Android白名单app广告模块api文档

（V1.0）

目录

[一、 版本变更记录 3](#_Toc489535305)

[二、aar配置方式 4](#_Toc489535306)

[1、 导入aar文件 4](#_Toc489535307)

[2、 引入其他依赖 4](#_Toc489535308)

[3、 AndroidManifest.xml配置 4](#_Toc489535309)

[3、 初始化应用信息 5](#_Toc489535310)

[4、 混淆配置 5](#_Toc489535311)

[三、 grade配置方式 6](#_Toc489535312)

[1、 gradle依赖配置 6](#_Toc489535313)

[四、 广告显示调用 7](#_Toc489535314)

[1、 插屏广告 7](#_Toc489535315)

[1.1、 加载插屏广告数据以及展示 7](#_Toc489535316)

[2、 banner广告 7](#_Toc489535317)

[2.1、 展示banner广告 7](#_Toc489535318)

[3、 信息流广告 9](#_Toc489535319)

[3.1、 信息流广告 9](#_Toc489535320)

[4、 开屏广告 11](#_Toc489535321)

[4.1、 开屏广告调用 11](#_Toc489535322)

[五、其他开关 15](#_Toc489535323)

[1、 广告开关 15](#_Toc489535324)

# 一、 版本变更记录

|  |  |  |
| --- | --- | --- |
| 版本号 | 修改人 | 修改时间 |
| V1.0 | 吴限亮 | 2017/8/2 |
|  |  |  |
|  |  |  |
|  |  |  |

# 二、aar配置方式

## 1、 导入aar文件

使用本广告模块，需要导入两个aar，一个是基础网络模块，一个是广告模块。两个aar需放入需要项目的工程lib目录下，并在其主工程的build.gradle中进行如下配：

repositories {  
 flatDir {  
 dirs 'libs'  
 }

}

需要引入aar的模块中的build.gradle则编译引入的aar：

compile(name:'adv.aar', ext:'aar')

compile(name:'network.aar', ext:'aar')

compile(name:'rounded\_img.aar', ext:'aar')

## 2、 引入其他依赖

compile 'com.android.support:appcompat-v7:25.3.1'

compile 'com.android.support:support-v4:25.3.1'  
compile 'io.reactivex.rxjava2:rxandroid:2.0.1'

compile 'io.reactivex.rxjava2:rxjava:2.1.0'  
compile 'com.squareup.retrofit2:retrofit:2.3.0'  
compile 'com.squareup.retrofit2:adapter-rxjava2:2.3.0'  
compile 'com.google.code.gson:gson:2.8.1'  
compile 'com.squareup.okhttp3:okhttp:3.8.0'  
compile 'com.squareup.okhttp3:logging-interceptor:3.8.0'

compile 'com.github.bumptech.glide:glide:4.0.0-RC0'  
annotationProcessor 'com.github.bumptech.glide:compiler:4.0.0-RC0'

## 3、 AndroidManifest.xml配置

在manifest中添加以下权限：

<uses-permission android:name="android.permission.INTERNET" />  
<uses-permission android:name="android.permission.READ\_EXTERNAL\_STORAGE" />  
<uses-permission android:name="android.permission.WRITE\_EXTERNAL\_STORAGE" />

<uses-permission android:name="android.permission.ACCESS\_NETWORK\_STATE" />  
<uses-permission android:name="android.permission.ACCESS\_WIFI\_STATE" />  
<uses-permission android:name="android.permission.READ\_PHONE\_STATE" />

在manifest中添加以下activity：

<activity  
 android:name="com.jmgzs.lib.adv.ui.AdvWebViewActivity"  
 android:configChanges="fontScale"  
 android:hardwareAccelerated="false"  
 android:screenOrientation="portrait" />

## 初始化应用信息

在应用Application子类（继承Application类）中的onCreate()方法中调用：

AdvUtil.getInstance().init (this, "应用缓存目录");

注意：不初始化就显示广告会抛出异常

## 4、 混淆配置

**-keep** class com.jmgzs.lib.adv.bean.\*\* { \* ;}

**-keep** class \* extends android.app.Activity {  
 \*;  
}

# 三、 grade配置方式

除了引入依赖的方式不同之外，其他的配置方式与上面aar的方式相同

## 1、 gradle依赖配置

主工程中的build.gradle中添加如下配置：

repositories {  
 maven{ url "http:// maven.weimob.com/content/repositories/snapshots/"}  
}

