
| 10401 | 活动结束失败 |
| 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 |
| 10411 | 用户套餐余额不足 |
| 20101 | 正在直播 |
| 20102 | 初始化视频信息失败 |
| 20103 | 预览失败,无法直播 |
| 20104 | 直播地址有误 |
| 20105 | 连接服务器失败 |

### (4) 结束直播

获取VhallSDK的实例 调用finishBroadcast() 传入参数活动ID、TOKEN、Broadcast实例、结束回调 当直播结束时，需要调用此方法，此方法用于结束直播，生成回放，如果不调用，则无法生成回放。

参数说明：

|  |  |
| --- | --- |
| 参数字段 | 描述 |
| id | 活动ID |
| accessToken | 访问token |
| Broadcast | 发起实例 |
| RequestCallback | 回调信息 |

以下是代码展示

VhallSDK.getInstance().finishBroadcast(param.id, param.token, getBroadcast(), new VhallSDK.RequestCallback() {

@Override

public void success() {*// 停止成功}*

@Override

public void failed(int errorCode, String reason) {*// 停止失败}*

});

### (5)相关功能接口

getBroadcast.changeCamera()切换摄像头；

getBroadcast.setMute(Boolean mute)设置是否静音；

getBroadcast. setVolumeAmplificateSize(float size)设置降噪级别[0-1]

cameraview.setFilter(GPUImageFilter filter)设置滤镜

## 4、看直播

### (1) 看直播

WatchLive 实例：

watchLive 实例 , 这里需要将一些设置信息传入 列如Context、containerLayout(这里需要传入一个RelativeLayout,用于生成观看)、回调callback, MessageEventCallback 消息回调 ，ChatCallback 聊天回调

WatchLive.Builder builder = new WatchLive.Builder()

.context()

.containerLayout() // 传入观看布局

.bufferDelay() // 缓冲几秒的BUFFER

.callback(new WatchCallback())

.messageCallback(new MessageEventCallback())

.chatCallback(new ChatCallback());//*如果使用聊天就加这个回调*

watchLive = builder.build();

一键观看直播：

一键观看直播，当Activity 被创建 观看界面Activity必须包涵一个RelativeLayout布局 此布局需要往VhallSDK中传递 用于一键生成回放，获取VhallSDK的实例 调用initWatch() 这里传入参数WatchLive实例、活动ID(必填)、用户名、用户邮箱、vhall\_id、K值校验等参数。

参数描述

|  |  |
| --- | --- |
| 参数字段 | 描述 |
| id | 活动ID |
| nickname | 用户名 |
| email | 用户邮箱 |
| vhall\_id | VhallId (登陆后获取,没有传空) |
| password | 密码（K值） |
| ~~recordId~~ | ~~回放片段ID（只在观看回放使用），这里传空~~ |
| WatchLive | 观看直播实例 |
| RequestCallback | 回调信息 |

备注：如果用户名和密码为空，则vhall\_id 不能为空，

如果vhall\_id为空，则用户名和密码不能为空，

如果都传，默认取vhall\_id的值。

以下是代码展示 详细见Demo

VhallSDK.getInstance().initWatch(param.id, "test", "test@vhall.com", vhallId , param.k, getWatchLive(),

new VhallSDK.RequestCallback() {

@Override

public void success() {*// 获取观看信息成功}*

@Override

public void failed(int errorCode, String reason) { 失败}

});

### (2) 观看事件回调

WatchCallback 观看回调

private class WatchCallback implements WatchLive.WatchEventCallback {

@Override

public void onError(int errorCode, String errorMsg) {*// 错误返回错误码*}

@Override

public void onStateChanged(int stateCode) { *// 返回状态码*}

@Override

public void uploadSpeed(String kbps) { *// 速度*}

}

状态码

|  |  |  |
| --- | --- | --- |
| 状态码 | 描述 | WatchLive常量 |
| 20251 | 观看直播连接成功 | WatchLive. STATE\_CONNECTED |
| 20254 | 开始加载 | WatchLive. STATE\_BUFFER\_START |
| 20255 | 停止加载 | WatchLive. STATE\_BUFFER\_STOP |
| 20256 | 停止观看直播 | WatchLive. STATE\_STOP |

