

## 微吼直播 SDK for Android



移动互联第一影响力  
中国最大的视频互动直播平台

一、 修订记录 .....	2
二、 简介 .....	2
三、 权限开通申请 .....	3
四、 SDK 使用准备 .....	3
五、 快速接入介绍 .....	5

## 一、 修订记录

日期	版本号	描述	修订者
2016.5.6	v2.2.0	1.新增文档演示 2.优化观看体验	
2016.5.13	v2.2.1	新增帧率配置	
2016.6.3	v2.2.2	优化横竖屏切换后的全屏观看	
2016.7.25	v2.3.1	多分辨率切换	

## 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”，帮助读者进行快速开发，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。

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 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	使用软编，码率固定在 300-400 之间
	支持环境	Android 4.0 以上，
观看直播	支持播放协议	RTMP/HLS

	延时	RTMP: 2-4 秒 , HLS : 20 秒左右
	支持解码	H.264
文档演示 ( new )	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	<a href="http://e.vhall.com/home/vhallapi/embed">http://e.vhall.com/home/vhallapi/embed</a> 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

### 三、权限开通申请

请点击 [API&SDK 权限申请](#) 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、Secret\_Key、App Secret\_Key (备注：必须获取权限,否则无法使用)

### 四、SDK 使用准备

#### 1、下载 SDK&DEMO

Demo 采用 MVP 模式

github 地址：[https://github.com/vhall20/vhallsdk\\_live\\_android\\_studio](https://github.com/vhall20/vhallsdk_live_android_studio)

#### 2、开发环境要求

Pc 操作系统：64window 系统

JDK: 1.7 以上

开发工具：Android studio 2.0

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、需要导入的 Jar

Vhallsdk.jar

vhallbusiness.jar

#### 4、动态库 SO

Libdynload.so

Libffmpeg.so

Libjingle.so

libstlport\_shared.so

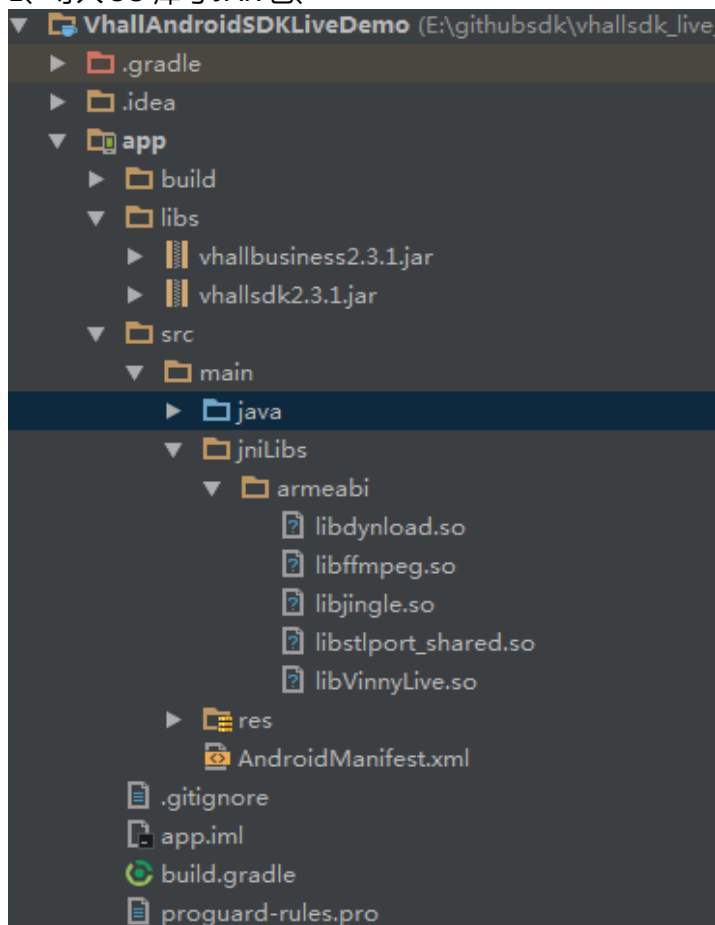
libVinnyLive.so

## 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" />
```

## 6、将 SDK 添加到工程


### 1、导入 SO 库与 JAR 包、



## 7、添加依赖

Glide 是 Demo 中用于显示 PPT 的，socket.io 是用于 SDK 网络连接，以下的 2 个依赖必须添加，否则 Demo 不能运行。

