# Android 插件化开发指南

**（会议插件）**

**会议产品研发部**

**修订记录**

|  |  |  |  |
| --- | --- | --- | --- |
| **版本号** | **修订人** | **修订日期** | **修订内容** |
| 1.0.0 | 高翔 | 20180807 | * 初始版本 |
| 1.0.1 | 高翔 | 20181015 | * 修改插件依赖 |
| 2.0.13 | 李应(武汉UC事业部) | 20201022 | * 部分页面更改 |

目录

[一、 Android版本会议插件的介绍及导入](#_Toc490790381_WPSOffice_Level1) [7](#_Toc490790381_WPSOffice_Level1)

[1.1 会议插件的介绍及结构](#_Toc2032168058_WPSOffice_Level2) [7](#_Toc2032168058_WPSOffice_Level2)

[1.1.1 插件介绍](#_Toc2032168058_WPSOffice_Level3) [7](#_Toc2032168058_WPSOffice_Level3)

[1.1.2会议插件主要对外暴露类](#_Toc1068628918_WPSOffice_Level3) [7](#_Toc1068628918_WPSOffice_Level3)

[1.2 引用插件](#_Toc1068628918_WPSOffice_Level2) [7](#_Toc1068628918_WPSOffice_Level2)

[1.3 配置工程](#_Toc1040484965_WPSOffice_Level2) [10](#_Toc1040484965_WPSOffice_Level2)

[1.3.1 清单文件中权限申请](#_Toc1040484965_WPSOffice_Level3) [10](#_Toc1040484965_WPSOffice_Level3)

[1.3.2 复制指定的类](#_Toc471469234_WPSOffice_Level3) [11](#_Toc471469234_WPSOffice_Level3)

[1.3.3 注册自定义通知](#_Toc1916242055_WPSOffice_Level3) [11](#_Toc1916242055_WPSOffice_Level3)

[二、会议插件初始化和配置](#_Toc2032168058_WPSOffice_Level1) [12](#_Toc2032168058_WPSOffice_Level1)

[2.1 在onCreate()中初始化](#_Toc471469234_WPSOffice_Level2) [12](#_Toc471469234_WPSOffice_Level2)

[2.2 会议插件功能配置](#_Toc1916242055_WPSOffice_Level2) [13](#_Toc1916242055_WPSOffice_Level2)

[2.2.1 功能配置简介](#_Toc467964326_WPSOffice_Level3) [13](#_Toc467964326_WPSOffice_Level3)

[2.2.2 必须配置](#_Toc991311768_WPSOffice_Level3) [15](#_Toc991311768_WPSOffice_Level3)

[2.2.3 建议配置](#_Toc798751350_WPSOffice_Level3) [23](#_Toc798751350_WPSOffice_Level3)

[三、登录和登出](#_Toc1068628918_WPSOffice_Level1) [29](#_Toc1068628918_WPSOffice_Level1)

[3.1 登录](#_Toc467964326_WPSOffice_Level2) [29](#_Toc467964326_WPSOffice_Level2)

[3.1.1 登录](#_Toc693662053_WPSOffice_Level3) [29](#_Toc693662053_WPSOffice_Level3)

[3.1.2 获取登录状态](#_Toc1836888855_WPSOffice_Level3) [31](#_Toc1836888855_WPSOffice_Level3)

[3.1.3 获取异地登录状态](#_Toc366076713_WPSOffice_Level3) [32](#_Toc366076713_WPSOffice_Level3)

[3.2 登出](#_Toc991311768_WPSOffice_Level2) [33](#_Toc991311768_WPSOffice_Level2)

[四、可接入界面](#_Toc1040484965_WPSOffice_Level1) [33](#_Toc1040484965_WPSOffice_Level1)

[4.1 进行中会议及预约会议列表（YHCReserveListFragment）](#_Toc798751350_WPSOffice_Level2) [33](#_Toc798751350_WPSOffice_Level2)

[1、发起会议；](#_Toc110666736_WPSOffice_Level3) [34](#_Toc110666736_WPSOffice_Level3)

[2、加入会议；](#_Toc254993650_WPSOffice_Level3) [34](#_Toc254993650_WPSOffice_Level3)

[3、查看进行中会议及历史会议；](#_Toc1448399785_WPSOffice_Level3) [34](#_Toc1448399785_WPSOffice_Level3)

[4、进去会议详情页面；](#_Toc1528047750_WPSOffice_Level3) [34](#_Toc1528047750_WPSOffice_Level3)

[5、其他功能。](#_Toc141599777_WPSOffice_Level3) [34](#_Toc141599777_WPSOffice_Level3)

[4.2 历史会议列表（YHCHistoryListFragment）](#_Toc693662053_WPSOffice_Level2) [34](#_Toc693662053_WPSOffice_Level2)

[1、按照时间线显示历史会议；](#_Toc455571163_WPSOffice_Level3) [35](#_Toc455571163_WPSOffice_Level3)

[2、历史会议包含已取消的会议和已结束的会议；](#_Toc1005334986_WPSOffice_Level3) [35](#_Toc1005334986_WPSOffice_Level3)

[3、进入会议详情页面；](#_Toc263775106_WPSOffice_Level3) [35](#_Toc263775106_WPSOffice_Level3)

[4、其他功能。](#_Toc861959134_WPSOffice_Level3) [35](#_Toc861959134_WPSOffice_Level3)

[4.3 立即会议页面（YHCConfStartAndReserveActivity）](#_Toc1836888855_WPSOffice_Level2) [35](#_Toc1836888855_WPSOffice_Level2)

[1、 立即会议；](#_Toc22482476_WPSOffice_Level3) [36](#_Toc22482476_WPSOffice_Level3)

[2、 其他功能。](#_Toc2053335907_WPSOffice_Level3) [36](#_Toc2053335907_WPSOffice_Level3)

[4.4 预约会议页面（YHCConfStartAndReserveActivity）](#_Toc366076713_WPSOffice_Level2) [36](#_Toc366076713_WPSOffice_Level2)

[1、 预约会议；](#_Toc354381659_WPSOffice_Level3) [36](#_Toc354381659_WPSOffice_Level3)

[2、其他功能。](#_Toc1120389682_WPSOffice_Level3) [37](#_Toc1120389682_WPSOffice_Level3)

[4.5 加入会议页面（YHCJoinConfActivity）](#_Toc110666736_WPSOffice_Level2) [37](#_Toc110666736_WPSOffice_Level2)