在需要引入广告的module中的build.gradle中添加如下配置：

dependencies {

compile ("com.jmgzs.lib.adv:adv:1.0.0-SNAPSHOT@aar"){transitive=true}

}

# 四、 广告显示调用

## 1、 插屏广告

### 1.1、 加载插屏广告数据以及展示

初始化后调用：

AdvUtil.getInstance().showInsertAdv(Context context, int templateId, IAdvStatusCallback callback);

参数说明：

context：上下文引用，即当前Activity引用；

templateId：需要显示的广告规格对应的广告位id；

callback：广告状态变化回调，比如广告关闭；

## 2、 banner广告

### 2.1、 展示banner广告

###### 1) 初始化后调用，以html方式展示，并回调方式返回html，由调用方处理展示：

AdvUtil.getInstance().showBannerAdv(Context context, int templateId, boolean isUseCache, boolean isIFrame, int width, IAdvHtmlCallback callback);

参数说明：

context：上下文引用，即当前Activity引用；

templateId：需要显示的广告规格对应的广告位id；

isUseCache：是否使用缓存，true则会直接读取缓存回调，然后异步请求缓存

isIFrame：是否在IFrame标签中展示广告，true则会调整广告html的内容适应IFrame标签

width：广告展示的宽度，通常为屏幕宽度，也可以自定义

callback：广告html内容回调，可以直接放入页面中展示；

###### 2) 初始化后调用，以html方式展示，传入webview，自动展示广告，回调用于处理广告关闭等广告事件：

AdvUtil.getInstance().showBannerAdv(Context context, int templateId, boolean isUseCache, boolean isIFrame, WebView wv, int width, IAdvStatusCallback callback)

参数说明：

context：上下文引用，即当前Activity引用；

templateId：需要显示的广告规格对应的广告位id；

isUseCache：是否使用缓存，true则会直接读取缓存回调，然后异步请求缓存

isIFrame：是否在IFrame标签中展示广告，true则会调整广告html的内容适应IFrame标签

wv：显示广告的webview，会自动进行相关配置，及显示

width：广告展示的宽度，通常为屏幕宽度，也可以自定义

callback：广告状态变化回调，比如广告关闭；

###### 3) 初始化后调用，以原生方式展示，以回调方式返回获取到的广告相关元素：

AdvUtil.getInstance().showBannerAdv(Context context, int templateId, boolean isUseCache, IAdvResponseCallback callback) {

参数说明：

context：上下文引用，即当前Activity引用；

templateId：需要显示的广告规格对应的广告位id；

isUseCache：是否使用缓存，true则会直接读取缓存回调，然后异步请求缓存

callback：广告数据对象回调，由调用方自己处理展示；

## 3、 信息流广告

### 3.1、 信息流广告

###### 1) 初始化后调用，以html方式展示，并回调方式返回html，由调用方处理展示：

AdvUtil.getInstance().showInfoAdv(Context context, int templateId, boolean isUseCache, boolean isIFrame, int width, IAdvHtmlCallback callback)

参数说明：

context：上下文引用，即当前Activity引用；

templateId：需要显示的广告规格对应的广告位id；

isUseCache：是否使用缓存，true则会直接读取缓存回调，然后异步请求缓存

isIFrame：是否在IFrame标签中展示广告，true则会调整广告html的内容适应IFrame标签

width：广告展示的宽度，通常为屏幕宽度，也可以自定义

callback：广告html内容回调，可以直接放入页面中展示；

###### 2) 初始化后调用，以html方式展示，传入webview，自动展示广告，回调用于处理广告关闭等广告事件：

AdvUtil.getInstance().showInfoAdv(Context context, int templateId, boolean isUseCache, boolean isIFrame, WebView wv, int width, IAdvStatusCallback callback)

