

微吼直播 SDK for Android

微吼直播 中国领先的商务视频直播平台

目录

一、	修订记录.....	4
二、	简介	4
三、	API&SDK 的使用框架	5
四、	主要流程图	5
五、	权限开通申请	6
六、	SDK 使用准备	8
1、	下载 SDK&DEMO	8
2、	开发环境要求	8
3、	需要导入的 Jar	8
4、	需要导入的动态库 SO	8
5、	添加依赖	8
6、	权限及配置	9
七、	快速接入介绍	9
1、	初始化配置	9
2、	基础参数说明	10
3、	用户登录	10
4、	发起直播	12
5、	结束直播	15
6、	观看直播	16
7、	聊天	18
8、	问答	21
9、	分辨率切换/切换到单音频	22
10、	设置观看布局	22
11、	结束观看	23
12、	观看回放	23
13、	文档演示功能	26
八、	DEMO 简介	27
九、	第三方 K 值认证	28

1、	认证流程.....	28
2、	开启设置.....	28
3、	K 值使用.....	29
十、	版本迁移重点说明	30
1、	2.3.2 迁移到 2.4.0.....	30

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	新增用户管理/聊天/问答/应用签名 /增加观看语音直播	xy

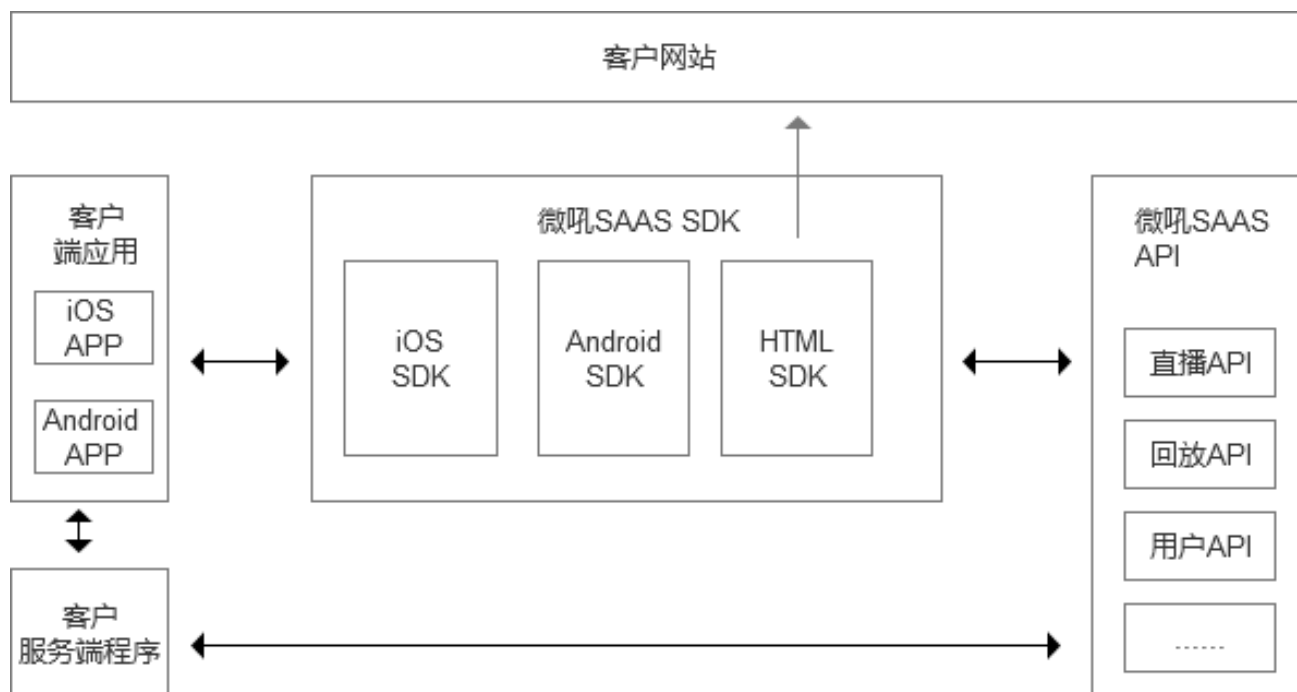
二、 简介