[1、加入会议；](#_Toc1252768478_WPSOffice_Level3) [37](#_Toc1252768478_WPSOffice_Level3)

[2、其他功能。](#_Toc1350134558_WPSOffice_Level3) [37](#_Toc1350134558_WPSOffice_Level3)

[4.6 会议页面（YHCConfDetailActivity）](#_Toc254993650_WPSOffice_Level2) [37](#_Toc254993650_WPSOffice_Level2)

[1、未开始的会议可以邀请成员，编辑议题；](#_Toc1399302104_WPSOffice_Level3) [38](#_Toc1399302104_WPSOffice_Level3)

[2、查看会议成员；](#_Toc977043631_WPSOffice_Level3) [38](#_Toc977043631_WPSOffice_Level3)

[3、结束的会议可以查看摘要；](#_Toc1512341255_WPSOffice_Level3) [38](#_Toc1512341255_WPSOffice_Level3)

[4、跳转会议开始页面；](#_Toc303026893_WPSOffice_Level3) [38](#_Toc303026893_WPSOffice_Level3)

[5、分享会议信息；](#_Toc1289263614_WPSOffice_Level3) [39](#_Toc1289263614_WPSOffice_Level3)

[6、取消未开始的会议；](#_Toc543562268_WPSOffice_Level3) [39](#_Toc543562268_WPSOffice_Level3)

[7、跳转更新会议页面；](#_Toc255603938_WPSOffice_Level3) [39](#_Toc255603938_WPSOffice_Level3)

[8、其他。](#_Toc968091966_WPSOffice_Level3) [39](#_Toc968091966_WPSOffice_Level3)

[五、主动调用的方法](#_Toc471469234_WPSOffice_Level1) [39](#_Toc471469234_WPSOffice_Level1)

[5.1 发起立即会议](#_Toc1448399785_WPSOffice_Level2) [39](#_Toc1448399785_WPSOffice_Level2)

[5.2 创建预约会议](#_Toc1528047750_WPSOffice_Level2) [39](#_Toc1528047750_WPSOffice_Level2)

[5.3 加入会议](#_Toc141599777_WPSOffice_Level2) [41](#_Toc141599777_WPSOffice_Level2)

[5.4 更新会议](#_Toc455571163_WPSOffice_Level2) [41](#_Toc455571163_WPSOffice_Level2)

[5.5 获取会议相关设置](#_Toc1005334986_WPSOffice_Level2) [42](#_Toc1005334986_WPSOffice_Level2)

[5.6 会议插件全局设置](#_Toc263775106_WPSOffice_Level2) [42](#_Toc263775106_WPSOffice_Level2)

[1) 设置自己加入会议是否关闭语音](#_Toc1385562890_WPSOffice_Level3) [43](#_Toc1385562890_WPSOffice_Level3)

[2) 设置自己加入会议是否关闭视频](#_Toc1990307809_WPSOffice_Level3) [43](#_Toc1990307809_WPSOffice_Level3)

[3) 设置他人加入会议是否关闭语音](#_Toc1898060191_WPSOffice_Level3) [44](#_Toc1898060191_WPSOffice_Level3)

[4) 设置他人加入会议是否关闭视频](#_Toc1975537599_WPSOffice_Level3) [44](#_Toc1975537599_WPSOffice_Level3)

[5.7 会议插件全局设置状态的获取](#_Toc861959134_WPSOffice_Level2) [44](#_Toc861959134_WPSOffice_Level2)

[1) 获取自己加入会议是否关闭语音的设置状态](#_Toc615760126_WPSOffice_Level3) [45](#_Toc615760126_WPSOffice_Level3)

[2) 获取自己加入会议是否关闭视频的设置状态](#_Toc356742789_WPSOffice_Level3) [45](#_Toc356742789_WPSOffice_Level3)

[3) 获取他人加入会议是否关闭语音的设置状态](#_Toc1712299_WPSOffice_Level3) [45](#_Toc1712299_WPSOffice_Level3)

[4) 获取他人加入会议是否关闭视频的设置状态](#_Toc861321882_WPSOffice_Level3) [45](#_Toc861321882_WPSOffice_Level3)

## Android版本会议插件的介绍及导入

温馨提示：有[下划线](#_Android版本IM插件的介绍及导入)的文字为超链接，超链接跳转请同时按ctrl和鼠标左键

### 会议插件的介绍及结构

#### 1.1.1 插件介绍

会议插件为开发者开发会议相关的应用提供的一套完善的开发框架，将会议基础能力和界面集成到aar文件中，方便开发者快速高效的集成会议能力，开发者可以基于我们提供的会议插件实现自己的应用，也可以基于Demo开发。

#### 1.1.2会议插件主要对外暴露类

1、SDKCoreHelper: 里面有登录/登出方法，注册/注销登陆状态的广播接收者；

2、YHCConferenceMgr:里面主要有创建会议、预约会议、加入会议等方法；

3、YHCConferenceConfig:这个类为会议插件配置类，可以配置

1) 邀请成员的点击事件

2) 配置昵称和头像及点击

3）会议信息分享的点击事件

4）会中白板的相关设置

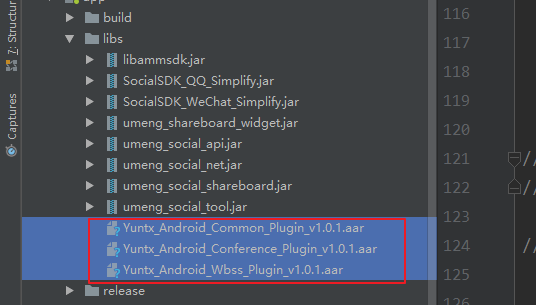
4、YHCConfInfo:创建会议、预约会议及更新会议函数中参数对象。

### 引用插件

1、开发者主Module的build.gradle配置文件中添加如下代码

android {  
 //.........  
 repositories {  
 flatDir {  
 dirs 'libs'  
 }  
 }  
}

