**OPPOSDK对接文档**

V2.6.5

目录

[1 SDK嵌入 1](#_Toc31039)

[1.1 步骤1：添加SDK到工程中 1](#_Toc30653)

[1.2 步骤2: 修改AndroidManifest.xml文件 1](#_Toc7191)

[1.3 步骤3: 初始化SDK 5](#_Toc16121)

[1.3.1 MobAdManager主要API 5](#_Toc6192)

[1.4 步骤４:释放 SDK资源 6](#_Toc12771)

[2 代码混肴 7](#_Toc29688)

[3 接入代码 7](#_Toc17055)

[3.1 闪屏广告 7](#_Toc4264)

[3.1.1 广告接入代码示例 7](#_Toc20102)

[3.1.2 闪屏广告主要API 14](#_Toc23342)

[3.2 Banner广告 15](#_Toc26589)

[3.1.1广告接入代码示例 15](#_Toc22919)

[3.1.2 Banner广告主要API 17](#_Toc28243)

[3.3 插屏广告 18](#_Toc8131)

[3.3.1 广告接入代码示例 18](#_Toc29801)

[3.3.2 插屏广告主要API 20](#_Toc21062)

[3.4 原生广告 21](#_Toc8500)

[3.4.1 广告接入代码示例 21](#_Toc18185)

[3.4.2 原生广告主要API 24](#_Toc31003)

[3.5 原生模板广告 26](#_Toc9701)

[3.5.1 广告接入代码示例 26](#_Toc3445)

[3.5.2 原生模板广告主要API 31](#_Toc10285)

[3.6 激励视频广告 33](#_Toc21050)

[3.6.1 广告接入代码示例 33](#_Toc23076)

[3.6.2 激励视频广告主要API 37](#_Toc4646)

[4 SDK相关问题排查 38](#_Toc5391)

[5 错误码 38](#_Toc31993)

# SDK嵌入

## 步骤1：添加SDK到工程中

1. 请在工程文件根目录下创建一个名为 libs 的子目录，并将OPPO SDK 的 JAR 包拷贝到 libs 目录下。

注意：如果开发者使用的是Unity开发环境，可以选择以下两种方式进行打包apk：

1. 采用Eclipse打包的方式：不要采用Unity默认的打包方式。
2. 采用Unity默认的打包方式：需要将我们提供的SDK jar文件里面的assets目录下的文件拷贝到对应项目工程的plugins/android目录下，同时引用我们的SDK jar文件。

如有疑问，直接联系OPPO技术人员解决。

1. 应用开发者需自己添加最新的support-v4.jar, support-v4.jar是SDK运行必须的Jar包。
2. 注意：因为我们的SDK Jar里面集成了第三方的okio、wire-runtime两个jar的代码；如果应用自己已经集成了这两个Jar，就会导致代码冲突；出现这种情况请联系OPPO技术人员为你提供一个分离okio、wire-runtime两个jar的SDK。

## 步骤2: 修改AndroidManifest.xml文件

添加权限声明：

<!--SDK 必选权限配置开始-->  
<uses-permission android:name="android.permission.ACCESS\_NETWORK\_STATE" />  
<uses-permission android:name="android.permission.ACCESS\_WIFI\_STATE" />  
<uses-permission android:name="android.permission.INTERNET" />  
<uses-permission android:name="android.permission.READ\_PHONE\_STATE" />  
<uses-permission android:name="android.permission.WRITE\_EXTERNAL\_STORAGE" />  
<uses-permission android:name="android.permission.MOUNT\_UNMOUNT\_FILESYSTEMS" />  
<uses-permission android:name="android.permission.READ\_CALENDAR" />  
<uses-permission android:name="android.permission.WRITE\_CALENDAR" />  
<!--SDK 必选权限配置结束-->  
  
<!--SDK 可选权限配置开始；建议应用配置定位权限，可以提升应用的广告收益-->  
<uses-permission android:name="android.permission.ACCESS\_COARSE\_LOCATION" />  
<!--如果应用需要精准定位的话加上该权限-->  
<uses-permission android:name="android.permission.ACCESS\_FINE\_LOCATION" />  
<!--SDK 可选权限配置结束-->

添加组件申明：

<!--SDK 必选组件配置开始-->  
<activity android:name="com.oppo.mobad.activity.AdActivity" />  
  
<service android:name="com.oppo.mobad.service.AdService" />  
<!--add 2018-05-01-->  
<!--如果奖励视频不需要随机器旋转⽽旋转的请将VideoActivity设置android:screenOrientation="sensor"修改横屏或竖屏-->  
<activity  
 android:name="com.oppo.mobad.activity.VideoActivity"  
 android:configChanges="keyboardHidden|orientation|screenSize"  
 android:screenOrientation="sensor" />  
<!--SDK 必选组件配置结束-->