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：

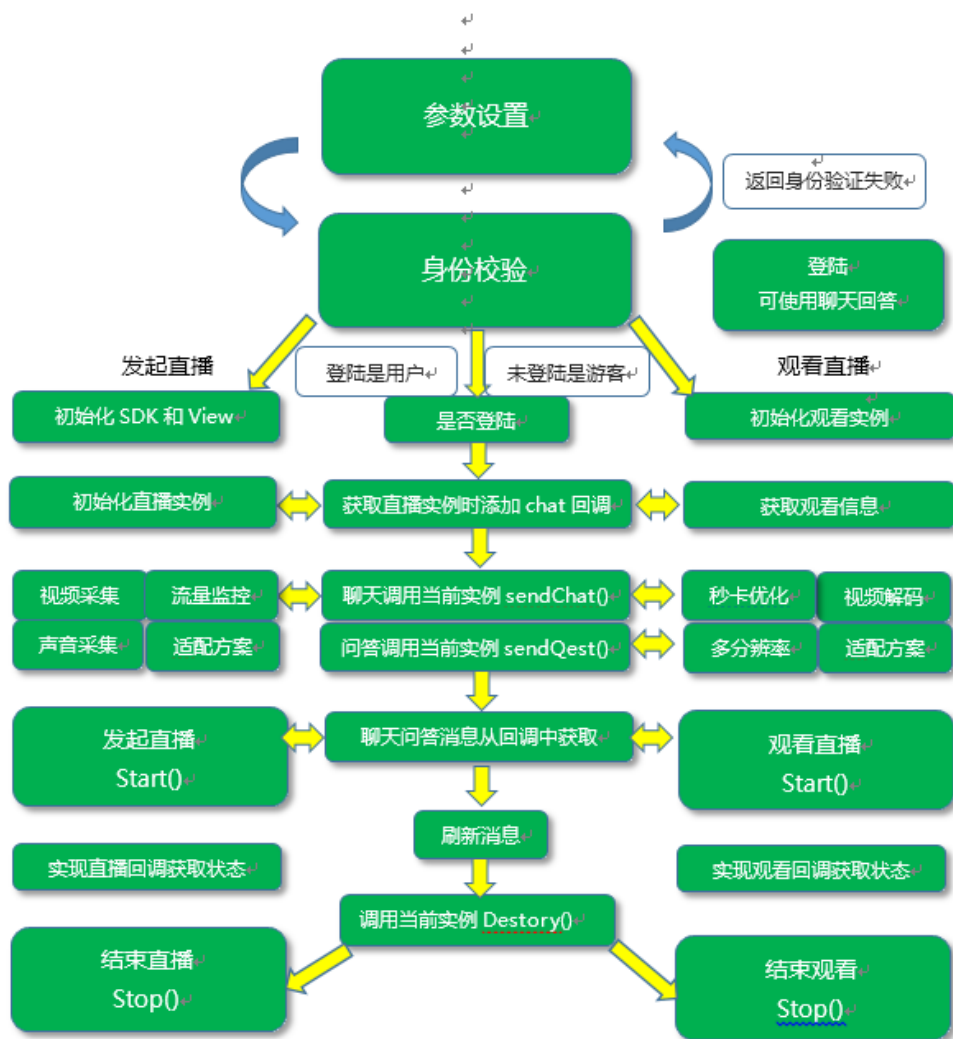
分类	特性名称	描述
发起直播	支持编码类型	音频编码：AAC，视频编码：H.264
	支持推流协议	RTMP
	视频分辨率	640*480
	屏幕朝向	横屏、竖屏
	闪光灯	开/关
	静音	开/关
	切换摄像头	前、后置摄像头
	目标码率	使用软编，码率固定在 300-400 之间，暂不可修改
	支持环境	Android 4.0 以上，
观看直播	支持播放协议	RTMP
	延时	RTMP: 2-4 秒
	支持解码	H.264
文档演示	支持文档演示	文档可与视频同步演示
观看回放	支持协议	HLS
权限	第三方 K 值认证	支持客户自己的权限验证机制来控制观看直播、观看回放的权限
用户登录（new）	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、 API&SDK 的使用框架



四、 主要流程图

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。
主要流程设计如下：



五、 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 Android-SDK 切页后输入以下信息：

*发行版本安全码SHA1 :