错误码

|  |  |
| --- | --- |
| 错误码 | 描述 |
| 10030 | 身份验证出错 |
| 10402 | 当前活动ID错误 |
| 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出错 |
| 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 |

### (3) 停止观看

当用户停止观看时，需要调用VhallSDK中停止观看直播方法，调用此方法，SDK会断开拉流。

代码展示如下

getWatchLive().stop();

### (4) 开启弹幕

弹幕实现依赖第三方库B站的弹幕引擎·烈焰弹幕 DanmakuFlameMaster

第一步，在我们视频view布局的上方再覆盖一个显示弹幕的View， 弹幕的View必须要做成完全透明的, 这样即使覆盖在视频界面的上方也不会影响到视频的正常观看，布局如下：

<master.flame.danmaku.ui.widget.DanmakuView

android:id="@+id/sv\_danmaku"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent" />

第二步，找到build.gradle添加如下依赖

dependencies {

compile 'com.github.ctiao:DanmakuFlameMaster:0.6.4'

compile 'com.github.ctiao:ndkbitmap-armv7a:0.6.4'

compile 'com.github.ctiao:ndkbitmap-armv5:0.6.4'

compile 'com.github.ctiao:ndkbitmap-x86:0.6.4'

}

第三步，初始化弹幕所需的信息，设置回调详情参考Demo，

HashMap<Integer, Integer> maxLinesPair = new HashMap<Integer, Integer>();

maxLinesPair.put(BaseDanmaku.TYPE\_SCROLL\_RL, 5); // 滚动弹幕最大显示5行

// 设置是否禁止重叠

HashMap<Integer, Boolean> overlappingEnablePair = new HashMap<Integer, Boolean>();

overlappingEnablePair.put(BaseDanmaku.TYPE\_SCROLL\_RL, true);

overlappingEnablePair.put(BaseDanmaku.TYPE\_FIX\_TOP, true);

备注： 具体实现方式参考Demo

### (5) VR活动

如果发起的直播时VR直播 , 可以使用陀螺仪功能

if (!getWatchLive().isVR()) { // 判断是否是VR活动

watchView.showToast("当前活动为非VR活动，不可使用陀螺仪");

return;

}

getWatchLive().setVRHeadTracker(!getWatchLive().isVRHeadTracker()); // 设置陀螺仪

### (6) 投屏功能

备注： 新增Support包 Suppot包中包含投屏功能,不需要的可以删掉, 此版本是投屏的实现,只有开始投屏功能,暂时没有结束,快进等功能

投屏是基于DLNA功能实现的, 使用的ClingDLNA类库 , 确保使用的机器是具备DLNA功能的手机 理论上支持API16以上的机型, 实际测试有部分API 16以上的机型不支持 .

DLNA 协议中包含多项协议及标准， 其中UPnP协议是最重要的部分

UPnP协议定义了设备之间，设备和控制点，控制点之间通信的协议

@Override

public void dlnaPost(DeviceDisplay deviceDisplay, AndroidUpnpService service) {

getWatchLive().dlnaPost(deviceDisplay, service, new Watch\_Support.DLNACallback() {

@Override

public void onError(int errorCode) {

watchView.showToast("投屏失败，errorCode:" + errorCode);

}

@Override

public void onSuccess() {

watchView.showToast("投屏成功!");

stopWatch();

}

});

}

## 5、看回放

### (1) 看回放

Watchplayback实例：

Watchplayback实例，和观看直播类似，传入Context , ContainerLayout(这里需要传入一个RelativeLayout,用于生成观看回放) ， callback获取观看回放时的一些状态。观看回放的操作和观看直播一样，请求的方法相同，参数相同。 代码可以参考上面的观看直播。

WatchPlayback.Builder builder = new WatchPlayback.Builder()

.context(playbackView.getmActivity())

.containerLayout(playbackView.getContainer())

.callback(new WatchCallback())

.docCallback(new DocCallback()); //新增 回放绘制PPT和白板

watchPlayback = builder.build();