```
compile 'com.android.support:support-v4:23.3.0'
compile 'com.github.bumptech.glide:glide:3.7.0'
compile 'com.github.nkzawa:socket.io-client:0.6.0'
```



## 8、Eclipse 集成

鉴于国内还有许多的用户使用 Eclipse 开发，我们提供用户的使用都是以 JAR 包的方式，所以可以从官网上下载 JAR 在集成进自己的 Eclipse 开发环境。JAR 内的方法使用可以参考说明文档。（官方已经集成 Android Studio）

- Doc
- VhallAndroidSDKLiveDemo
- VhallAndroidSDKLiveDemo\_Eclipse
- VhallSDK

VhallAndroidSDKLiveDemo\_Eclipse 版是 2.3.1 使用 Eclipse 集成，为方便客户使用 Eclipse 开发。此版本之后不会继续维护。

## 五、快速接入介绍

### 1、权限认证信息配置

以下信息配置到文件里。文件名称：com.vhall.live.Constants.java

```
public class Constants {
    public static final String APP_KEY = ""; // 获取到的 App_Key
    public static final String APP_SECRET_KEY = ""; // 获取到的 APP_SECRET_KEY
}
```

在调用 VhallSDK 的方法前，一定要先调用 init 方法。

```
/**
 * VhallSDK init (再调用任何SDK方法之前先调用init方法)
 */
VhallSDK.init(Constants.APP_KEY, Constants.APP_SECRET_KEY);
```

## 2、基础参数说明

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

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 : 密码、F 码、活动如果有 K 值需要传

分辨率 : 选择当前发起的分辨率

备注 : 当连接失败 SDK 默认重新连接一次 重连时间约为 5 秒

## 3、APP 开始发起 -broadcast

### 1 在发起直播之前需要处理做一些操作

保持屏幕常亮

```
/**
 * 保持屏幕常亮 (必须设置)
 */
getWindow().setFlags(WindowManager.LayoutParams.FLAG_KEEP_SCREEN_ON,
    WindowManager.LayoutParams.FLAG_KEEP_SCREEN_ON);
```

横竖屏发起视频

```
/**
 * 设置横竖屏发起 如果是竖屏发起设置ActivityInfo.SCREEN_ORIENTATION_PORTRAIT
 * 如果是横屏发起设置ActivityInfo.SCREEN_ORIENTATION_LANDSCAPE
 */
this.setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_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"/>
```

2 当 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

```
/**
 * @param Pixex_Type 发起分辨率
 * VhallCameraView.TYPE_HDPI 640 * 480
 * VhallCameraView.TYPE_XHDPI 720 * 1280
 * @param Activity activity
 * @param LayoutParams layoutparam;
 */
cameraview.init(VhallCameraView.TYPE_HDPI, this, new RelativeLayout.LayoutParams(0, 0));
```

2 获取 VhallSDK 的实例 调用 initBroadcast() 这里传入参数活动 ID、TOKEN、Broadcast 实例、发起回调。此方法获取一次视频信息，返回成功则可以发起直播。

以下是代码展示

```
/**
 * 获取直播相关信息
 * @param id 活动ID
 * @param token 访问TOKEN
 * @param Broadcast broadcast实例
 * @param VhallSDK.InitBroadcastCallback 发起回调
 */
VhallSDK.getInstance().initBroadcast(param.id, param.token, getBroadcast(),
    new VhallSDK.InitBroadcastCallback() {
        @Override
        public void initBroadcastSuccess() {
            startBroadcast();
        }

        @Override
        public void initBroadcastFailed(String reason) {
        }
    });
```

3 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

```
Broadcast.Builder builder = new Broadcast.Builder()
    .cameraView(cameraview)
    .frameRate(frameRate)
    .videoBitrate(CodeRate)
    .callback(new Broadcast.BroadcastEventCallback() {
        @Override
        public void startFailed(String reason) {/** 发起失败 resaon失败原因*/}
        @Override
        public void onConnectSuccess() {/** 连接成功*/}
        @Override
        public void onErrorConnect() {/** 连接失败*/}
        @Override
        public void onErrorParam() {/** 直播参数错误*/}
        @Override
        public void onErrorSendData() {/** 数据传输失败*/}
        @Override
        public void uploadSpeed(String kbps) {/** kbps 返回每秒的速度*/}
        @Override
        public void onNetworkWeak() {/** 网络环境差*/}
        @Override
        public void onNetworkFluency() {/** 网络通畅*/}
        @Override
        public void onStop() {/** 停止*/}
    });
broadcast = builder.build();
```

#### 4、APP 停止发起 -broadcast

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，无法生成回放。

以下是代码展示：

