微吼直播 SDK for Android

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	gao
2016.5.13	v2.2.1	新增帧率配置	qing
2016.6.3	v2.2.2	优化横竖屏切换后的全屏观看	qing
2016.7.25	v2.3.1	多分辨率切换	qing

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。

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
	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	http://e.vhall.com/home/vhallapi/embed 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

github 地址：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

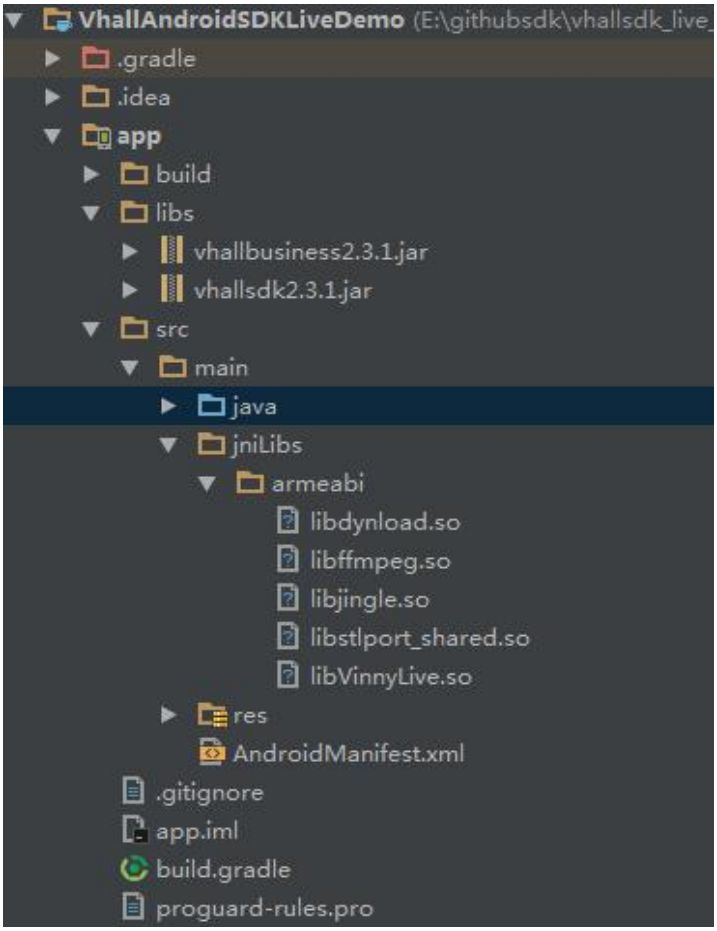
libVinnyLive.so

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、 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。

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
}
```

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： 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 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,传输等关于 Camera 的操作 VhallCameraView 需要初始化获得一些信息 调用 init()方法

```
getCameraView().init(pixel_type, Activity(), new RelativeLayout.LayoutParams(0, 0));
```

2 获取 VhallSDK 的实例 调用 startBroadcast() 这里传入参数活动 ID、TOKEN、Broadcast 实列、发起回调，该方法返回成功则推流成功，返回失败则推流失败。
以下是代码展示

```
VhallSDK.getInstance().startBroadcast(param.id, param.token, getBroadcast(), new
VhallSDK.BroadcastCallback() {
```

```
        @Override
        public void getStreaminfoSuccess() { //  Success }
        @Override
        public void getStreaminfoFailed(String reason) {//  Failed  reason 是原因}
    });
  }
```

3 Broadcast 实列 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
mView.getCameraView()).frameRate(param.frameRate).videoBitrate(param.videoBitrate).call
back(BroadcastEventCallback() {
```

new Broadcast.BroadcastEventCallback() 这里包含了发起后回调的一些信息和状态 ,这里做一个展示和简单描述

```
new Broadcast.BroadcastEventCallback() {
            @Override
            public void startFailed(String reason) { //发起失败  resaon 是原因}
             @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) { // 加载成功后的速度 kbps 返回每秒的速度}
            @Override
            public void onNetworkWeek() { // 网络环境差}
            @Override
            public void onNetworkFluency() { //网络通畅}
            @Override
            public void onStop() { // 已经停止}
        });
```

4、 APP 停止发起 -broadcast

获取 VhallSDK 的实例 调用 stopBroadcast() 传入参数活动 ID、TOKEN、Broadcast 实列、结束回调 以下是代码展示：

```
public void stopBroadcast() {
```

```
        VhallSDK.getInstance().stopBroadcast(param.id, param.token, getBroadcast(), new
VhallSDK.StopBroadcastCallback() {
            @Override
            public void stopSuccess() { // 停止成功}
            @Override
            public void stopFailed(String reason) { //停止失败}
        });
    }
```

5、 APP 开始观看 -watch

1 当 Activity 被创建 当前 Activity 必须包涵一个 FrameLayout 布局 此布局需要往 VhallSDK 中传递
2 获取 VhallSDK 的实例 调用 watchRtmpVideo() 这里传入参数 WatchRtmp 实列、活动 ID(必填)、用户名(必填)、用户邮箱(必填)、K 值校验、PPT(这里默认传空,后面会有介绍)观看回调
以下是代码展示：

```
VhallSDK.getInstance().watchRtmpVideo(getWatchRtmp(), params.id, "name",
"www.123@qq.com", params.k, ppt , new VhallSDK.WatchRtmpCallback() {
            @Override
            public void watchSuccess() { // 观看成功}
            @Override
            public void watchFailed(String msg) { // 观看失败}
            @Override
            public void watchGetPixelAvailable(Map map) { // 当前可用的观看线路}
        });
```