2、在开发者主Module的libs里面添加需要的aar文件（如下图所示）



3、在开发者主Module的build.gradle配置文件中依赖导入的aar文件

**注意事项：**若之前集成过其他插件，已依赖如下某个aar文件，如

Yuntx\_Android\_Common\_Plugin\_v1.0.x，则无需再次依赖

dependencies {  
 // ........  
 implementation (name: 'Yuntx\_Android\_Common\_Plugin\_v1.0.1', ext: 'aar')  
 implementation (name: 'Yuntx\_Android\_Conference\_Plugin\_v1.0.1', ext: 'aar')  
 implementation (name: 'Yuntx\_Android\_Wbss\_Plugin\_v1.0.1', ext: 'aar')  
 // ........  
}

4、在开发者主Module的build.gradle配置文件中依赖如下第三方包

**注意事项:**如果之前已有以下某个第三方库，可使用原有的，也可以根据自己需求更换新的版本

dependencies {  
 // ........ x.x.x建议修改为 buildToolsVersion 同样的内容  
 implementation 'com.android.support:support-v4:x.x.x'

implementation 'com.android.support:appcompat-v7:x.x.x'

implementation 'com.android.support:cardview-v7: x.x.x '  
 implementation 'com.android.support:design:x.x.x '  
 implementation 'com.android.support:recyclerview-v7: x.x.x'  
 implementation 'com.belerweb:pinyin4j:2.5.1'  
 implementation 'org.jsoup:jsoup:1.10.1'  
 implementation 'com.github.bumptech.glide:glide:3.7.0'  
 implementation 'org.jsoup:jsoup:1.10.1'  
 implementation 'io.reactivex:rxjava:1.1.1'  
 implementation 'io.reactivex:rxandroid:1.1.0'  
 implementation 'com.android.support:cardview-v7:26.1.0'

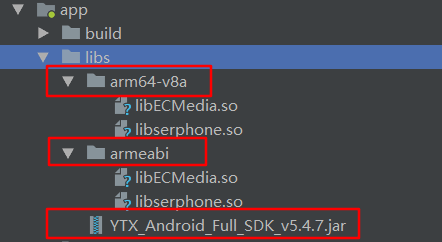
implementation ('com.squareup.retrofit2:converter-gson:2.1.0') {

transitive = true;

}

}

### 5、添加对应的so文件



\*如果需要集成会议功能，则需要导入全功能的SDK：YTX\_Android\_Full\_SDK.zip。

\*将解压后的文件导入您工程下的libs目录下。Jar版本号以SDK目录下放置的Jar包实际版本为准。

全能力版 SDK导入libs目录如图所示：

说明：

1).libserphone.so: IM消息和音视频以及音视频会议功能库，必须添加。

2).libECMedia.so: 使用音视频功能，必须添加。

3).libyuntx\_gl\_disp.so: 5.3.0以及之后版本添加了视频图像绘制库，使用ECOpenGLView控件需要添加。

4).libvoicechange.so: 5.3.0以及之后版本添加了语音变声接口库，用来将录制的语音文件进行音频数据处理转换成不同发声效果。

5).arm64-v8a目录中是64位的so库，适用于CPU是ARM64位的终端设备。

6).armeabi目录中是32位的so库，适用于CPU是通用ARM的终端设备。

7). YTX\_Android\_Full\_SDK\_v5.x.xr.jar是SDK对外提供的Jar包。

注意事项：

1).如果开发者项目需要添加armeabi-v7a目录，可将armeabi目录下的so拷贝至armeabi-v7a目录。

2).目前云通讯只支持arm系列的CPU。其他类型（X86，MIPS等）不支持，因此不能在libs目录下添加ARM类型以外的适配目录

### 配置工程

#### 1.3.1 清单文件中权限申请

// 开发者的 APP 接入时，请将 com.yuntongxun.plugin.conf替换为自己的包名  
// 和下面的 uses-permission 一起加入到你的 AndroidManifest 文件中  
<permission  
 android:name="com.yuntongxun.plugin.conf.permission.RECEIVE\_MSG"  
 android:protectionLevel="signature" />  
<uses-permission android:name="com.yuntongxun.plugin.conf.permission.RECEIVE\_MSG"/>  
<uses-permission android:name="android.permission.INTERNET" />  
<!-- 允许程序访问有关GSM网络信息 -->  
<uses-permission android:name="android.permission.ACCESS\_NETWORK\_STATE" />  
<!-- 允许程序访问Wi-Fi网络状态信息 -->  
<uses-permission android:name="android.permission.ACCESS\_WIFI\_STATE" />  
<uses-permission android:name="android.permission.WRITE\_EXTERNAL\_STORAGE" />  
<uses-permission android:name="android.permission.READ\_PHONE\_STATE" />  
<uses-permission android:name="android.permission.GET\_TASKS" />  
<!-- 允许访问振动设备 -->  
<uses-permission android:name="android.permission.VIBRATE" />  
<uses-permission android:name="android.permission.RECORD\_AUDIO" />  
<uses-permission android:name="android.permission.READ\_CONTACTS" />  
<uses-permission android:name="android.permission.CALL\_PHONE" />  
<!-- 允许程序禁用键盘锁 -->  
<uses-permission android:name="android.permission.DISABLE\_KEYGUARD" />  
<!-- 允许使用PowerManager的WakeLocks保持进程在休眠时从屏幕消失(非必选,由应用选择是否需要) -->  
<uses-permission android:name="android.permission.WAKE\_LOCK" />  
<uses-permission android:name="android.permission.CAMERA" />  
<!-- 允许程序修改全局音频设置 -->  
<uses-permission android:name="android.permission.MODIFY\_AUDIO\_SETTINGS" />  
<uses-permission android:name="android.permission.SYSTEM\_ALERT\_WINDOW" />

#### 1.3.2 复制指定的类

请把下面的自定义广播类YuntxNotifyReceiver.java文件粘贴到开发者的module中



#### 1.3.3 注册自定义通知

**注意事项：**

1、如果之前集成过容联云的其他插件，已经注册过自定义YuntxNotifyReceiver，那么可以使用已注册过的，也可以重新按照以下流程注册

2、本配置是属于application的属性，需要复制到AndroidManifest 清单文件的application节点内。

3、声明自定义通知的广播接收器，开发者需要把action 中的 com.yuntongxun.plugin.conf替换为自有项目APP的包名

// 声明自定义通知的广播接收器（需要替换包名，详见注意事项）  
<receiver  
 android:name="YuntxNotifyReceiver在开发者module中的位置"  
 android:exported="false">  
 <intent-filter>  
 // 接收push消息  
 // action匹配规则{packageName + ".action" + "功能"}  
 // com.yuntongxun.plugin.conf 替换成自己的报名  
 // 以下三条action 可以根据自己需求来进行增删  
 // 用于电话会议  
 <action android:name="com.yuntongxun.plugin.conf.action.VOIP\_CALL" />  
 // 用于IM消息  
 <action android:name="com.yuntongxun.plugin.conf.action.MESSAGE" />  
 // 用于通知点击  
 <action android:name="com.yuntongxun.plugin.conf.action.NOTIFICATION\_CLICK" />  
 </intent-filter>  
</receiver>  
// 应用处理程序逻辑服务 name为YuntxNotifyReceiver的位置加 $NotifyService  
<service android:name="YuntxNotifyReceiver的位置$NotifyService" />