参数说明：

context：上下文引用，即当前Activity引用；

templateId：需要显示的广告规格对应的广告位id；

isUseCache：是否使用缓存，true则会直接读取缓存回调，然后异步请求缓存

isIFrame：是否在IFrame标签中展示广告，true则会调整广告html的内容适应IFrame标签

wv：显示广告的webview，会自动进行相关配置，及显示

width：广告展示的宽度，通常为屏幕宽度，也可以自定义

callback：广告状态变化回调，比如广告关闭；

###### 3) 初始化后调用，以html方式展示，并回调方式返回html，由调用方处理展示：

AdvUtil.getInstance().showInfoAdv(Context context, int templateId, final boolean isUseCache, IAdvResponseCallback callback)

参数说明：

context：上下文引用，即当前Activity引用；

templateId：需要显示的广告规格对应的广告位id；

isUseCache：是否使用缓存，true则会直接读取缓存回调，然后异步请求缓存

callback：广告数据对象回调，由调用方自己处理展示；

## 4、 开屏广告

### 4.1、 开屏广告调用

开屏广告有两种集成方式：

1. 使用Intent调用，也就是说不作为应用的启动界面，由调用方手动调用启动。集成方式如下：

首先要在AndroidManifest.xml中注册com.jmgzs.lib.adv.ui. AdvSplashActivity

<activity android:name="com.jmgzs.lib.adv.ui. AdvSplashActivity"  
 android:screenOrientation="portrait"/>

然后在需要启动开屏广告处，根据需求调用showOpenAdv：

1) 以随机规格方式获取缓存后返回广告数据，不执行自动跳转：

AdvUtil.getInstance().requestOpenAdv (Activity context, IRequestCallBack<AdvResponseBean.AdInfoBean> callback)

参数说明：

context：上下文引用，即当前Activity引用；

callback：开屏广告数据对象回调，用于调用者自行跳转开屏界面；

2) 获取到缓存后自动跳转开屏界面

AdvUtil.getInstance().requestOpenAdv(Context context, List<Integer> templateIds, int logoResId, String appInfo, String className, String tempDir)

参数说明：

context：上下文引用，即当前Activity引用；

templateIds：需要缓存的广告规格对应的广告位id列表；

logoResId：logo图片的资源id

appInfo：应用信息，通常是一小段简介

className：开屏广告后跳转的应用界面的class全限定名

tempDir：该字段无用，新版本会去除

3) 获取到缓存后，根据回调对象，判断是否自动跳转开屏界面：

AdvUtil.getInstance().requestOpenAdv(Context context, List<Integer> templateIds, int logoResId, String appInfo, String className, String tempDir, IAdvResponseCallback callback)

参数说明：

context：上下文引用，即当前Activity引用；

templateIds：需要缓存的广告规格对应的广告位id列表；

logoResId：logo图片的资源id

appInfo：应用信息，通常是一小段简介

className：开屏广告后跳转的应用界面的class全限定名

tempDir：该字段无用，新版本会去除

callback：广告数据对象回调，如果为null则会自动跳转至默认开屏界面，否则回调广告数据对象；

2. 作为应用启动界面，在调用者的应用开始时自动调用。集成方式如下：

首先新建一个类继承com.jmgzs.lib.adv.ui.AdvSplashActivity，比如com.weimob.mjsdk.ui.SplashActivity，然后在

AndroidManifest.xml中注册这个Activity，并作为应用入口（如果需要透明状态栏样式可以在标签中设置）

<activity  
 android:name="com.weimob.mjsdk.ui.SplashActivity"  
 android:screenOrientation="portrait">  
 <intent-filter>  
 <action android:name="android.intent.action.MAIN" />  
  
 <category android:name="android.intent.category.LAUNCHER" />  
 </intent-filter>  
 </activity>

至于跳转目标等参数设置可以在继承的activity中重写对应方法：

1）该方法无用，新版本会废弃

String getTempDir()

2）设置跳转界面

String getActivityName()

返回值为启动开屏结束后需要跳转的界面的完整类名。

3）设置应用简介

String getAppInfo()

返回值为启动开屏广告界面的底部应用简介的文本

4）设置半屏广告界面的logo图片资源id

int getLogoResId()

返回值为启动半屏广告界面的必要参数，底部应用logo的图片资源id，如：R.drawable.logo

# 五、其他开关

## 1、 广告开关

AdvUtil. setAdvOpen(boolean isOpen)

设置广告开关，true为打开广告，false为关闭。默认关闭

AdvUtil. isOpenAdv()

返回广告是否打开。