注意：

1. 以上可选权限应用可根据自身情况自行决定是否配置，配置后，可以提升广告投放效果及收益。

2. 如果您打包App时的targetSdkVersion >= 23：请在先获取到SDK要求的所有权限，然后再调用SDK的广告接口。否则OPPO 广告SDK将无法工作，我们建议您在App启动时就去获取OPPO广告SDK需要的权限，Demo工程中的SplashActivity也提供了基本的权限处理示例代码供开发者参考。

3. READ\_PHONE\_STATE、WRITE\_EXTERNAL\_STORAGE 两个权限是必须权限，没有这两个权限SDK无法正常获得广告。

4.如果接入激励视频广告，必须在AndroidMainfest.xml里注册VideoActivity：

<!--如果奖励视频不需要随机器旋转⽽旋转的请将VideoActivity设置android:screenOrientation="sensor"修改横屏或竖屏-->  
<activity  
 android:name="com.oppo.mobad.activity.VideoActivity"  
 android:configChanges="keyboardHidden|orientation|screenSize"  
 android:screenOrientation="sensor" />

权限处理的示例代码如下：

/\*\*  
 \* 申请SDK运行需要的权限  
 \* 注意：READ\_PHONE\_STATE、WRITE\_EXTERNAL\_STORAGE 两个权限是必须权限，没有这两个权限SDK无法正常获得广告。  
 \* WRITE\_CALENDAR、ACCESS\_FINE\_LOCATION 是两个可选权限；没有不影响SDK获取广告；但是如果应用申请到该权限，会显著提升应用的广告收益。  
 \*/  
 private void checkAndRequestPermissions() {  
 /\*\*  
 \* READ\_PHONE\_STATE、WRITE\_EXTERNAL\_STORAGE 两个权限是必须权限，没有这两个权限SDK无法正常获得广告。  
 \*/  
 if (PackageManager.PERMISSION\_GRANTED != ActivityCompat.checkSelfPermission(this, Manifest.permission.READ\_PHONE\_STATE)) {  
 mNeedRequestPMSList.add(Manifest.permission.READ\_PHONE\_STATE);  
 }  
 if (PackageManager.PERMISSION\_GRANTED != ActivityCompat.checkSelfPermission(this, Manifest.permission.WRITE\_EXTERNAL\_STORAGE)) {  
 mNeedRequestPMSList.add(Manifest.permission.WRITE\_EXTERNAL\_STORAGE);  
 }  
  
 /\*\*  
 \* WRITE\_CALENDAR、ACCESS\_FINE\_LOCATION 是两个可选权限；没有不影响SDK获取广告；但是如果应用申请到该权限，会显著提升应用的广告收益。  
 \*/  
 if (PackageManager.PERMISSION\_GRANTED != ActivityCompat.checkSelfPermission(this, Manifest.permission.WRITE\_CALENDAR)) {  
 mNeedRequestPMSList.add(Manifest.permission.WRITE\_CALENDAR);  
 }  
 if (PackageManager.PERMISSION\_GRANTED != ActivityCompat.checkSelfPermission(this, Manifest.permission.ACCESS\_FINE\_LOCATION)) {  
 mNeedRequestPMSList.add(Manifest.permission.ACCESS\_FINE\_LOCATION);  
 }  
 //  
 if (0 == mNeedRequestPMSList.size()) {  
 /\*\*  
 \* 权限都已经有了，那么直接调用SDK请求广告。  
 \*/  
 fetchSplashAd();  
 } else {  
 /\*\*  
 \* 有权限需要申请，主动申请。  
 \*/  
 String[] temp = new String[mNeedRequestPMSList.size()];  
 mNeedRequestPMSList.toArray(temp);  
 ActivityCompat.requestPermissions(this, temp, REQUEST\_PERMISSIONS\_CODE);  
 }  
 }  
  
 /\*\*  
 \* 处理权限申请的结果  
 \*  
 \* @param requestCode  
 \* @param permissions  
 \* @param grantResults  
 \*/  
 @Override  
 public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull int[] grantResults) {  
 switch (requestCode) {  
 /\*\*  
 \*处理SDK申请权限的结果。  
 \*/  
 case REQUEST\_PERMISSIONS\_CODE:  
 if (hasNecessaryPMSGranted()) {  
 /\*\*  
 \* 应用已经获得SDK运行必须的READ\_PHONE\_STATE、WRITE\_EXTERNAL\_STORAGE两个权限，直接请求广告。  
 \*/  
 fetchSplashAd();  
 } else {  
 /\*\*  
 \* 如果用户没有授权，那么应该说明意图，引导用户去设置里面授权。  
 \*/  
 Toast.makeText(this, "应用缺少SDK运行必须的READ\_PHONE\_STATE、WRITE\_EXTERNAL\_STORAGE两个权限！请点击\"应用权限\"，打开所需要的权限。", Toast.LENGTH\_LONG).show();  
 Intent intent = new Intent(Settings.ACTION\_APPLICATION\_DETAILS\_SETTINGS);  
 intent.setData(Uri.parse("package:" + getPackageName()));  
 startActivity(intent);  
 finish();  
 }  
 break;  
 default:  
 break;  
 }  
 }  
  