## 二、会议插件初始化和配置

### 2.1 在onCreate()中初始化

**注意事项：**请开发者拷贝下面代码到自己的application的onCreate()方法中，此步骤是为保证用户的app在自动登陆模式中的数据完整

//初始化插件上下文  
SDKCoreHelper.setContext(this);

### 2.2 会议插件功能配置

#### 2.2.1 功能配置简介

会议插件功能配置代码必须放置在开发者的application的onCreate[()](#_2.1_实现插件公共接口)方法中，分为文档[2.3.2](#_2.3.2_必须配置)必须配置和文档[2.3.3](#_2.3.3建议配置)建议配置。

文档2.3.1为代码编写方式示例，共有[方法1](#_1)（方法1）配置IM通知消息点击事件)到[方法5](#_2)（方法5）会议信息通知回调)六 五个方法，五个方法的详细使用方法见文档2.3.2和2.3.3介绍：

##### 1) 推荐配置方式一

/\*\*  
 \* 推荐配置一  
 \* 第一步 先创建 YHCConfConfig 的实例  
 \* 第二步 YHCConferenceMgr.getHelper().init(YHCConfConfig)  
 \* 注意事项：下面的方法可以根据需求进行配置，详情见文档后面接口讲解  
 \*/  
YHCConferenceConfig configuration = new YHCConferenceConfig.ConferenceConfigBuilder(this)  
 /\*\* 方法1 配置昵称、头像及点击事件 ↓ \*/  
 .setOnConferBindViewListener(onConferBindViewListener)  
 /\*\* 方法2 配置获取成员的点击事件↓ \*/  
 .setOnGetMemberListener(onGetMemberListener)  
 /\*\* 方法3 配置分享会议信息的点击事件↓ \*/  
 .setOnConfShareListener(onConfShareListener)  
 /\*\* 方法4 配置会中白板的相关功能↓ \*/  
 .setOnConfWbssShowListener(onConfWbssShowListener)

/\*\* 方法5 配置会议信息的相关通知↓ \*/

setOnReceiveConfNotifylistener(onReceiveConfNotifylistener)  
 /\*\* 创建 \*/  
 .build();  
YHCConferenceMgr.getManager().init(configuration);

##### 2) 推荐配置方式二

**注意事项：**推荐配置二 不再过多介绍，详见会议插件demo中ConfApplication中

/\*\*  
 \* 推荐配置方式二 同为链式调用，但创建统一的接口实现类，在实现类中配置接口的方法  
 \* 第一步 先创建 YHCConfConfig 的实例  
 \* 第二步 YHCConferenceMgr.getHelper().init(YHCConfConfig)  
 \* 推荐配置二 不再过多介绍，详见会议插件demo中ConfApplication中  
 \*/  
YHCConferenceConfig configuration = new YHCConferenceConfig.ConferenceConfigBuilder(this)  
 /\*\* 方法1 配置昵称、头像及点击事件 ↓ \*/  
 .setOnConferBindViewListener(ConfImpl.getInstance())  
 /\*\* 方法2 配置获取成员的点击事件↓ \*/  
 .setOnGetMemberListener(ConfImpl.getInstance())  
 /\*\* 方法3 配置分享会议信息的点击事件↓ \*/  
 .setOnConfShareListener(ConfImpl.getInstance())  
 /\*\* 方法4 配置会中白板的相关功能↓ \*/  
 .setOnConfWbssShowListener(ConfImpl.getInstance())

/\*\* 方法5 配置会议信息的相关通知↓ \*/

setOnReceiveConfNotifylistener(ConfImpl.getInstance())  
 /\*\* 创建 \*/  
 .build();  
YHCConferenceMgr.getManager().init(configuration);

#### 2.2.2 必须配置

##### 1)（方法1）配置昵称、头像及点击事件

**作用位置：**

  
**使用场景：**如果不配置方法1，那么默认把memberId当做昵称，头像使用默认头像，头像默认不能点击

//此方法为 初始化头像，昵称，头像点击事件的  
private OnConferBindViewListener onConferBindViewListener = new OnConferBindViewListener() {  
 /\*\*  
 \* 此方法是为 设置聊天对象昵称  
 \* **@param** context

**\* @param** tag 扩展字段，后续扩展使用，可不关注  
 \* **@param** memberId 成员的memberId  
 \* **@param** idType 成员的memberId类型 MEMBER\_APP\_NUM 成员app参会,  
 \* MEMBER\_PHONE\_NUM 成员手机参会  
 \*/  
 @Override  
 public String onBindNickName(Context context, String tag, String memberId, MEMBER\_TYPE idType) {  
 //此处由开发者设置昵称，不设置则默认使用userId作为昵称  
 //示例  
 //memberIdList为开发者的memberId集合  
 List<String> memberIdList = new ArrayList<String>();  
 for (String s : memberIdList) {  
 if (s.equals(memberId)) {  
 if (idType == MEMBER\_TYPE.MEMBER\_APP\_NUM) {  
 return "开发者查询后定义的昵称";  
 } else {  
 return "开发者查询后定义的昵称";  
 }  
 }  
 }  
 return memberId;  
 }  
 /\*\*  
 \* 此方法是为 配置会议成员的头像  
 \* **@param** context  
 \* **@param** avatar 头像的显示控件 ImageView  
 \* **@param** avatarType 头像显示的页面类型 可参考页面来设置不同头像  
 \* CONF\_RUNNING\_MEMBER 会中头像,  
 CONF\_CONTROL\_MEMBER 会控页面头像,  
 CONF\_NOW\_LIST 会议列表页面;  
 \* **@param** memberId 成员的memberId  
 \* **@param** idType 成员的memberId类型  
 \* **@return** 返回值为头像的 url 地址  
 \*/  
 @Override  
 public void onBindAvatar(Context context, ImageView avatar, AVATAR\_TYPE avatarType, String memberId, MEMBER\_TYPE idType) {  
 //代码示例  
 if (memberId.equals("00000") && idType == MEMBER\_TYPE.MEMBER\_APP\_NUM) {  
 avatar.setImageResource(R.drawable.ic\_launcher);  
 }  
 }  
 /\*\*  
 \* 此方法是为 配置头像点击事件  
 \* **@param** context  
 \* **@param** memberId 成员的memberId  
 \* **@param** idType 成员的memberId类型  
 \*/  
 @Override  
 public void OnAvatarClickListener(Context context, String memberId, MEMBER\_TYPE idType) {  
 //代码示例  
 if (memberId.equals("aaa")){  
 ToastUtil.showMessage("bbb");  
 }  
 }  
};