一键观看回放：（参数和观看直播相同）

一键观看回放，参数和观看直播相同，传递的观看实例变成WatchedPlayBack,

|  |  |
| --- | --- |
| 参数字段 | 描述 |
| id | 活动ID |
| nickname | 用户名 |
| email | 用户邮箱 |
| vhall\_id | VhallId (登陆后获取,没有传空) 回放这里传空 |
| password | 密码（K值） |
| recordId | 回放片段ID（只在观看回放使用） |
| WatchPlayBack | 观看回放实例 |
| RequestCallback | 回调信息 |

VhallSDK.getInstance().initWatch(param.id, "test", "test@vhall.com", vhallId, param.k, , recordId getWatchPlayback(),

new VhallSDK.RequestCallback() {

@Override

public void success() {*// 获取观看信息成功}*

@Override

public void failed(int errorCode, String reason) { 失败}

});

### (2) 观看回放事件回调

VhallSDK.getInstance().initWatch(param.id, "test", "test@vhall.com", vhallId , param.k, recordId , getWatchPlayback(),

private class WatchCallback implements WatchPlayBack.WatchEventCallback {

@Override

public void onError(int errorCode, String errorMsg) {*// 错误返回错误码*}

@Override

public void onStateChanged(boolean playWhenReady, int playbackState) {*/*

switch (playbackState) {*/播放器过程中的状态信息*

case VhallHlsPlayer.STATE\_IDLE:*// 闲置状态*

break;

case VhallHlsPlayer.STATE\_PREPARING:*// 准备状态*

break;

case VhallHlsPlayer.STATE\_BUFFERING:*// 正在加载*

break;

case VhallHlsPlayer.STATE\_READY:*// 正在加载*

break;

case VhallHlsPlayer.STATE\_ENDED:*// 准备就绪*

break;

case VhallHlsPlayer.STATE\_ENDED:*// 结束*

default:

break;

}

}

@Override

public void uploadSpeed(String kbps) { *// 速度*}

@Override

public void onStartFailed(String errorMsg) {*// 初始化观看播放器时的错误*}

}

### (3) 播放器方法

当观看信息请求成功，虽然和观看直播请求的是相同的方法，但是逻辑处理不同，SDK会默认得到播放地址并设置进播放器中，用户只需调用watchPlayback实例中的各种方法来获取想要得到的信息。

开始播放：

getWatchPlayback().start();

暂停播放：

getWatchPlayback().pause();

停止播放：

getWatchPlayback().stop();

获取播放进度：

getWatchPlayback().seekTo(playerCurrentPosition);

获取当前播放进度：

getWatchPlayback().getCurrentPosition();

获取播放时长：

getWatchPlayback().getDuration();

是否正在播放

getWatchPlayback().isPlaying();

### (4) 发送/获取评论信息

发送参数

|  |  |
| --- | --- |
| 参数字段 | 描述 |
| text | 评论内容 |
| user\_id | 用户登陆返回的唯一标识 |
| RequestCallback | 回调信息 |

代码展示

getWatchPlayback().sendComment(text, user\_id, new VhallSDK.RequestCallback() {

@Override

public void success() { //接口请求成功

chatView.clearInputContent();

chatView.clearChatData();

}

  @Override

public void failed(int errorCode, String reason) {}

});

|  |  |
| --- | --- |
| 错误码 | 描述 |
| 10010 | 活动不存在 |
| 10011 | 不是该平台下的活动 |
| 10017 | 活动ID不能为空 |
| 10806 | 内容不能为空 |
| 10807 | 用户ID不能为空 |
| 10808 | 当前用户未参会 |

获取参数

|  |  |
| --- | --- |
| 参数字段 | 描述 |
| webinar\_id | 活动ID |
| limit | 每页的数量(取条目最多为50条) |
| pos | 偏移量(如果从0开始就是最新的一条) |
| ChatServer.ChatRecordCallback() | 回调信息 |

watchPlayback.requestCommentHistory(webinar\_id, String.valueOf(limit), String.valueOf(pos), new ChatServer.ChatRecordCallback() {

@Override

public void onDataLoaded(List<ChatServer.ChatInfo> list) {//接口请求成功

chatView.notifyDataChanged(list);

chatView.clearInputContent();

}

@Override

public void onFailed(int errorcode, String messaage) {

Log.e(TAG , "messaage--> " + messaage);

}

});