2 watchRtmp 实列 , 这里需要将一些设置信息传入 列如 actvity 、frameLayout(这里需要传入一个 FragmeLayout,用于生成观看)、回调 callback

```
  WatchRtmp.Builder builder = new
WatchRtmp.Builder().activity(activity).frameLayout(frameLayout).onBufferDelay(bufferSecond).ca
llback(WatchEventCallback()))
```

3 new WatchRtmp.WatchEventCallback() 这里包含了观看后回调的一些信息和状态 ,这里做一个展示和简单描述

```
 new WatchRtmp.WatchEventCallback() {
                @Override
                public void watchFailed(String reason) { // 观看失败}
                @Override
                public void watchConnectSuccess() { // 观看连接成功}
                @Override
```

```
        public void watchConnectFailed(String reason) { // 观看连接失败}
        @Override
        public void watchNeedReConnection(String reason) { // 需要重新连接 reason 原因}
        @Override
        public void watchLoadingSpeed(String kbps) { // 缓冲每秒的速率}
        @Override
        public void watchStartBuffering() {// 开始缓冲}
        @Override
        public void watchStopBuffering() {  // 停止缓冲}
       @Override
        public void watchStopVideoSuccess(String reson) { // 观看停止成功}
        @Override
        public void watchStopVideoFailed(String reson) { // 观看停止失败}
        @Override
        public void watchSwitchPixelSuccess() {// 观看切换分辨率成功}
        @Override
        public void watchSwitchPixelFailed(String reson) { // 观看切换分辨率失败}
    });
```

4 观看端多分辨率切换功能

目前定义的分辨率有

```
    public static final int SWITCH_PIXEL_DEFAULT = 0;   // 默认
    public static final int SWITCH_PIXEL_SD = 1;  // 标清
    public static final int SWITCH_PIXEL_HD = 2;  // 高清
    public static final int SWITCH_PIXEL_UHD = 3; // 超高清
```

功能实现 ：只需要将定义好的常亮传到 watchRtmp 实列中 ,调用 setDefinition(pixel)

备注 ： 通过 watchGetPixelAvailable 回调信息 我们可以知道当前的分辨率是否可用 状态为 0 不可用 ： 1 可用

5 观看端 PPT 播放

在直播的时候有时候会伴随着 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

```
  VhallPPT ppt = new VhallPPT();
```

2) 设置回调信息 返回当前文档的页数

```
setCallback(new VhallPPT.PPTChangeCallback() {
        @Override
        public void onPPTChage(String url) {
```

```
            // 每当 PPT 翻页回调次方法
        }
    });
```

3) 连接观看的时候将 PPT 传入，上文介绍观看连接时，讲此时的参数传入

6、 APP 结束观看

调用 VhallSDK.watchStopVideo（ ）底层会断开拉流,终止播放：

7、 APP 回放

当 App 发起直播结束时会生成回访,用户需要获取回访的地址用微吼的播放器播放即可
1) 请求播放地址 传入参数 活动 ID , name ,email ,K 值 回调信息 代码展示如下

```
VhallSDK.getInstance().getVideoURL(param.id, "test", "test@vhall.com", param.k, new
VhallSDK.VideoURLCallback() {
        @Override
        public void getURLSuccess(String url) { // 获得地址成功}
        @Override
        public void getURLFailed(String reason) { // 获得地址失败}
    }, ppt = new VhallPPT());
```

调用 VhallPPT 中的 getPPT(miao) 当前播放的时间
2) 初始化 VhallPlayer 并设置播放 代码展示如下：

```
String userAgent = Util.getUserAgent(playbackView.getActivity(), "VhallAPP");
    VhallPlayerListener mVhallPlayerListener = new VhallPlayerListener();
    VhallHlsPlayer mMediaPlayer = new VhallHlsPlayer(new
HlsRendererBuilder(playbackView.getActivity(),userAgent, mediaUrl));
    mMediaPlayer.addListener(mVhallPlayerListener);
    mMediaPlayer.prepare();
    mMediaPlayer.setSurface(videoView.getSurfaceView().getHolder().getSurface());
    mMediaPlayer.setPlayWhenReady(true);  // 设置播放
```

3) 播放器监听事件 代码展示如下：

```
 private class VhallPlayerListener implements VhallHlsPlayer.Listener {
    @Override
    public void onStateChanged(boolean playWhenReady, int playbackState) {
        switch (playbackState) {
            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://准备就绪
                break;
            case VhallHlsPlayer.STATE_ENDED:// 结束
                break;
            default:
                break;
            }
        }
    @Override
    public void onError(Exception e) {releasePlayer();}
    @Override
    public void onVideoSizeChanged(int width, int height,int unappliedRotationDegrees, float
pixelWidthHeightRatio) {// 宽高改变 }
  }
```