##### 2)（方法2）配置添加成员事件

**作用位置（添加成员）：** 

###### a 声明回调和设置监听事件（配置页面）

**使用场景：**先声明静态成员变量,返回YHCConfMember集合的回调onReturnMemberCallBack，然后设置监听事件

//设置返回 YHCConfMember的回调  
public static OnReturnMemberCallback onReturnMemberCallback;  
  
//设置返回 YHCConfMember的监听  
OnGetMemberListener onGetMemberListener = new OnGetMemberListener() {  
 /\*\*  
 \* **@param** return\_type 后期扩展字段 可考虑去除  
 \* RETURN\_TYPE.APPOINTMENT\_CONFERENCE 预约会议  
 \* RETURN\_TYPE.INSTANT\_CONFERENCE 即时会议

RETURN\_TYPE.OUTCALL\_CONFERENCE 落地呼叫  
 \* **@param** callback 返回YHCConfMember的回调  
 \* **@param** members 目前列表里面已存在的成员

\* **@param** maxMembers 当前会议支持得最大人数  
 \*/  
 @Override  
 public void onGetConfMember(Context context, RETURN\_TYPE return\_type, OnReturnMemberCallback callback, List<YHCConfMember> members, int maxMembers) {  
 onReturnMemberCallback = callback;  
 if (return\_type == RETURN\_TYPE.APPOINTMENT\_CONFERENCE || return\_type == RETURN\_TYPE.INSTANT\_CONFERENCE) {  
 context.startActivity(new Intent(context, ConferenceSelectContactActivity.class));  
 }  
 }  
};

###### b 返回选取值（跳转页面）

**注意事项**：1、请开发者用 声明onReturnMemberCallback所在的applicaton名字替换下面方法中的ConfApplication  
2、调用该方法，传入YHCConfMember bean的集合

**使用场景： 在会控页面或者发起预约会议页面**点击添加成员按钮，跳转到开发者在 [onGetConfMember()](#_1)_声明回调和设置监听事件（配置页面）)指定的页面，需要添加成员的对象集合后，调用该方法

/\*\*  
 \* 此方法为 返回选取的YHCConfMember集合   
 \* 如果是app参会，格式如下   
 \* YHCConfMember member = new YHCConfMember ();  
 member.setAccount(账号);  
 member.setName(昵称);  
 \* 如果是手机落地电话参会，格式如下  
 YHCConfMember member = new YHCConfMember ();  
 member.setPhoneNum (电话号);  
 member.setName(昵称);  
 \*/  
ConfApplication.onReturnMemberCallback.returnMembers(meetingMembers);

##### 3)（方法3）配置分享会议信息点击事件

**作用位置：**



**使用场景：**如果不配置[方法3](#_3)（方法3）配置昵称、头像及点击事件)，那么默认隐藏分享

//此方法为配置分享会议信息点击事件  
private OnConfShareListener onConfShareListener = new OnConfShareListener() {  
 /\*\*  
 \* 配置分享会议信息的回调  
 \* **@param** activity 分享页面  
 \* **@param** info 分享的会议信息 包含inviterUserId 邀请者id  
 \* createrUserId 创建者id meetingName 会议名字  
 \* meetingNo 会议号 meetingStartTime 会议开始事件  
 \* **@param** type 分享的类型 SMS QQ WEIXIN  
 \*/  
 @Override  
 public void onConfShare(Activity activity, YHCShareInfo info, SHARE\_CONF\_TYPE  
 type) {  
 //代码示例  
 ShareUtil.getInstance().sendSMS(activity,"【有会】"  
 +info.getInviterUserId()+"邀请您参加会议\n"  
 +"会议名字为："+info.getMeetingName()+"\n"  
 +"开始时间为："+info.getMeetingStartTime()+"\n"  
 +"会议号为："+info.getMeetingNo()+"。" );  
 }

/\*\*

\* 配置显示分享的类型

\* @return SHARE\_SHOW\_TYPE[] 开发者可自行设置显示那几个分享图标

\*/

@Override

public SHARE\_SHOW\_TYPE[] onGetShareShows() {

return new SHARE\_SHOW\_TYPE[]{SHARE\_SHOW\_TYPE.QQ\_SHARE, SHARE\_SHOW\_TYPE.WEIXIN\_SHARE, SHARE\_SHOW\_TYPE.SMS\_SHARE};

}

/\*\*

\* 此方法是为了 便于开发者知道分享的结果 可不关注

\*/

@Override

public void onShareActivityResult(Activity activity, int requestCode, int resultCode, Intent data) {

//此行代码为示例

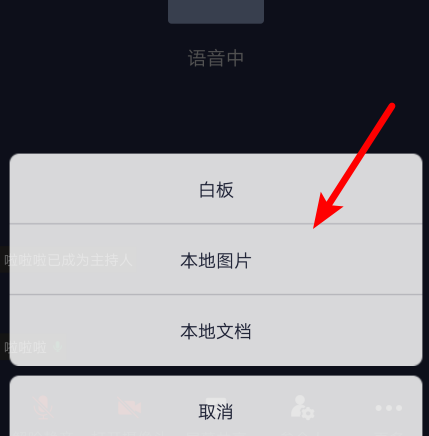
UMShareAPI.get(activity).onActivityResult(requestCode, resultCode, data);//完成回调

}  
};