 /\*\*  
 \* 判断应用是否已经获得SDK运行必须的READ\_PHONE\_STATE、WRITE\_EXTERNAL\_STORAGE两个权限。  
 \* @return  
 \*/  
 private boolean hasNecessaryPMSGranted() {  
 if (PackageManager.PERMISSION\_GRANTED == ActivityCompat.checkSelfPermission(this, Manifest.permission.READ\_PHONE\_STATE)) {  
 if (PackageManager.PERMISSION\_GRANTED == ActivityCompat.checkSelfPermission(this, Manifest.permission.WRITE\_EXTERNAL\_STORAGE)) {  
 return true;  
 }  
 }  
 return false;  
 }  
}

接入任意广告都需要在XML中添加以下声明：

<!--SDK 必选组件配置开始-->  
<activity android:name="com.oppo.mobad.activity.AdActivity" />  
  
<service android:name="com.oppo.mobad.service.AdService" />  
<!--SDK 必选组件配置结束-->

## 步骤3: 初始化SDK

在应用的Application或者应用的入口/主Activity的onCreate()方法里面调用下面的函数：

InitParams initParams = new InitParams.Builder()

.setDebug(true)

//true打开SDK日志，当应用发布Release版本时，必须注释掉这行代码的调用，或者设为false  
.build();  
/\*\*  
 \* 调用这行代码初始化广告SDK  
 \*/  
MobAdManager.getInstance().init(this, Constants.APP\_ID, initParams);

### MobAdManager主要API

* **com.oppo.mobad.api.MobAdManager:**

| **方法名** | **方法介绍** |
| --- | --- |
| getInstance() | 获取MobAdManager单例对象 |
| init(Context context, String appId) throws NullPointerException | 初始化SDK；该方法必须在初始化广告对象之前调用；建议在Application的onCreate中初始化，该方法只需要调用一次即可 |
| public void init(Context context, String appId, IInitListener iInitListener) throws NullPointerException | 初始化SDK；该方法必须在初始化广告对象之前调用；建议在Application的onCreate中初始化，该方法只需要调用一次即可 |
| public void init(Context context, String appId, IInitListener iInitListener) throws NullPointerException | 初始化SDK；该方法必须在初始化广告对象之前调用；建议在Application的onCreate中初始化，该方法只需要调用一次即可 |
| public void init(Context context, String appId, InitParams initParams, IInitListener iInitListener) throws NullPointerException | 初始化SDK；该方法必须在初始化广告对象之前调用；建议在Application的onCreate中初始化，该方法只需要调用一次即可 |
| getSdkVerCode() | 获取SDK版本号 |
| getSdkVerName() | 获取SDK版本名称 |
| exit(Context context) throws NullPointerException | 退出应用时调用、释放SDK资源 |
| isSupportedMobile() | 判断手机机型是否支持联盟SDK |

## 步骤４:释放 SDK资源

在应用退出时调用下面的函数来释放SDK资源：

/\*\*  
 \* 在你的应用程序进程退出时，调用该方法释放SDK 资源  
 \* \*/  
MobAdManager.getInstance().exit(this);

# 代码混肴

如果您需要使用proguard混淆代码，需确保不要混淆SDK的代码。

请在proguard.cfg文件(或其它混淆文件)尾部添加如下配置:

#oppo sdk

-keep class com.oppo.\*\* {

public protected \*;

}

-keep class okio.\*\*{ \*; }

-keep class com.squareup.wire.\*\*{ \*; }

-keep public class \* extends com.squareup.wire.\*\*{ \*; }

# Keep methods with Wire annotations (e.g. @ProtoField)

-keepclassmembers class \*\* {

@com.squareup.wire.ProtoField public \*;

@com.squareup.wire.ProtoEnum public \*;

}

-keep public class com.cdo.oaps.base.\*\*{ \*; }

-keepattributes \*Annotation\*

-keepattributes \*JavascriptInterface\*

#support-v4

-keep class android.support.v4.\*\* { \*; }

注意：SDK代码被混淆后会导致广告无法展现或者其它异常

# 接入代码

## 闪屏广告

### 广告接入代码示例

(详细内容请参考压缩包中的代码示例)

public class SplashActivity extends Activity implements ISplashAdListener {  
 private static final String TAG = "SplashActivity";  
 private static final int REQUEST\_PERMISSIONS\_CODE = 100;  
 /\*\*  
 \* 从请求广告到广告展示出来最大耗时时间，只能在[3000,5000]ms之内。  
 \*/  
 private static final int FETCH\_TIME\_OUT = 3000;  
 //  
 /\*\*  
 \* 闪屏广告是半屏广告，广告下面半屏是:应用ICON+应用标题+应用描述，  
 \* 应用标题和应用描述由应用在SplashAd构造函数里传入，  
 \* 应用标题限制最多不超过个 8 个汉字，应用描述限制不超过 13 个汉字。  
 \*/  
 private static final String APP\_TITLE = "OPPO广告联盟";  
 private static final String APP\_DESC = "让天下没有难做的广告";  
 //  
 private List<String> mNeedRequestPMSList = new ArrayList<>();  
 //  
 private SplashAd mSplashAd;  
 /\*\*  
 \* 判断是否可以立刻跳转应用主页面。  
 \*/  
 private boolean mCanJump = false;  
  
