VhallTech

Vhall 业务层 for Android

修订记录

日期	版本号	描述	修订者
2016.5.06	v2.2.0	新增文档演示,优化观看体验	
2016.5.13	v2.2.1	新增帧率配置	
2016.6.03	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	
2016.9.19	v2.4.0	1 添加聊天问答功能 2 SDK架构调整	

快速集成

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。

权限开通申请

API&SDK权限申请 通过在线客服申请，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)

准备

1、 下载DEMO

https://github.com/vhall20/vhallsdk_live_android [https://github.com/vhall20/vhallsdk_live_android]

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、 需要导入的Jar

vhallsdk.jar
bussiness.jar

4、 需要导入的SO

Libdynload.so
Libffmpeg.so
Libjingle.so
libstlport_shared.so
libVinnyLive.so

5、 添加依赖

```
compile 'com.github.nkzawa:socket.io-client:0.6.0'//用于SDK网络连接
compile 'com.github.bumptech.glide:glide:3.7.0'//用于加载PPT
```

6、 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" />
```

快速接入

一、 基础配置

1、 初始化**SDK**

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  第三方用户注册应用时获得
 * @param APP_SECRET_KEY 第三方用户注册应用时获得
 */
VhallSDK.init(this,APP_KEY, APP_SECRET_KEY);
```

2、 **Vhall**第三方认证

认证地址： http://e.vhall.com/home/vhallapi/authlist [http://e.vhall.com/home/vhallapi/authlist]
Android端需要填写包名还有签名 这里需要保证SDK获取的包名和签名和认证时候输入的包名还有签名需要一致，否则返回身份验证失败

3、 基础参数说明

参数	描述
id	对应创建的活动ID(在官网创建)
recordId	片段ID（只在观看回放时使用，观看直播可以传空）
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	选择当前发起的分辨率
APP_KEY	第三方用户注册应用时获得
APP_SECRET_KEY	第三方用户注册应用时获得
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默认重新连接一次 重连时间约为5秒

登陆相关 **-login**

登陆 **-login**

1、 当用户在vhall平台注册成功用户之后，调用VhallSDK中的login方法，如果用户需要使用如聊天，问答等功能则必须登录。如果不登录则默认是游客模式
2、 参数描述

参数字段	参数描述

username	用户名
userpass	用户密码
vhallSDK.RequestCallback()	回调信息

3、 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) {}
    });
```

4、 返回参数描述

参数字段	参数描述
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 -errorCode

错误码	描述
10501	用户不存在
10502	登陆密码不正确

发起直播 -broadcast

准备工作

1、屏幕保持常亮。

```
getWindow().setFlags(WindowManager.LayoutParams.FLAG_KEEP_SCREEN_ON,
          WindowManager.LayoutParams.FLAG_KEEP_SCREEN_ON);
```

2、 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);
```

3、 设置发起布局

```
<com.vhall.business.VhallCameraView
  android:id="@+id/cameraview"
  android:layout_width="match_parent"
  android:layout_height="match_parent" />
```

4、 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));
```

发起直播

1、参数描述

参数字段	参数描述
id	活动ID
accessToken	访问token
Broadcast	发起实列
RequestCallback	回调信息

2、 以下是代码展示

```
VhallSDK.getInstance().initBroadcast(param.id, param.token, getBroadcast(), new VhallSDK.RequestCallback(){
        @Override
        public void success() {}

        @Override
        public void failed(int errorCode, String reason) {}
    });
```

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、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());
broadcast = builder.build();
```

4 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*/}
    }
```

4、 状态码和错误码

状态码	描述	**Broadcast**实列
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	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	
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	
20106	上传数据失败	

5 发起聊天 调用sendChat方法 ，

参数	描述
text	聊天内容
VhallSDK.RequestCallback	回调信息

```
getBroadcast().sendChat(text, new VhallSDK.RequestCallback() {
            @Override
            public void success() {
                chatView.clearInputContent();
            }

            @Override
            public void failed(int errorCode, String reason) {
                chatView.showToast(reason);
            }
        });
```

6 聊天信息返回 这里的ChatCallback 就是获取发起直播实例时传入的ChatCallback 回调。