#### 2.2.3 建议配置

##### 1)（方法4）会中白板相关功能配置

**作用位置：**



**使用场景：**如果开发者需要开启会中白板功能，请实现该回调

/\*\*  
 \* 此函数为配置会中白板功能  
 \*/  
private OnConfWbssShowListener onConfWbssShowListener = new OnConfWbssShowListener() {  
 /\*\*  
 \* 白板功能初始化  
 \* **@param** context  
 \* **@return** true 代表启用会用白板功能 false 代表关闭白板功能  
 \*/  
 @Override  
 public boolean onInitConfig(Context context) {  
 /\*\*白板功能配置\*/

WbssConfiguration configuration = new WbssConfiguration.Builder(context)

.serverUrl("你的白板服务器地址")

.appKey("你的白板appkey").build();

WbssManagerUtil.getInstance().init(configuration);

return true;  
 }

/\*\*  
 \* 此方法为配置白板弹出的类型，包括本地图片、本地文件、网络文件  
 \* **@return** WBSS\_SHOW\_TYPE   
 \*/  
 @Override  
 public WBSS\_SHOW\_TYPE[] onGetWbssShowTypes() {  
 //示例代码 显示图片和本地文件选项  
 return new WBSS\_SHOW\_TYPE[]{WBSS\_SHOW\_TYPE.SELECT\_PHOTO,

WBSS\_SHOW\_TYPE.SELECT\_FILE};

}

/\*\*  
 \* 白板弹框中选择本地图片、本地文件等的点击事件  
 \* **@param** fragment 所在页面  
 \* **@param** type 选择的类型 WBSS\_SHOW\_TYPE.SELECT\_PHOTO 选择本地图片  
 \* WBSS\_SHOW\_TYPE.SELECT\_FILE 选择本地文件  
 \* WBSS\_SHOW\_TYPE.SELECT\_NETWORK\_FILE 选择网络文件   
 \* **@param** requestCode 抛出的跳转页面requestCode  
 \* **@param** documentDataName 指定的result携带data键值  
 \*/  
 @Override  
 public void onWbssSelectDocument(Fragment fragment, WBSS\_SHOW\_TYPE type, int requestCode, String documentDataName) {  
 //代码示例  
 if (type == WBSS\_SHOW\_TYPE.SELECT\_PHOTO) { PhotoPicker.builder().showCamera(true).showOriginal(true).limit(1).isEditImg(false).start(RongXinApplicationContext.getContext(), fragment, requestCode);  
 } else if (type == WBSS\_SHOW\_TYPE.SELECT\_FILE) {  
 Intent intent = new Intent(fragment.getContext(), DefaultSelectFileActivity.class);  
 intent.putExtra("dataname", documentDataName);  
 fragment.startActivityForResult(intent,requestCode);  
 }  
 }

/\*\*  
 \* 此方法为适配开发者的自定义选取document模块的返回类型   
 \* 可以不关注 可以不关注 可以不关注  
 \* **@param** type 选择结果的类型 SELECT\_PHOTO 和 SELECT\_FILE已实现，

SELECT\_NETWORK\_FILE未实现  
 \* **@return** YHCWbssSelectResult（RESULT\_TYPE type, String dataName）  
 \* 第一个参数type是设置返回的类型 RESULT\_TYPE.URI\_RESULT Uri类型  
 \* RESULT\_TYPE.STRING\_RESULT String 类型  
 \* 第二个参数dataName是设置返回的键值 相当于替换 onWbssSelectDocument接口

中的 documentDataName默认值  
 \*/  
 @Override  
 public YHCWbssSelectResult onGetDocumentResult(WBSS\_SHOW\_TYPE type) {  
 //代码示例 可以不关注 可以不关注 可以不关注  
 if (type == WBSS\_SHOW\_TYPE.SELECT\_PHOTO) {  
 return new YHCWbssSelectResult(YHCWbssSelectResult.RESULT\_TYPE.URI\_RESULT, "OUT\_PUT");  
 } else if (type == WBSS\_SHOW\_TYPE.SELECT\_FILE) {  
 return new YHCWbssSelectResult

(YHCWbssSelectResult.RESULT\_TYPE.STRING\_RESULT, "select\_file\_result");  
 }  
 return null;  
 }  
};

###### 跳转页面选择document完成后

//DATANAME 为上面onWbssSelectDocument接口回调抛出的documentDataName

Intent intent = new Intent();  
intent.putExtra(DATANAME,path);  
setResult(Activity.RESULT\_OK, intent);  
finish();

##### 2)（方法5）会议信息通知回调

**注意事项**：1、Internal通知为插件内部通知，一般情况开发者无需关注；2、External通知为插件外部处理通知，建议处理，示例代码详见demo

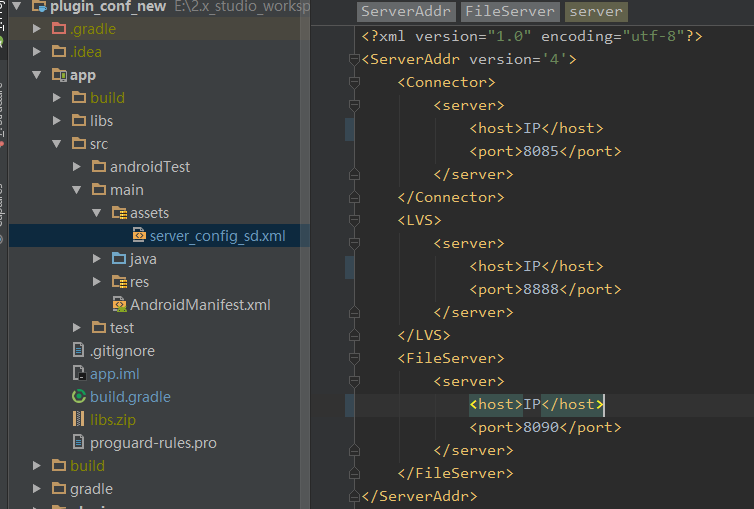
// 此回调为会议信息相关通知   
// internal 为插件内部处理通知,一般情况开发者无需关注  
// external 为插件外部处理通知 建议处理 示例详见demo  
private OnReceiveConfNotifylistener onReceiveConfNotifylistener = new OnReceiveConfNotifylistener() {  
 /\*\*  
 \* 插件外部需要处理的通知   
 \* **@param** notification 会议通知信息  
 \*/  
 @Override  
 public void onExternalDealNotify(ECConferenceNotification notification) {  
 }  
 /\*\*  
 \* 插件内部处理的通知 一般情况开发者无需关注  
 \* **@param** notification 会议通知信息  
 \*/  
 @Override  
 public void onInternalDealNotify(ECConferenceNotification notification) {  
 }  
};