*包名 :

1) 包名: 在 AndroidManifest.xml 清单文件中

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.vhall.live">
```

其中 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:

- a) 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:

```
C:\Windows\system32\cmd.exe
Microsoft Windows [版本 6.1.7601]
版权所有 (c) 2009 Microsoft Corporation。保留所有权利。

C:\Users\LilWen>keytool -list -v -keystore d:\key2.keystore
输入keystore密码:

Keystore 类型: JKS
Keystore 提供者: SUN

您的 keystore 包含 1 输入

别名名称: asdf
创建日期: 2014-7-14
项类型: PrivateKeyEntry
认证链长度: 1
认证 [1]:
所有者: CN=sdfa
签发人: CN=sdfa
序列号: c8bd992
有效期: Mon Jul 14 16:02:54 CST 2014 至 Wed Jul 06 16:02:54 CST 2014
证书指纹:
    MD5:35:49:83:D1:CD:BB:8D:22:75:3F:12:C7:72:F7:B5:4E
    SHA1:7F:E0:4F:E3:83:71:89:B0:64:B5:17:29:77:8C:C9:70:C0:6A:98:AC
    签名算法名称:SHA256withRSA
    版本: 3

扩展:
#1: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
KeyIdentifier [
0000: 0A 4C 04 65 93 11 2E 78   BE 60 60 57 D4 BF CC 8F   .L.e...x.`W....
0010: 4F 7C F7 62               0..b
]
]
```

- b) 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,

assets	2016/9/22 14:59	文件夹	
com	2016/9/22 14:59	文件夹	
lib	2016/9/22 14:59	文件夹	
META-INF	2016/9/22 14:59	文件夹	
res	2016/9/22 14:59	文件夹	
src	2016/9/22 14:59	文件夹	
AndroidManifest.xml	2016/9/9 15:24	XML 文档	32 KB
classes.dex	2016/9/9 15:23	DEX 文件	5,138 KB
resources.arsc	2016/9/9 15:14	ARSC 文件	290 KB

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，

```
G:\META-INF>keytool -printcert -file CERT.RSA
所有者: CN=O=, L=nj, ST=js
发布者: CN=, L=nj, ST=js
序列号: 4d2
有效期开始日期: Sat Jan 08 15:50:51 CST 2011, 截止日期: Thu May 11 15:50:51 CST 3009
证书指纹:
MD5: F4:40:FC:CE:E:48:BD:83:D4:2:F:5D:41
SHA1: 14:24:C1:CC:4F:40:49:4F:60:04:03:01:00:85:07:45:01:08
SHA256: F2:7B:9D:FF:98:59:A9:0F:
:24:00:1D:E2
签名算法名称: SHA1withRSA
版本: 3
G:\META-INF>
```

六、 SDK 使用准备

1、 下载 SDK&DEMO

从 github 下载: <https://github.com/vhall20/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、 需要导入的 Jar

vhallsdk.jar、bussiness.jar

4、 需要导入的动态库 SO

Libffmpeg.so、libVinnylive.so

5、 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'

}
```

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" />
```

七、 快速接入介绍

1、 初始化配置

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);

```

其中：App_Key、App_Secret_Key：从此页面获取到，[立即查看权限信息](#)

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	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

3、用户登录

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) 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

5) 错误码

错误码	描述
10501	用户不存在
10502	登陆密码不正确

4、 发起直播

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	发起实例
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());

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*/}

}

```

5) 状态码和错误码

状态码	描述	Broadcast 实例
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、 结束直播

- 1) 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。

2) 参数

参数字段	参数描述
id	活动 ID
accessToken	访问 token
Broadcast	发起实例
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}

});
```

6、 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 实例、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
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) 以下是代码展示 详细见 Demo

```
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) watchRtmp 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(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

    }

}

```

7、 聊天

- 1) 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()`。
- 2) 参数说明：

参数	描述
----	----

text	聊天内容
VhallSDK.RequestCallback	回调信息

3) 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) {}
});
```

4) 获取聊天信息，ChatCallback 就是获取发起直播实例或者观看直播实例时传入的 ChatCallback 回调。

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

5) 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

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 :
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

8、 问答

1) 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。

2) 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 }
});
```