  
 @Override  
 protected void onCreate(@Nullable Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 //  
 setContentView(R.layout.activity\_splash);  
  
 if (Build.VERSION.SDK\_INT > Build.VERSION\_CODES.LOLLIPOP\_MR1) {  
 /\*\*  
 \* 如果你的targetSDKVersion >= 23，就要主动申请好权限。如果您的App没有适配到Android6.0（即targetSDKVersion < 23），那么只需要在这里直接调用fetchSplashAd方法。  
 \*  
 \*/  
 checkAndRequestPermissions();  
 } else {  
 /\*\*  
 \* 如果是Android6.0以下的机器，默认在安装时获得了所有权限，可以直接调用SDK。  
 \*/  
 fetchSplashAd();  
 }  
 }  
  
 private void fetchSplashAd() {  
 try {  
 /\*\*  
 \* SplashAd初始化参数、这里可以设置获取广告最大超时时间，  
 \* 广告下面半屏的应用标题+应用描述  
 \* 注意：应用标题和应用描述是必传字段，不传将抛出异常  
 \*/  
 SplashAdParams splashAdParams = new SplashAdParams.Builder()  
 .setFetchTimeout(FETCH\_TIME\_OUT)  
 .setTitle(APP\_TITLE)  
 .setDesc(APP\_DESC)  
 .build();  
 /\*\*  
 \* 构造SplashAd对象  
 \* 注意：构造函数传入的几个形参都不能为空，否则将抛出NullPointerException异常。  
 \*/  
 mSplashAd = new SplashAd(this, Constants.SPLASH\_POS\_ID, this, splashAdParams);  
 } catch (Exception e) {  
 Log.w(TAG, "", e);  
 /\*\*  
 \* 出错，直接finish(),跳转应用主页面。  
 \*/  
 Intent intent = new Intent(this, MainActivity.class);  
 startActivity(intent);  
 finish();  
 }  
 }  
  
 /\*\*  
 \* 结束闪屏页面，跳转主页面。  
 \*/  
 private void next() {  
 if (mCanJump) {  
 Intent intent = new Intent(this, MainActivity.class);  
 startActivity(intent);  
 finish();  
 } else {  
 mCanJump = true;  
 }  
 }  
  
 @Override  
 public void onAdShow() {  
 Log.d(TAG, "onAdShow");  
 }  
  
 @Override  
 public void onAdFailed(String errMsg) {  
 /\*\*  
 \* 如果加载广告失败，直接finish(),跳转应用主页面。  
 \*/  
 Log.d(TAG, "onAdFailed errMsg=" + (null != errMsg ? errMsg : "null"));  
 Intent intent = new Intent(this, MainActivity.class);  
 startActivity(intent);  
 finish();  
 }  
  
 @Override  
 public void onAdClick() {  
 Log.d(TAG, "onAdClick");  
 }  
  
 @Override  
 public void onAdDismissed() {  
 /\*\*  
 \*广告播放完毕或者用户点击“跳过”按钮，跳转应用主页面。  
 \*/  
 Log.d(TAG, "onAdDismissed");  
 next();  
 }  
  
 @Override  
 protected void onResume() {  
 super.onResume();  
 /\*\*  
 \* 这里包含对于点击闪屏广告以后、然后返回闪屏广告页面立刻跳转应用主页面的处理。  
 \*/  
 if (mCanJump) {  
 next();  
 }  
 mCanJump = true;  
 }  
  
 @Override  
 protected void onPause() {  
 super.onPause();  
 /\*\*  
 \* 这里包含对于点击闪屏广告以后、然后返回闪屏广告页面立刻跳转应用主页面的处理。  
 \*/  
 mCanJump = false;  
 }  
  
 @Override  
 protected void onDestroy() {  
 if (null != mSplashAd) {  
 mSplashAd.destroyAd();  
 }  
 super.onDestroy();  
 }  
  
 /\*\*  
 \* 开屏页一定要禁止用户对返回按钮的控制，否则将可能导致用户手动退出了App而广告无法正常曝光和计费。  
 \*  
 \* @param keyCode  
 \* @param event  
 \* @return  
 \*/  
 @Override  
 public boolean onKeyDown(int keyCode, KeyEvent event) {  
 if (keyCode == KeyEvent.KEYCODE\_BACK || keyCode == KeyEvent.KEYCODE\_HOME) {  
 return true;  
 }  
 return super.onKeyDown(keyCode, event);  
 }  
  