```
private class ChatCallback implements ChatServer.Callback {
        @Override
        public void onChatServerConnected() {// 聊天服务器连接}

        @Override
        public void onConnectFailed(ChatServer chatServer) {// 连接错误，考虑是否重连chatServer.reConnect()}

        @Override
        public void onChatMessageReceived(ChatServer.ChatInfo chatInfo) {
            switch (chatInfo.event) {
                case ChatServer.eventMsgKey:// 消息
                    break;
                case ChatServer.eventOnlineKey:// 上线消息
                    break;
                case ChatServer.eventOfflineKey:// 下线消息
                    break;
                case ChatServer.eventQuestion:// 问答
                    break;
            }
        }

        @Override
        public void onChatServerClosed() { // 连接关闭}
    }
```

6 chatInfo描述

字段	描述
account_id	用户id
user_name	用户昵称
avatar	头像
room	活动id
event	消息类型 (用于区分消息用途)
time	发送时间

event 返回 **msg**	聊天消息
text	聊天消息

event 返回 **online offline**	上下线
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

event 返回 **question**	问答
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

7、 Broadcast 中其他方法简易描述

方法名	作用
changeCamera	切换摄像头
changeFlash	切换闪光灯
isAvaliable	是否已初始化
setAudioing	设置声音
destory	销毁当前直播实例

结束直播

1、 获取VhallSDK的实例 调用finishBroadcast() 传入参数活动ID、TOKEN、Broadcast实列、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。

2、 参数

参数字段	参数描述
id	活动ID
accessToken	访问token
Broadcast	发起实列
RequestCallback	回调信息

3、 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}
        });
```

观看直播 -watch

观看直播

1、 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watchRtmpVideo() 这里传入参数WatchRtmp实列、活动ID(必填)、用户名(未登录必填)、用户邮箱(未登录必填)、vhall_id（登录必填）K值校验

2、 参数描述

参数字段	参数描述
id	活动ID
nickname	用户名
email	用户邮箱
vhall_id	登陆获取的vhall_id 不传视为游客
password	密码
Watch	观看实例
RequestCallback	回调信息

3、 以下是代码展示 详细见Demo

```
VhallSDK.getInstance().initWatch(params.id, "test", "test@vhall.com",vhall_id, params.k, getWatchLive(), new VhallSDK.RequestCallback() {
            @Override
            public void success() {}

            @Override
            public void failed(int errorCode, String msg) {}
```

```
        });
```

4、 watchRtmp 实列 , 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(liveView.getmActivity())
               .containerLayout(liveView.getWatchLayout())
               .bufferDelay(params.bufferSecond)
               .callback(new WatchCallback())
               .messageCallback(new MessageEventCallback())
               .chatCallback(new ChatCallback());  // 如果使用聊天就加这个回调
      watchLive = builder.build();
```

5、 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
        }

    }
```

6、 状态码和错误码

状态码	描述	**Broadcast**实列
20251	观看连接成功	WatchLive.STATE_CONNECTED
20254	开始加载	WatchLive.STATE_BUFFER_START
20255	停止加载	WatchLive.STATE_BUFFER_STOP
20255	停止观看	WatchLive.STATE_STOP

错误码	描述
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
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
10049	用户信息不能为空
10410	用户信息不存在

聊天问答

1、 发送聊天， 在用户登陆成功的情况下可以发送消息。首先获取观看实列（此处需参考观看直播），调用sendChat方法，发送聊天信息.

2、 代码展示

```
getWatchLive().sendChat(text, new VhallSDK.RequestCallback() {
            @Override
            public void success() { // 发送成功 }

            @Override
            public void failed(int errorCode, String reason) {// 发送失败}
        });
```

3、 发送问答，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。

4、 代码展示

```
getWatchLive().sendQuestion(content, new VhallSDK.RequestCallback() {
            @Override
            public void success() { // 发送成功 }

            @Override
            public void failed(int errorCode, String reason) {// 发送失败 }
        });
```