错误码

|  |  |
| --- | --- |
| 错误码 | 描述 |
| 10030 | 身份验证出错 |
| 10402 | 当前活动ID错误 |
| 10403 | 活动不属于自己 |
| 10407 | 查询数据为空 |
| 10412 | 直播中，获取失败 |
| 10413 | 获取条目最多为50条 |
| 10409 | 参会信息不存在 |
| 10410 | 活动开始时间不存在 |

备注: 如果出现相同的评论信息，请根据请求后返回的ID去重

### (5) 观看回放绘制PPT和白板

1 回调WatchPlayBack.DocumentEventCallback

private class DocCallback implements WatchPlayback.DocumentEventCallback {

@Override

public void onEvent(String key, List<MessageServer.MsgInfo> msgInfos) {

if (msgInfos != **null** && msgInfos.size() > 0) {

documentView.paintPPT(key, msgInfos);

documentView.paintBoard(key, msgInfos);

}

}

@Override

public void onEvent(MessageServer.MsgInfo msgInfo) {

documentView.paintPPT(msgInfo);

documentView.paintBoard(msgInfo);

}

}

备注： 在Builder观看回放实例的时候加入 docCallback()

## 6 滤镜直播:

使用滤镜发起直播, 需要使用新的CameraFilterView发起直播，目前提供美颜滤镜,

<!- 滤镜使用的布局文件->

<com.vhall.vhalllive.pushlive.CameraFilterView

android:id="@+id/cameraview"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent" />

方法实现 调用CameraFilterView 中实现滤镜的方法

第一步 创建美颜滤镜 VhallBeautyFilter

第二步 设置CameraFilterView

第三步 设置美颜等级

if (beautyFilter == null) {

beautyFilter = new VhallBeautyFilter();

cameraview.setFilter(beautyFilter);

<!- 设置美颜滤镜的等级 ->

beautyFilter.setBeautyLevel(level) （1~5）

}

<!- 自动判断当前机型硬件是否可以支持滤镜 ->

cameraFilterView.setAutoCloseFilterCallback(new GPUImageRenderer.AutoCloseBaeutyFilter() {

@Override

public void onAutoCloseBaeutyFilter() {

}

});

<!- @param int adjuster >

cameraFilterView .setFilterAdjuster(int adjuster)

# 六、功能接入介绍

## 1、聊天服务器相关功能

当用户在创建发起直播实例或者观看直播实例时的.ChatCallback中传入chatCallback回调，聊天服务器就已经开启了，具体参考快速接入介绍中的发起观看创建实例的描述

以下是聊天服务器回调

private class ChatCallback implements ChatServer.Callback {

@Override

public void onChatServerConnected() {}// 聊天服务器建立

@Override

public void onConnectFailed() {} // 聊天服务器连接失败

@Override

public void onChatMessageReceived(ChatServer.ChatInfo chatInfo) {//消息接收

switch (chatInfo.event) {

case ChatServer.eventMsgKey: // 聊天消息通知

break;

case ChatServer.eventOnlineKey: // 上线消息通知

break;

case ChatServer.eventOfflineKey: // 下线消息通知

break;

case ChatServer.eventQuestion: // 问答消息

break;

}

}

@Override

public void onChatServerClosed() {}// 聊天服务器关闭

}

公共字段

|  |  |
| --- | --- |
| 字段 | 描述 |
| account\_id | 用户ID |
| user\_name | 用户昵称 |
| avater | 用户头像 |
| room | 活动id |
| event | 消息类型 (用于区分消息用途) |
| time | 发送时间 |

聊天服务器其他方法

连接聊天服务器：

getBroadcast().connectChatServer ();

关闭聊天服务器：

getBroadcast().disconnectChatServer ();

### (1)、上下线消息通知

使用此功能默认聊天服务器已开启

消息体说明：chatInfo.event = “online offline”

|  |  |
| --- | --- |
| OnlineData | 描述 |
| role | 用户类型 host:主持人 guest：嘉宾 assistant：助手 user：观众 |
| concurrent\_user | 房间内当前用户数 |
| is\_gag | 是否被禁言 |
| attend\_count | 参会人数 |

### (2)、聊天消息

使用此功能默认聊天服务器已开启

目前发起直播和观看直播中可以聊天。发起直播调用getBroadCast().sendChat()方法 。观看直播调用getWatchLive().sendChat()。