 /\*\*  
 \* 申请SDK运行需要的权限  
 \* 注意：READ\_PHONE\_STATE、WRITE\_EXTERNAL\_STORAGE 两个权限是必须权限，没有这两个权限SDK无法正常获得广告。  
 \* WRITE\_CALENDAR、ACCESS\_FINE\_LOCATION 是两个可选权限；没有不影响SDK获取广告；但是如果应用申请到该权限，会显著提升应用的广告收益。  
 \*/  
 private void checkAndRequestPermissions() {  
 /\*\*  
 \* READ\_PHONE\_STATE、WRITE\_EXTERNAL\_STORAGE 两个权限是必须权限，没有这两个权限SDK无法正常获得广告。  
 \*/  
 if (PackageManager.PERMISSION\_GRANTED != ActivityCompat.checkSelfPermission(this, Manifest.permission.READ\_PHONE\_STATE)) {  
 mNeedRequestPMSList.add(Manifest.permission.READ\_PHONE\_STATE);  
 }  
 if (PackageManager.PERMISSION\_GRANTED != ActivityCompat.checkSelfPermission(this, Manifest.permission.WRITE\_EXTERNAL\_STORAGE)) {  
 mNeedRequestPMSList.add(Manifest.permission.WRITE\_EXTERNAL\_STORAGE);  
 }  
  
 /\*\*  
 \* WRITE\_CALENDAR、ACCESS\_FINE\_LOCATION 是两个可选权限；没有不影响SDK获取广告；但是如果应用申请到该权限，会显著提升应用的广告收益。  
 \*/  
 if (PackageManager.PERMISSION\_GRANTED != ActivityCompat.checkSelfPermission(this, Manifest.permission.WRITE\_CALENDAR)) {  
 mNeedRequestPMSList.add(Manifest.permission.WRITE\_CALENDAR);  
 }  
 if (PackageManager.PERMISSION\_GRANTED != ActivityCompat.checkSelfPermission(this, Manifest.permission.ACCESS\_FINE\_LOCATION)) {  
 mNeedRequestPMSList.add(Manifest.permission.ACCESS\_FINE\_LOCATION);  
 }  
 //  
 if (0 == mNeedRequestPMSList.size()) {  
 /\*\*  
 \* 权限都已经有了，那么直接调用SDK请求广告。  
 \*/  
 fetchSplashAd();  
 } else {  
 /\*\*  
 \* 有权限需要申请，主动申请。  
 \*/  
 String[] temp = new String[mNeedRequestPMSList.size()];  
 mNeedRequestPMSList.toArray(temp);  
 ActivityCompat.requestPermissions(this, temp, REQUEST\_PERMISSIONS\_CODE);  
 }  
 }  
  
 /\*\*  
 \* 处理权限申请的结果  
 \*  
 \* @param requestCode  
 \* @param permissions  
 \* @param grantResults  
 \*/  
 @Override  
 public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull int[] grantResults) {  
 switch (requestCode) {  
 /\*\*  
 \*处理SDK申请权限的结果。  
 \*/  
 case REQUEST\_PERMISSIONS\_CODE:  
 if (hasNecessaryPMSGranted()) {  
 /\*\*  
 \* 应用已经获得SDK运行必须的READ\_PHONE\_STATE、WRITE\_EXTERNAL\_STORAGE两个权限，直接请求广告。  
 \*/  
 fetchSplashAd();  
 } else {  
 /\*\*  
 \* 如果用户没有授权，那么应该说明意图，引导用户去设置里面授权。  
 \*/  
 Toast.makeText(this, "应用缺少SDK运行必须的READ\_PHONE\_STATE、WRITE\_EXTERNAL\_STORAGE两个权限！请点击\"应用权限\"，打开所需要的权限。", Toast.LENGTH\_LONG).show();  
 Intent intent = new Intent(Settings.ACTION\_APPLICATION\_DETAILS\_SETTINGS);  
 intent.setData(Uri.parse("package:" + getPackageName()));  
 startActivity(intent);  
 finish();  
 }  
 break;  
 default:  
 break;  
 }  
 }  
  
 /\*\*  
 \* 判断应用是否已经获得SDK运行必须的READ\_PHONE\_STATE、WRITE\_EXTERNAL\_STORAGE两个权限。  
 \* @return  
 \*/  
 private boolean hasNecessaryPMSGranted() {  
 if (PackageManager.PERMISSION\_GRANTED == ActivityCompat.checkSelfPermission(this, Manifest.permission.READ\_PHONE\_STATE)) {  
 if (PackageManager.PERMISSION\_GRANTED == ActivityCompat.checkSelfPermission(this, Manifest.permission.WRITE\_EXTERNAL\_STORAGE)) {  
 return true;  
 }  
 }  
 return false;  
 }  
}