3) 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

4) 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

9、 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
```

2) 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)
```

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();
```

10、 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);
```

11、 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，调用此方法，SDK 会断开拉流。

2) 代码展示如下

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

12、 观看回放

1) 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) 获取观看实例，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:

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。 `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:

- 1) 获取播放进度

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

- 2) 获取当前播放进度

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

- 3) 获取播放时长

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

13、 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。

观看直播时的调用：

```
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}
}

```

观看回放视频时的调用：

```

private void setPPT() {
    if (ppt == null)
        ppt = new VhallPPT(); // 获取 VhallPPT 的引用
    getWatchPlayback().setVhallPPT(ppt); // 传入 PPT
}

if (ppt != null) {
    String url = ppt.getPPT(playerCurrentPosition / 1000); //根据播放器当前进度，获取当前 PPT 地址
    documentView.showDoc(url);
}
}

```

八、 DEMO 简介

1、 简介

DEMO 只针对核心功能进行演示，不包括 UI 界面设计。

2、 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：640*480
- 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](#)

3、 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](#) 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。

4、 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

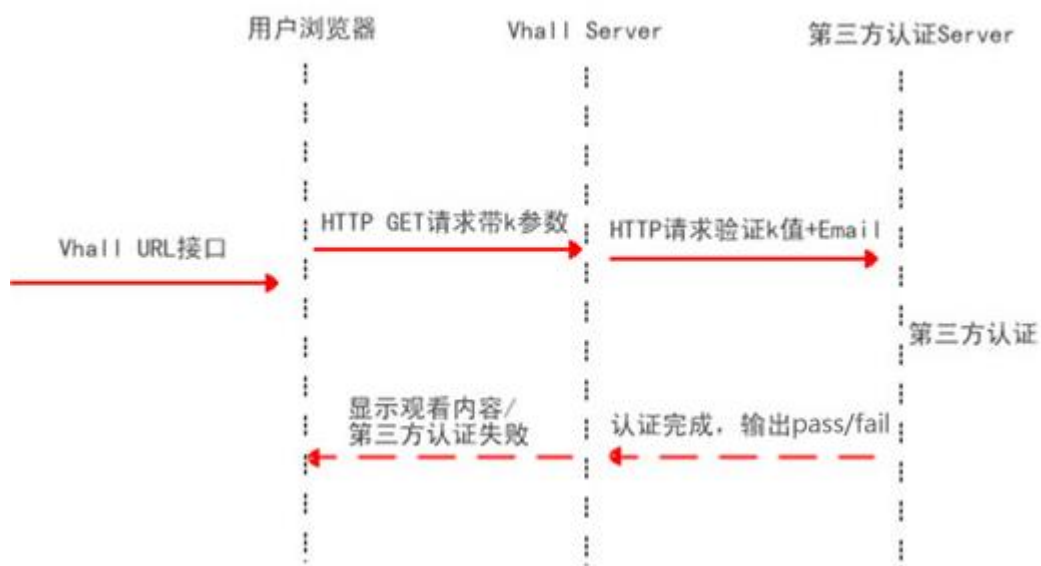
- 1) 通过微吼直播官网注册：<http://e.vhall.com/auth/register>
- 2) 通过接口调用注册：http://e.vhall.com/home/vhallapi/active#user_register [第三方创建用户](#)

九、 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



2、 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

- 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。
- 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
 - 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。
 - 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'

3、 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。
- 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

十、 版本迁移重点说明

1、 2.3.2 迁移到 2.4.0

1) 绑定签名信息

参考“四、权限开通申请”说明

2) 初始化

VhallSDK.init()添加 Context 参数

3) 发直播

- 丰富 Broadcast.builder，添加聊天服务器和消息服务器 callback，优化 BroadcastEventCallback 事件回调
- 添加发聊天功能

4) 看直播

- 丰富 WatchLive.builder，添加聊天服务器和消息服务器 callback，优化 WatchEventCallback 事件回调
- 废弃 VhallPPT，PPT 消息从消息服务器获取
- 添加发聊天、问答功能
- 初始化观看信息时获取增加字段 vhallid 和 片段 ID 发直播时传空，观看回放时传片段 ID ，如果没有则传空

5) 初始化发直播和看直播

添加 vhallid 参数，调用前需要注册用户