参数说明：

|  |  |
| --- | --- |
| 参数字段 | 描述 |
| text | 聊天内容 |
| VhallSDK.RequestCallback | 回调信息 |

发起直播代码展示 && 观看直播代码展示：

getBroadcast().sendChat(text, new VhallSDK.RequestCallback() { // 发起直播时聊天

@Override

public void success() {}

@Override

public void failed(int errorCode, String reason) {}

});

getWatchLive().sendChat(text, new VhallSDK.RequestCallback() {// 观看直播时聊天

@Override

public void success() {}

@Override

public void failed(int errorCode, String reason) {}

});

### (3)、自定义消息

使用此功能默认聊天服务器已开启

消息体说明：chatInfo.event = “broadcast”

|  |  |
| --- | --- |
| ChatData | 描述 |
| text | 聊天内容 |

getWatchLive().sendCustomt(text, new VhallSDK.RequestCallback() {

@Override

public void success() {}

@Override

public void failed(int errorCode, String reason) {}

});

### (4)、聊天记录

获取SDK聊天记录。聊天记录只在观看直播时候获取，调用acquireChatRecord()

参数说明：

|  |  |
| --- | --- |
| 参数 | 描述 |
| showAll | 显示当次直播聊天最多为20条,true显示所有聊天最条为20条 |
| ChatRecordCallback | 聊天记录回调 |

代码展示：

getWatchLive().acquireChatRecord(false, new ChatServer.ChatRecordCallback() {

@Override

public void onDataLoaded(List<ChatServer.ChatInfo> list) {}

@Override

public void onFailed(int errorcode, String messaage) {}

});

错误码

|  |  |
| --- | --- |
| 错误码 | 描述 |
| 10030 | 身份验证出错 |
| 10402 | 当前活动ID错误 |
| 10403 | 活动不属于自己 |
| 10407 | 查询数据为空 |
| 10408 | 当前活动非直播状态 |
| 10409 | 参会信息不存在 |
| 10410 | 活动开始时间不存在 |

### (5)、问答消息

发送活动问答

目前只支持观看端发送活动问答。在用户登陆成功的情况下可以发送问答，问答每5分钟发送一次，避免一些用户恶意发送。调用sendQues()方法，发送问答信息。

代码展示

getWatchLive().sendQuestion (text, new VhallSDK.RequestCallback() {

@Override

public void success() { *// 发送成功 }*

@Override

public void failed(int errorCode, String reason) {*// 发送失败}*

});

消息体说明：chatInfo.event = “question”

|  |  |
| --- | --- |
| QuestionData | 描述 |
| id | 问题的id |
| nick\_name | 昵称 |
| content | 提问内容 |
| join\_id | 参会ID |
| created\_at | 创建时间 |
| role\_name | 角色 |
| is\_open | 是否为私密回答 |
| QuestionData | answer 参数和此消息体相同 |

错误码

|  |  |
| --- | --- |
| 错误码 | 描述 |
| 10601 | 不是直播 |
| 10602 | 参会者不存在 |
| 10603 | 5分钟内不可提问 |
| 10604 | 活动ID不能为空 |
| 10605 | 问题不能为空 |
| 10606 | 用户不能为空 |

## 2、消息服务器相关功能

消息服务器，目前只用于观看端使用，用于接收观看直播时的一些消息处理

private class MessageEventCallback implements MessageServer.Callback {

@Override

public void onEvent(MessageServer.MsgInfo messageInfo) {

switch (messageInfo.event) {

case MessageServer.EVENT\_PPT\_CHANGED:*//PPT翻页消息*

documentView.showDoc(messageInfo.pptUrl);

break;

case MessageServer.EVENT\_DISABLE\_CHAT:*//禁言 暂不可用*

break;

case MessageServer.EVENT\_KICKOUT:*//踢出 暂不可用*

break;

case MessageServer.EVENT\_OVER:*//直播结束*

break;

case MessageServer.EVENT\_PERMIT\_CHAT:*//解除禁言 暂不可用*

break;

case MessageServer.EVENT\_START\_LOTTERY:*//抽奖开始*

break;

case MessageServer.EVENT\_END\_LOTTERY:*//抽奖结束*

break;

case MessageServer.EVENT\_NOTICE: *公告消息*

break;

case MessageServer.EVENT\_SIGNIN: *//签到消息*

break;

case MessageServer.EVENT\_QUESTION: *// 问答开关*

break;