**注意点：**

<!--闪屏广告必须全屏展示，通过android:theme="@android:style/Theme.Light.NoTitleBar.Fullscreen"设置你的SplashActivity为全屏显示，  
闪屏只支持竖屏展示，通过android:screenOrientation="portrait" 设置你的SplashActivity为竖屏显示-->  
<activity  
 android:name="com.oppo.mobaddemo.activity.SplashActivity"  
 android:screenOrientation="portrait"  
 android:theme="@android:style/Theme.Light.NoTitleBar.Fullscreen">  
 <intent-filter>  
 <action android:name="android.intent.action.MAIN" />  
 <category android:name="android.intent.category.LAUNCHER" />  
 </intent-filter>  
</activity>

### 闪屏广告主要API

* **com.oppo.mobad.api.ad.SplashAd:**

| **方法名** | **方法介绍** |
| --- | --- |
| SplashAd(Activity activity, String posId, ISplashAdListener iSplashAdListener, SplashAdParams splashAdParams) throws NullPointerException | 构造函数  注意：传入的形参如果为空，会抛出NullPointerException异常 |
| destroyAd() | 销毁闪屏广告,该广告被销毁后，不能再请求广告 |

* **com.oppo.mobad.api.listener.ISplashAdListener:**

闪屏广告状态回调接口，对于大部分开发者只需要关注其中少量的方法。

| **方法名** | **方法介绍** | |
| --- | --- | --- |
| onAdShow() | | 当广告曝光时发起的回调 |
| onAdFailed(String errMsg) | | 广告加载失败，errMsg用于描述失败原因。参考文档中errMsg相关章节 |
| onAdClick() | | 当广告点击时发起的回调，由于点击去重等原因可能和联盟平台最终的统计数据有差异 |
| onAdDismissed() | 广告消失回调 | |

* **com.oppo.mobad.api.params.SplashAdParams.Builder:**

闪屏广告初始化参数，可以设置请求超时时间、闪屏广告下面应用的标题、描述。

| **方法名** | **方法介绍** | |
| --- | --- | --- |
| setFetchTimeout(long fetchTimeout) | | 设置请求广告的超时时间，单位ms，范围[3000,5000] |
| setTitle(String title) | | 设置闪屏广告下部应用标题  注意：如果应用没有传入该字段，将显示应用的名称 |
| setDesc(String desc) | | 设置闪屏广告下部应用描述  注意：如果应用没有传入该字段，将显示默认的文案“欢迎使用” |
| setShowPreLoadPage(boolean showPreLoadPage) | | 设置是否展示SDK默认的闪屏加载界面  注意：默认展示SDK默认的闪屏加载界面 |
| setBottomArea(View bottomArea) | | 设置闪屏底部区域视图  注意：这个方法允许应用自定义闪屏底部区域的视图 |
| build() | | 构造返回一个SplashAdParams对象 |

## Banner广告

### 3.1.1广告接入代码示例

(详细内容请参考压缩包中的代码示例)

public class BannerActivity extends Activity implements IBannerAdListener {

private static final String TAG = "BannerActivity";

private FrameLayout mFrameLayout;

private BannerAd mBannerAd;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_banner);

init();

}

private void init() {

mFrameLayout = (FrameLayout) getWindow().getDecorView().findViewById(android.R.id.content);

/\*\*

\* new bannerAd

\*/

mBannerAd = new BannerAd(this, Constants.BANNER\_POS\_ID);

/\*\*

\* set banner action listener.

\*/

mBannerAd.setAdListener(this);

/\*\*

\* get banner view and add it to your window.

\*

\*/

View adView = mBannerAd.getAdView();

/\*\*

\* adView maye be null.here must judge whether adView is null.

\*/

if (null != adView) {

mFrameLayout.addView(adView);

}

/\*\*

\* invoke loadAd() method to request ad.

\*/

mBannerAd.loadAd();

}

public void onBnClick(View view) {

switch (view.getId()) {

case R.id.load\_ad\_bn:

mBannerAd.loadAd();

break;

case R.id.destroy\_ad\_bn:

/\*\*

\* invoke destroyAd() method to release bannerAd's ad resources.

\*/

mBannerAd.destroyAd();

break;

}

}

@Override

public void onAdShow() {

Log.d(TAG, "onAdShow");

}

@Override

public void onAdFailed(String errMsg) {

Log.d(TAG, "onAdFailed:errMsg=" + (null != errMsg ? errMsg : ""));

}

@Override

public void onAdReady() {

Log.d(TAG, "onAdReady");

}

@Override

public void onAdClick() {

Log.d(TAG, "onAdClick");

}

@Override

public void onAdClose() {

Log.d(TAG, "onAdClose");