```
/**
 * 结束直播，生成回放
 * @param id 活动ID
 * @param token 访问TOKEN
 * @param Broadcast broadcast实例
 * @param FinishBroadcastCallback 结束回调
 */
VhallSDK.getInstance().finishBroadcast(param.id, param.token, getBroadcast(),
    new VhallSDK.FinishBroadcastCallback() {
        @Override
        public void finishSuccess() { }

        @Override
        public void finishFailed(String reason) {}
    });
```

#### 5、APP 开始观看 -watch

1 调用 VhallSDK .initWatch() 这里传入参数活动 ID(必填)、用户名(必填)、用户邮箱(必填)、K 值校验、WatchLive 实例、观看回调。当回调返回成功说明播放信息已经获取，如果没有成功，会返回错误信息。

以下是代码展示：

```
/**
 * 开始观看直播
 * @param id 活动ID
 * @param name 姓名
 * @param email 邮箱
 * @param password K值
 * @param WatchLive 观看实例
 * @param VhallPPT vhallPPT展示
 * @param VhallSDK.InitWatchCallback 观看回调
 */
VhallSDK.getInstance().initWatch(params.id, "test", "test@vhall.com",
    params.k, getWatchLive(), new VhallSDK.InitWatchCallback() {
        @Override
        public void initSuccess() {}

        @Override
        public void initFailed(String msg) {}
    });
```



3 获取 watchLive 实例，这里需要将一些设置信息传入。如 context、加载秒数、视频容器(只支持 relativeLayout)、回调 callback。BufferDelay 并不是延迟播放，而是第一次加载几秒的数据，最高 10 秒的数据。

```
/**
 * 获取WatchLive实例
 */
WatchLive.Builder builder = new WatchLive.Builder().
    context(liveView.getmActivity()).
    containerLayout(liveView.getWatchLayout())./** 传入相对布局*/
    bufferDelay(params.bufferSecond). /** 初次加载几秒的数据*/
    callback(new WatchLive.WatchEventCallback() {
        @Override
        public void watchFailed(String reason) { /** 观看失败*/}
        @Override
        public void watchConnectSuccess() { /** 连接成功*/}
        @Override
        public void watchConnectFailed(String reason) {/** 连接失败*/}
        @Override
        public void watchLoadingSpeed(String kbps) {/** 下载速度*/}
        @Override
        public void watchStartBuffering() {/** 开始缓冲*/}
        @Override
        public void watchStopBuffering() {/** 停止缓冲*/}
    });
watchLive = builder.build();
```

#### 4 观看端多分辨率切换功能

当直播发起成功，用户可以调用此方法获得当前可以切换的分辨率，返回 HashMap

返回 0 为 无效不可用 ： 1 为 有效可用

```
/**
 * 获取当前视频可以切换的分辨率
 */
getWatchLive().getDPIStatus();
```

观看为默认播放线路 当线路返回有效时，用户可以进行切换操作

```
/** 默认播放线路*/
public static final int DPI_DEFAULT = 0;
/** 标清线路*/
public static final int DPI_SD = 1;
/** 高清线路*/
public static final int DPI_HD = 2;
/** 超高清线路*/
public static final int DPI_UHD = 3;
```

当切换分辨率的时候必须要先停止直播，当直播停止时不能立即重连，需要延时 1 秒，否则会报错。

```
/**
 * 设置分辨率
 */
getWatchLive().setDefinition(level);
```

## 5 观看端展示方案

目前定义了 5 种方案供用户选择，用户可以根据自己的实际情况进行选择

```
/** 默认方案 根据视频宽高控制 居中自适应 会留有黑边 但视频不会拉伸*/
public static final int FIT_DEFAULT = 0x00;
/** 视频原始尺寸居中显示*/
public static final int FIT_CENTER_INSIDE = 0x01;
/** 拉伸x轴 等比例放大 (适合PC发起)*/
public static final int FIT_X = 0x02;
/** 拉伸x轴 等比例放大 (适合移动端发起)*/
public static final int FIT_Y = 0x03;
/** 拉伸xy轴 等比例放大 会拉伸 也会全屏*/
public static final int FIT_XY = 0x04;
```

设置展示方案：