5、 MessageEventCallback 消息监听 用于接收直播过程中的一些事件，目前可以使用PPT功能

```
 private class MessageEventCallback implements MessageServer.Callback {
        @Override
        public void onEvent(MsgInfo messageInfo) {
            switch (messageInfo.event) {
                case MsgInfo.EVENT_PPT_CHANGED://PPT翻页消息
                    // 如果有PPT 从messageInfo中取出PPT地址
                    break;
                case MsgInfo.EVENT_DISABLE_CHAT://禁言
                    // 暂不可用
                    break;
                case MsgInfo.EVENT_KICKOUT://踢出
                    // 暂不可用
                    break;
                case MsgInfo.EVENT_OVER://直播结束
                    // 直播结束时回调
                    break;
                case MsgInfo.EVENT_PERMIT_CHAT://解除禁言
                    // 暂不可用
                    break;
            }
        }
    }
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
                    如果有PPT 从messageInfo中取出PPT地址
                    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
                    liveView.showToast("直播已结束");
                    break;
                case MessageServer.EVENT_PERMIT_CHAT://解除禁言
                    // 暂不可用
                    break;
            }

        }

        @Override
        public void onMsgServerConnected() {// 消息服务器连接}

        @Override
        public void onConnectFailed() {消息服务器连接失败}

        @Override
        public void onMsgServerClosed() {//消息服务器关闭}
    }
```

6、 ChatCallback 聊天监听 发送的聊天问答信息从监听里获取 详情参考Demo

```
private class ChatCallback implements ChatServer.Callback {
        @Override
        public void onChatServerConnected() {// 聊天服务器连接}

        @Override
        public void onConnectFailed() {// 连接错误}

        @Override
        public void onChatMessageReceived(ChatServer.ChatInfo chatInfo) {
```

```
            switch (chatInfo.event) {
                case ChatServer.eventMsgKey:// 消息
                    break;
                case ChatServer.eventOnlineKey:// 上线消息
                    break;
                case ChatServer.eventOfflineKey:// 下线消息
                    break;
                case ChatServer.eventQuestion:// 问答
                    break;
            }
        }

        @Override
        public void onChatServerClosed() { // 连接关闭}
    }
```

7 新增方法

方法名	功能描述
connectChatServer	连接聊天服务器
disconnectChatServer	关闭聊天服务器
connectMsgServer	连接消息服务器
disconnectMsgServer	关闭消息服务器

8、 错误码

状态码	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

分辨率切换

1、 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(目前不能切换)
```

2、 功能实现 ： 只需要将定义好的常量传到watchLive实列中 ,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)
```

3、 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();
```

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)

2、 设置方法

```
//scaleType 观看类型
getWatchLive().setScaleType(scaleType);
```

结束观看

1、 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。

2、 代码展示如下

```
getWatchLive().stop();
```

观看回放

观看回放

1、 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。展示方案可以参考观看直播

2、 参数描述(和观看直播相同，实列要换成Watchplayback)

参数字段	参数描述
id	活动ID
nickname	用户名
email	用户邮箱
password	密码
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

3、 代码展示

```
VhallSDK.getInstance().initWatch(param.id, "test", "test@vhall.com", recordId , param.k, 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}
        });
```

4、 获取观看实列，代码展示

```
WatchPlayback.Builder builder = new WatchPlayback.Builder().context(context)
.containerLayout(playbackView.getContainer())
.callback(new WatchPlayback.WatchEventCallback() {
                @Override
                public void onStartFailed(String reason) {//开始播放失败}

                @Override
                public void onStateChanged(boolean playWhenReady, int playbackState) {//播放器过程中的状态信息
                 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:// 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 onError(Exception e) {//播放出错}

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

```
}
```

开始、暂停、结束

1、 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实列中的start方法，播放就会开始 。 pause方法就会暂停。stop方法就会停止

```
getWatchPlayback().start();
getWatchPlayback().pause();
getWatchPlayback().stop();
```

其他常用方法

5、 获取播放进度

```
getWatchPlayback().seekTo(playerCurrentPosition);
```

6、 获取当前播放进度

```
getWatchPlayback().getCurrentPosition();
```

7、 获取播放时长

```
getWatchPlayback().getDuration();
```