}

}

### 3.1.2 Banner广告主要API

* **com.oppo.mobad.api.ad.BannerAd:**

BannerAd类，需要容器提供57dp的高度用于展示Banner广告，宽度Banner广告会自适应，但要求不少于300dp。

| **方法名** | **方法介绍** |
| --- | --- |
| BannerAd(Activity activity, String posId) | 构造函数 |
| setAdListener(IBannerAdListener iBannerAdListener) | 设置广告行为监听器 |
| loadAd() | 请求广告 |
| destroyAd() | 销毁Banner广告,该广告被销毁后，不能再请求广告 |
| getAdView() | 获取一个广告View对象，然后应用可以把该View展示在需要的地方，该View可能为空，需要应用判断后再添加 |

* **com.oppo.mobad.api.listener. IBannerAdListener:**

Banner广告回调接口，对于大部分开发者只需要关注其中少量的方法。

| **方法名** | **方法介绍** |
| --- | --- |
| onAdShow() | 当广告曝光时发起的回调 |
| onAdFailed(String errMsg) | 广告加载失败，errMsg用于描述失败原因。参考文档中errMsg相关章节 |
| onAdReady() | 广告加载成功回调，表示广告相关的资源已经加载完毕，Ready To Show |
| onAdClick() | 当广告点击时发起的回调，由于点击去重等原因可能和联盟平台最终的统计数据有差异 |
| onAdClose() | 当广告关闭时调用，只有在使用了Banner广告自身的关闭按钮时生效 |

## 插屏广告

### 广告接入代码示例

(详细内容请参考压缩包中的代码示例)

public class InterstitialActivity extends Activity implements IInterstitialAdListener {

private static final String TAG = "InterstitialActivity";

private InterstitialAd mInterstitialAd;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_interstitial);

init();

}

private void init() {

/\*\*

\* new InterstitialAd.

\*/

mInterstitialAd = new InterstitialAd(this, Constants.INTERSTITIAL\_POS\_ID);

/\*\*

\* set InterstitialAd action listener.

\*/

mInterstitialAd.setAdListener(this);

/\*\*

\* invoke loadAd() method to request ad.

\*/

mInterstitialAd.loadAd();

}

public void onBnClick(View view) {

switch (view.getId()) {

case R.id.load\_ad\_bn:

mInterstitialAd.loadAd();

break;

case R.id.show\_ad\_bn:

mInterstitialAd.showAd();

break;

case R.id.destroy\_ad\_bn:

/\*\*

\* invoke destroyAd() method to release InterstitialAd's ad resources.

\*/

mInterstitialAd.destroyAd();

break;

}

}

@Override

public void onAdShow() {

Log.d(TAG, "onAdShow");

}

@Override

public void onAdFailed(String errMsg) {

Log.d(TAG, "onAdFailed:errMsg=" + (null != errMsg ? errMsg : ""));

}

@Override

public void onAdReady() {

Log.d(TAG, "onAdReady");

/\*\*

\* invoke showAd() method to show ad when ad ready.

\*

\*/

mInterstitialAd.showAd();

}

@Override

public void onAdClick() {

Log.d(TAG, "onAdClick");

}

@Override

public void onAdClose() {

Log.d(TAG, "onAdClose");

}

}

### 插屏广告主要API

* **com.oppo.mobad.api.ad. InterstitialAd:**

| **方法名** | **方法介绍** |
| --- | --- |
| InterstitialAd(Activity activity, String posId) | 构造函数 |
| setAdListener(IInterstitialAdListener iInterstitialAdListener) | 添加广告监听器 |
| loadAd() | 加载广告 |
| showAd() | 展示插屏广告 |
| destroyAd() | 销毁插屏广告,该广告被销毁后，不能再请求广告 |

* **com.oppo.mobad.api.listener.IInterstitialAdListener:**

插屏广告状态回调接口，对于大部分开发者只需要关注其中少量的方法。

| **方法名** | **方法介绍** | |
| --- | --- | --- |
| onAdShow() | | 当广告曝光时发起的回调 |
| onAdFailed(String errMsg) | | 广告加载失败，errMsg用于描述失败原因。参考文档中errMsg相关章节 |
| onAdReady() | | 广告加载成功回调，表示广告相关的资源已经加载完毕，Ready To Show |
| onAdClick() | | 当广告点击时发起的回调，由于点击去重等原因可能和联盟平台最终的统计数据有差异 |
| onAdClose() | 广告消失回调 | |

## 原生广告

### 广告接入代码示例

(详细内容请参考压缩包中的代码示例)

public class NativeOneActivity extends Activity implements INativeAdListener {  
 private static final String TAG = "NativeOneActivity";  
 private NativeAd mNativeAd;  
 /\*\*  
 \* 原生广告数据对象。  
 \*/  
 private INativeAdData mINativeAdData;  
 //  
 private AQuery mAQuery;  
  