## 三、登录和登出

### 3.1 登录

**注意事项：**

目前需使用私有云，那么服务器的地址书写格式请参考Demo的assets目录下的文件（位置如下图所示）



#### 3.1.1 登录

登录分两种模式：

1）[强制登入](#_1)_任意账号体系登录)

2）[自动登入](#_2)_通讯帐号体系登录)

##### 1) 登入(强制登入/自动登入方式)

两个接口区别是强制登入需要传用户信息，自动登入的是使用上一次的用户信息登入

**强制登入**：一般用于App首次使用填写完用户账号密码之后使用 如下接口

SDKCoreHelper.*login*(builder.build(), "server\_config\_sd.xml");

**自动登入**：一般用于不需要进行用户账号密码填写应用主页地方使用 如下接口

if (AppMgr.*getClientUser*() != null) {  
 LogUtil.*d*(*TAG*, "SDK auto connect...");  
 SDKCoreHelper.*init*(context);  
}

##### 完整登入逻辑代码如下(强制登入):

PluginUser.UserBuilder builder = new PluginUser.UserBuilder(account, name);  
builder.setAppKey("appId");// AppId(私有云使用)  
builder.setAppToken("appToken");// AppToken(私有云使用)  
SDKCoreHelper.*login*(builder.build(),"server\_config\_sd.xml");

#### 3.1.2 获取登录状态

##### 1) 注册获取登录状态的广播接收者

IntentFilter intentFilter = new IntentFilter();  
intentFilter.addAction(SDKCoreHelper.ACTION\_SDK\_CONNECT);  
registerReceiver(mSDKNotifyReceiver, intentFilter);

private BroadcastReceiver mSDKNotifyReceiver = new BroadcastReceiver() {  
 @Override  
 public void onReceive(Context context, Intent intent) {  
 if (SDKCoreHelper.ACTION\_SDK\_CONNECT.equals(intent.getAction())) {  
 if (SDKCoreHelper.isLoginSuccess(intent)) {  
 //登录成功

} else {  
 int error = intent.getIntExtra("error", 0);  
 if (error == SdkErrorCode.CONNECTING) return;  
 dismissDialog();  
 LogUtil.e(TAG, "登入失败[" + error + "]");  
 ToastUtil.showMessage("登入失败[" + error + "]");  
 }  
 }  
 }  
};

##### 2) 注销获取登录状态的广播接收者

@Override  
protected void onDestroy() {  
 super.onDestroy();  
 // 注销广播  
 unregisterReceiver(mSDKNotifyReceiver);  
}

#### 3.1.3 获取异地登录状态

##### 1) 注册获取异地登录状态的广播接收者

IntentFilter intentFilter = new IntentFilter();  
intentFilter.addAction(SDKCoreHelper.*ACTION\_SDK\_CONNECT*);// SDK状态广播  
intentFilter.addAction(RongXInUtils.*ACTION\_KICK\_OFF*);// 账号异地登入广播

private BroadcastReceiver mSDKNotifyReceiver = new BroadcastReceiver() {  
 @Override  
 public void onReceive(Context context, Intent intent) {  
 if (RongXInUtils.ACTION\_KICK\_OFF.equals(intent.getAction())) {  
 //代码示例  
 ToastUtil.showMessage("您的账号被他人登陆，请确定您的账号安全");  
 startActivity(new Intent(MainActivity.this, LoginActivity.class));  
 finish();  
 }  
 }  
};

##### 2) 注销获取异地登录状态的广播接收者

@Override  
protected void onDestroy() {  
 super.onDestroy();  
 // 注销广播  
 unregisterReceiver(mSDKNotifyReceiver);  
}

### 3.2 登出

/\*\*  
 \* 此方法为退出当前登录账户 \*/  
SDKCoreHelper.*logout*();

## 四、可接入界面

### 4.1 进行中会议及预约会议列表（YHCReserveListFragment）

会议插件提供进行会议及预约会议页面（YHCReserveListFragment），开发者可根据需求接入在自己的项目中，使用示例：

/\*\*  
 \* 在开发者指定的activity中复制下面代码，  
 \*/  
Fragment mFragment = Fragment.instantiate(this,   
 YHCReserveListFragment.class.getName(), bundle);  
getSupportFragmentManager()  
 .beginTransaction()  
 .add(R.id.convert\_frame, mFragment)  
 .commit();

效果图如下：



**页面功能：**

1、发起会议；

2、加入会议；

3、查看进行中会议及历史会议；

4、进去会议详情页面；

5、其他功能。

### 4.2 历史会议列表（YHCHistoryListFragment）

如果开发者需要，可接入插件中提供的YHCHistoryListFragment

使用示例：

/\*\*  
 \* 开发者在容器activity中按照一般fragment嵌入得方法使用  
 \*/  
getSupportFragmentManager().beginTransaction()  
 .add(R.id.convert\_frame, Fragment.instantiate(this,  
 YHCHistoryListFragment.class.getName(), null)).commit();

其效果图如下图：



**页面功能：**

1、按照时间线显示历史会议；

2、历史会议包含已取消的会议和已结束的会议；

3、进入会议详情页面；

4、其他功能。

### 4.3 立即会议页面（YHCConfStartAndReserveActivity）

其效果图如下图：



**页面功能：**

1. 立即会议；
2. 其他功能。

### 4.4 预约会议页面（YHCConfStartAndReserveActivity）

其效果图如下图：



**页面功能：**

1. 预约会议；

2、其他功能。

### 4.5 加入会议页面（YHCJoinConfActivity）

其效果图如下图：



**页面功能：**

1、加入会议；

2、其他功能。

### 4.6 会议页面（YHCConfDetailActivity）

使用示例：

/\*\*  
 \* 跳转会议详情页面  
 \* **@param ctx** 上下文  
 \* **@param** confId 会议Id  
 \*/

public void enterConfDetail(final Context ctx, final String confId) {  
 Intent intent = new Intent(ctx, YHCConfDetailActivity.class);  
 intent.putExtra(YHCConfDetailActivity.CONFERENCE\_ID, confId);  
 ctx.startActivity(intent);  
}