}

}

@Override

public void onConnectFailed() {}

@Override

public void onMsgServerClosed() {}

}

公共参数：

|  |  |
| --- | --- |
| 字段 | 描述 |
| event | 消息类型 (用于区分消息用途) |

连接消息服务器：

getWatchLive().connectMsgServer ();

关闭消息服务器：

getWatchLive().disconnectMsgServer ();

### (1)、活动结束消息

msgInfo.event = “MessageServer.EVENT\_OVER”

### (2)、抽奖消息

观看直播时播主可以发起抽奖，而在移动端观看时可以接受抽奖消息，目前默认游客和参会用户都可以抽奖，而且只能被中奖一次

开始抽奖消息体msgInfo.event = “MessageServer. EVENT\_START\_LOTTERY”

|  |  |
| --- | --- |
| 字段 | 描述 |
| Num | 可以中奖的数量 |
| type | 消息类型键 |

结束抽奖消息体msgInfo.event = “MessageServer. EVENT\_END\_LOTTERY”

|  |  |
| --- | --- |
| 字段 | 描述 |
| type | 消息类型键 |
| Msginfo.Lists | 返回中奖的数组 |
| IsLottery | 是否中奖 |
| id | 当前中奖人的参会ID |
| LotteryID | 抽奖ID |

Msginfo.Lists 结构体

|  |  |
| --- | --- |
| 字段 | 描述 |
| nickname | 中奖昵称 |
| user\_id | 用户ID |
| third\_user\_id | 登陆用户名 |

当用户中奖后，需要提交信息，访问SDK中的submitLotteryInfo() , 所需参数如下

字段描述

|  |  |
| --- | --- |
| 字段 | 描述 |
| id | 当前中奖人的参会ID |
| LotteryID | 抽奖ID |
| name | 中奖人姓名 |
| phone | 中奖人电话 |

SDK代码展示

VhallSDK.getInstance().submitLotteryInfo(joid\_id, lottery\_id, nickname, phone,

new VhallSDK.RequestCallback() {

@Override

public void success() {}

@Override

public void failed(int errorCode, String reason) {}

});

}

错误码

|  |  |
| --- | --- |
| 字段 | 描述 |
| 10030 | 身份验证失败 |
| 10409 | 参会ID不能为空 |
| 10410 | 抽奖ID不能为空 |
| 10411 | 用户名称不能为空 |
| 10412 | 用户手机不能为空 |

### (3)、公告消息

msgInfo.event = “MessageServer.EVENT\_NOTICE”

当主播正在直播，可以发送公告，公告发送后所有观看的用户都可以接受到消息，Demo中用跑马灯展示

|  |  |
| --- | --- |
| 字段 | 描述 |
| content | 公告内容 |
| publish\_release\_time | 公告发布时间 |

### (4)、签到消息

msgInfo.event = “MessageServer.EVENT\_SIGNIN”

当主播正在直播，可以发布签到，观看用户必须登陆才可以签到，签到后web端会进行展示

|  |  |
| --- | --- |
| 字段 | 描述 |
| webinar\_id | 活动ID |
| sign\_id | 签到ID |
| user\_id | 用户ID |
| sign\_show\_time | 签到初始时间 |

### (5)、问卷消息

msgInfo.event = “MessageServer. EVENT\_SURVEY”

当主播正在直播，可以发送问卷,消息服务器会接收到survey\_id ,需要使用这个survey\_id 获取问卷信息

VhallSDK.getInstance().getSurveyInfo(surveyid, new SurveyDataSource.SurveyInfoCallback() {

@Override

public void onSuccess(Survey survey) {

watchView.showSurvey(survey);

}

@Override

public void onError(int errorCode, String errorMsg) {

chatView.showToast(errorMsg);