```
/**
 * 设置展示类型
 */
getWatchLive().setScaleType(scaleType);
```

## 5 观看端 PPT 播放

在直播的时候有时会伴随着 PPT 播放 如果想用手机观看 PPT 通过 VhallPPT 就可以实现 实现过程如下：

### 1) 创建 VhallPPT 对象

```
if (vhallPPT == null){
    vhallPPT = new VhallPPT();
}
```

### 2) 设置回调信息 返回当前文档的地址

```

/**
 * 设置PPT回调
 */
VhallPPT.setCallback(new VhallPPT.PPTChangeCallback() {
    @Override
    public void onPPTChange(String url) {

    }
});
getWatchLive().setVhallPPT(vhallPPT);

```

## 6、APP 结束观看

当用户停止观看时，需要调用 VhallSDK 中停止观看直播方法 当用户调用此方法，SDK 会断开拉流。代码展示如下：

```

/**
 * 停止拉流
 */
getWatchLive().stop();

```

## 7、APP 回放

当 App 发起直播结束时调用结束直播方法,视频会自动生成回放，用户可以重新根据需求查看已经直播过的视频，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。

获取观看实例：

```

WatchPlayback.Builder builder = new WatchPlayback.Builder().
    context(playbackView.getmActivity()). /** context*/
    containerLayout(playbackView.getContainer())./** 包含回放的布局*/
    callback(new WatchPlayback.WatchEventCallback() { /** 回放事件回调*/
        @Override
        public void onStartFailed(String reason) {
            /** 开始播放失败*/
        }

        @Override
        public void onStateChanged(boolean playWhenReady, int playbackState) {
            /** 播放过程中的状态信息*/
        }

        @Override
        public void onError(Exception e) {
            /** 播放出错*/
        }

        @Override
        public void onVideoSizeChanged(int width, int height) {
            /** 视频宽高改变*/
        }
    });
watchPlayback = builder.build();

```

```

/** 回放状态*/
switch (playbackState) {
    /** 闲置状态*/
    case VhallHlsPlayer.STATE_IDLE:
        break;
    /** 准备状态*/
    case VhallHlsPlayer.STATE_PREPARING:
        playbackView.showProgressbar(true);
        break;
    /** 正在加载*/
    case VhallHlsPlayer.STATE_BUFFERING:
        playbackView.showProgressbar(true);
        break;
    /** 准备就绪*/
    case VhallHlsPlayer.STATE_READY:
        playbackView.showProgressbar(false);
        playerDuration = getWatchPlayback().getDuration();
        playbackView.setSeekBarMax((int) playerDuration);
        break;
    /** 结束*/
    case VhallHlsPlayer.STATE_ENDED:
        stopPlay();
        break;
    default:
        break;
}

```

调整播放进度：

在观看回放的过程中，用户可以自由的拖动进度条来显示想看到的画面，调整播放进度。虽然播放器是在 SDK 中内部实现的，SDK 依旧回调了关于播放进度的方法。用户可以根据自己的逻辑展开调整，目前给出的方法如下（目前回调出的方法还不完善，用户需要什么信息是播放器中的而我们的 SDK 并没有回调出，也请联系我们的工作人员，我们会继续优化）

```

/**
 * 跳转到的位置
 */
getWatchPlayback().seekTo(playerCurrentPosition);

```

```

/**
 * 获取播放时长
 */
playerDuration = getWatchPlayback().getDuration();

```

```

/**
 * 获取当前的播放位置
 */
playerCurrentPosition = getWatchPlayback().getCurrentPosition();

```

开始播放：

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 start 方法，播放就会开始

```
/**
 * 请求观看回放
 */
getWatchPlayback().start();
```

暂停播放：

观看回放不同于观看直播，拥有暂停的功能。在观看直播的时候，并没有暂停的功能，只有停止直播的功能，而观看回放可以。执行的方法如下

```
/**
 * 暂停播放
 */
getWatchPlayback().pause();
```

结束播放：

当视频播放结束，播放器不在使用的时候，调用此方法。此方法会释放播放器所占用所有的资源

```
/**
 * 停止播放
 */
getWatchPlayback().stop();
```

调整播放界面：

调整播放界面的方式和观看直播相同，调用如下方法

```
/**
 * 调整播放界面
 */
getWatchPlayback().setScaleType(scaleType);
```

具体集成和调用方式请参考 [Demo](#)