其效果图如下图：



**页面功能：**

1、未开始的会议可以邀请成员，编辑议题；

2、查看会议成员；

3、结束的会议可以查看摘要；

4、跳转会议开始页面；

5、分享会议信息；

6、取消未开始的会议；

7、跳转更新会议页面；

8、其他。

## 五、主动调用的方法

### 5.1 发起立即会议

**使用场景：1）**在开发者需要立即发起会议时，可主动调用该方法

///创建插件会议对象  
YHCConfInfo info = new YHCConfInfo();  
//设置会议名字 可选  
info.setConfName(confName);  
//设置他人加入会议是否打开语音 可选  
info.setOpenVoice(!meeting\_mute.isChecked());  
//设置他人加入会议是否打开视频 可选  
info.setOpenVideo(!meeting\_video.isChecked());  
//发起会议  
YHCConferenceMgr.getManager().startConference(“上下文context”，info);

### 5.2 创建预约会议

**使用场景：**开发者若需要创建预约会议，可主动调用该方法

//创建插件会议对象  
YHCConfInfo confInfo = new YHCConfInfo();  
//设置预约会议中成员 可选  
confInfo.setMemberList(YHCConfMemberMeetingMemberList);  
//设置会议名字 可选  
confInfo.setConfName(theme\_tv.getText().toString().trim());  
//设置预约会议的开始时间 时间戳 必选  
confInfo.setReserStartTime(appointmentTime);  
//设置会议的进行时长 单位分钟 可选  
confInfo.setDuration(30);  
//设置他人加入会议是否打开语音 可选  
confInfo.setOpenVoice(!enter\_voice.isChecked());  
//设置他人加入会议是否打开视频 可选  
confInfo.setOpenVideo(!enter\_video.isChecked());  
//创建预约会议  
YHCConferenceMgr.*getManager*().reserveConference(confInfo, new OnReserResultListener() {  
 @Override  
 public void onCreateResult(int resultCode, ECConferenceInfo info) {  
 confirm.setEnabled(true);  
 if (resultCode == 200) {  
 //会议创建成功 开发者处理对应的逻辑  
 }  
 }  
});

### 5.3 加入会议

**使用场景：**开发者若需要加入会议，可主动调用该方法

/\*\*  
 \* **@param** String 会议号 必选  
 \* **@param** String 自己在会议中的名字 可选  
 \* **@param** boolean openVoice 自己加入会议后是否打开语音  
 \* **@param** boolean openVideo 自己加入会议后是否打开视频  
 \*/  
YHCConferenceMgr.getManager().joinConference("会议号","昵称",true,true);

### 5.4 更新会议

**使用场景：**开发者需要更新会议时，可调用此方法

//代码示例  
YHCConfInfo info = new YHCConfInfo();

//设置会议号 必选  
info.setMeetingNo("会议号");

//设置会议的名字 可选  
info.setConfName("更新后的会议名字");  
YHCConferenceMgr.*getManager*().updateConference(info, new OnReserResultListener() {  
 @Override  
 public void onCreateResult(int resultCode, ECConferenceInfo info) {  
 //代码示例  
 if (resultCode == 200) {  
 //更新成功  
 }  
 }  
});

### 5.5 获取会议相关设置

**使用场景：**开发者需要获取插件相关信息时，可调用此方法

/\*\*  
 \* **@ConferenceSetting** 包含最大人数、最小人数、接入号的集合、appid  
 \*/  
YHCConferenceMgr.*getManager*().getConfSetting(new OnConfSettingListener(){  
 @Override  
 public void onConfSetting(int resultCode, ConferenceSetting setting) {  
 //代码示例  
 if (resultCode == 200) {  
 LogUtil.d(setting.toString());  
 }  
 }  
});

### 5.6 会议插件全局设置

**作用位置：（此图为示例，实际页面应是开发者的设置页面）**



**使用场景：**开发者应把这些设置放置在自己的某个页面内来进行全局设置会议插件，根据下面提供的方法来进行设置，并根据[5.7](#_5.7_会议插件全局设置状态的获取)来获取这些设置的状态，更新开发者的这些开关状态。true打开，false关闭

#### 1) 设置自己加入会议是否关闭语音

/\*\*  
 \* 加入会议后自己是否开启语音  
 \* **@param** boolean   
 \*/  
YHCConferenceMgr.getManager().setSelfConfCloseVoice(selfVoice.isChecked());

#### 2) 设置自己加入会议是否关闭视频

/\*\*  
 \* 加入会议后自己是否开启视频  
 \* **@param** boolean  
 \*/  
YHCConferenceMgr.getManager().setSelfConfCloseVideo(selfVideo.isChecked());

#### 3) 设置他人加入会议是否关闭语音

/\*\*  
 \* 加入会议后他人是否开启语音  
 \* **@param** boolean  
 \*/  
YHCConferenceMgr.getManager().setOtherConfCloseVoice(otherVoice.isChecked());

#### 4) 设置他人加入会议是否关闭视频

/\*\*  
 \* 加入会议后他人是否开启视频  
 \* **@param** boolean  
 \*/  
YHCConferenceMgr.getManager().setOtherConfCloseVideo(otherVideo.isChecked());

### 5.7 会议插件全局设置状态的获取

**使用场景：**开发者若需要获取下列设置的状态来配置自己的设置页面，则可以调用对应的方法对自己设置页面的控件更新

#### 1) 获取自己加入会议是否关闭语音的设置状态

/\*\*  
 \* 获取加入会议后自己是否关闭语音 默认false  
 \*/  
boolean selfConfCloseVoice = YHCConferenceMgr.getManager().getSelfConfCloseVoice();

#### 2) 获取自己加入会议是否关闭视频的设置状态

/\*\*  
 \* 获取加入会议后自己是否关闭视频 默认false  
 \*/  
boolean selfConfCloseVideo = YHCConferenceMgr.getManager().getSelfConfCloseVideo();

#### 3) 获取他人加入会议是否关闭语音的设置状态

/\*\*  
 \* 获取加入会议后他人是否关闭语音 默认false  
 \*/  
boolean otherConfCloseVoice = YHCConferenceMgr.getManager().getOtherConfCloseVoice();

#### 4) 获取他人加入会议是否关闭视频的设置状态

/\*\*  
 \* 获取加入会议后他人是否关闭视频 默认false  
 \*/  
boolean otherConfCloseVideo = YHCConferenceMgr.getManager().getOtherConfCloseVideo();