}

});

|  |  |
| --- | --- |
| 字段 | 描述 |
| survey\_id | 问卷ID |

### (6)、白板消息

msgInfo.event = “MessageServer.EVENT\_SHOWBOARD” // 开启或者关闭白板

msgInfo.event = “MessageServer.EVENT\_INITBOARD” // 初始化白板

msgInfo.event = “MessageServer.EVENT\_PAINTBOARD” // 绘制白板

msgInfo.event = “MessageServer.EVENT\_DELETEBOARD // 删除白板内容

msgInfo.event = “MessageServer.EVENT\_CLEARBOARD” // 清空白板

1. 配置白板XML (在需要显示的界面中加入， 具体详见文档)

<com.vhall.business.widget.WhiteBoardView

android:id="@+id/board"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"/>

2、 获取到白板的实例调用setStep方法 ， 将消息服务器获取的信息传递进方法里就可以了

public void paintBoard(MessageServer.MsgInfo msgInfo) {

board.setStep(msgInfo);

}

public void paintBoard(String key, List<MessageServer.MsgInfo> msgInfos) {

board.setSteps(key, msgInfos);

}

### (7)、PPT消息（修改）

msgInfo.event = “MessageServer.EVENT\_CHANGEDOC” // 切换文档页面

msgInfo.event = “MessageServer.EVENT\_PAINTDOC” // 绘制文档信息

msgInfo.event = “MessageServer.EVENT\_DELETEDOC” // 删除绘制信息

msgInfo.event = “MessageServer.EVENT\_CLEARDOC” // 清空绘制信息

1 配置文档XML (在需要显示的界面中加入， 具体详见文档)

<com.vhall.business.widget.PPTView

android:id="@+id/iv\_doc"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent" />

2 获取到文档的实例调用setStep方法 ， 将消息服务器获取的信息传递进方法里就可以了

@Override

public void paintPPT(MessageServer.MsgInfo msgInfo) {

doc.setStep(msgInfo);

}

@Override

public void paintPPT(String key, List<MessageServer.MsgInfo> msgInfos) {

doc.setSteps(key, msgInfos);

}

## 3、分辨率切换/切换到单音频

1. 目前定义的分辨率有

public static final int DPI\_DEFAULT= 0; *// 默认*

public static final int DPI\_SD= 1; *// 标清*

public static final int DPI\_HD= 2; *// 高清*

public static final int DPI\_UHD= 3; *// 超高清*

public static final int DPI\_AUDIO= 4; *// 纯音频*

1. 功能实现 ：只需要将定义好的常量传到watchLive实例中 ,调用setDefinition(pixel) ,当停止直播之后，不能立即重连，需要延迟1秒

getWatchLive().setDefinition(level)

1. 获取分辨率是否可用 返回一个map 状态为 0 不可用 ： 1 可用

getWatchLive().getDefinitionAvailable();

## 4、设置观看布局

1. 目前观看端WatchLive定义了5种适配类型，用户可以根据自己的场景去设定

|  |  |
| --- | --- |
| **观看类型** | **描述** |
| FIT\_DEFAULT | 默认 自适应 |
| FIT\_CENTER\_INSIDE | 视频的原始尺寸居中显示 |
| FIT\_X | 拉伸X轴，等比例放大(适合PC端发起) |
| FIT\_Y | 拉伸Y轴，等比例放大(适合移动端发起) |
| FIT\_XY | 拉伸XY轴 (会拉伸，也会全屏) |

1. 设置方法

*//scaleType 观看类型*

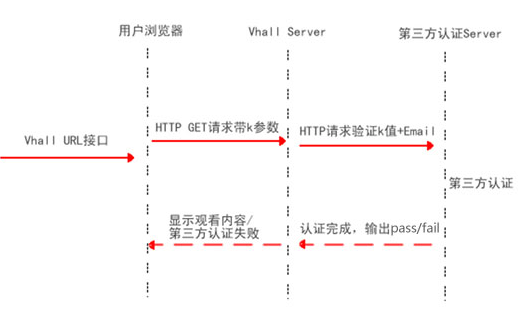
getWatchLive().setScaleType(scaleType);

**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**



* 1. **开启设置**

1. 第三方回调接口设置

* 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](http://e.vhall.com/home/vhallapi/active#webinar_whole-auth-url_全局配置第三方k值验证url)
* 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
* 针对某个活动的配置方式二：通过接口(webinar/create或webinar/update)设置
* 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。

1. 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. 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。**

1. 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'
   1. **K值使用**
2. 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。

具体查看上面的“观看直播”里的参数说明。

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