 @Override  
 protected void onCreate(@Nullable Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.activity\_native\_one);  
 initView();  
 initData();  
 }  
  
 private void initView() {  
 mAQuery = new AQuery(this);  
 findViewById(R.id.native\_ad\_container).setVisibility(View.GONE);  
 mAQuery.id(R.id.load\_native\_ad\_bn).clicked(this, "loadAd");  
 mAQuery.id(R.id.show\_native\_ad\_bn).clicked(this, "showAd").enabled(false);  
 }  
  
 public void loadAd() {  
 if (null != mNativeAd) {  
 /\*\*  
 \*调用loadAd方法加载原生广告。  
 \*/  
 mNativeAd.loadAd();  
 }  
 }  
  
 public void showAd() {  
 /\*\*  
 \*在展示原生广告前调用isAdValid判断当前广告是否有效，否则即使展示了广告，也是无效曝光、点击，不计费的  
 \*注意：每个INativeAdData对象只有一次有效曝光、一次有效点击；多次曝光，多次点击都只扣一次费。  
 \*/  
 if (null != mINativeAdData && mINativeAdData.isAdValid()) {  
 findViewById(R.id.native\_ad\_container).setVisibility(View.VISIBLE);  
 /\*\*  
 \*展示推广应用的ICON。  
 \*/  
 if (null != mINativeAdData.getIconFiles() && mINativeAdData.getIconFiles().size() > 0) {  
 mAQuery.id(R.id.icon\_iv).image(mINativeAdData.getIconFiles().get(0).getUrl(), false, true);  
 }  
 /\*\*  
 \* 判断是否需要展示“广告”Logo标签  
 \*/  
 if (null != mINativeAdData.getLogoFile()) {  
 mAQuery.id(R.id.logo\_iv).image(mINativeAdData.getLogoFile().getUrl(), false, true);  
 }  
 mAQuery.id(R.id.title\_tv).text(null != mINativeAdData.getTitle() ? mINativeAdData.getTitle() : "");  
 mAQuery.id(R.id.desc\_tv).text(null != mINativeAdData.getDesc() ? mINativeAdData.getDesc() : "");  
 mAQuery.id(R.id.action\_bn).text(null != mINativeAdData.getClickBnText() ? mINativeAdData.getClickBnText() : "").clicked(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 /\*\*  
 \*原生广告被点击时必须调用onAdClick方法通知SDK进行点击统计；  
 \* 注意：onAdClick方法必须在onAdShow方法之后再调用才有效，否则是无效点击。  
 \*/  
 mINativeAdData.onAdClick(v);  
 }  
 });  
 /\*\*  
 \* 原生广告曝光时必须调用onAdShow方法通知SDK进行曝光统计，否则就没有曝光数据。  
 \*/  
 mINativeAdData.onAdShow(findViewById(R.id.native\_ad\_container));  
 }  
 }  
  
 private void initData() {  
 /\*\*  
 \*构造NativeAd对象。  
 \*/  
 mNativeAd = new NativeAd(this, Constants.NATIVE\_ONE\_POS\_ID, this);  
 }  
  
 @Override  
 protected void onDestroy() {  
 if (null != mNativeAd) {  
 /\*\*  
 \*銷毀NativeAd对象，释放资源。  
 \*/  
 mNativeAd.destroyAd();  
 }  
 super.onDestroy();  
 }  
  
 /\*\*  
 \* 原生广告加载成功，在onAdSuccess回调广告数据  
 \*  
 \* @param iNativeAdDataList  
 \*/  
 @Override  
 public void onAdSuccess(List<INativeAdData> iNativeAdDataList) {  
 if (null != iNativeAdDataList && iNativeAdDataList.size() > 0) {  
 mINativeAdData = iNativeAdDataList.get(0);  
 mAQuery.id(R.id.show\_native\_ad\_bn).enabled(true);  
 Toast.makeText(NativeOneActivity.this, "加载原生广告成功", Toast.LENGTH\_SHORT).show();  
 }  
 }  
  
 @Override  
 public void onAdFailed(NativeAdError nativeAdError) {  
 Toast.makeText(NativeOneActivity.this, "加载原生广告失败,错误码：" + nativeAdError.toString(), Toast.LENGTH\_LONG).show();  
 }  
  
 @Override  
 public void onAdError(NativeAdError nativeAdError, INativeAdData iNativeAdData) {  
 Toast.makeText(NativeOneActivity.this, "调用原生广告统计方法出错,错误码：" + nativeAdError.toString(), Toast.LENGTH\_LONG).show();  
 }  
}

### 原生广告主要API

* **com.oppo.mobad.api.ad.NativeAd:**

| **方法名** | **方法介绍** |
| --- | --- |
| NativeAd(Context context, String posId, INativeAdListener iNativeAdListener) | 构造函数 |
| loadAd() | 加载广告 |
| loadAd(NativeAdParams nativeAdParams) | 加载广告，nativeAdParams请求广告的配置参数 |
| destroyAd() | 销毁原生广告,该广告被销毁后，不能再请求广告 |

* **com.oppo.mobad.api.listener.INativeAdListener:**

插屏广告状态回调接口，对于大部分开发者只需要关注其中少量的方法。

| **方法名** | **方法介绍** |
| --- | --- |
| onAdSuccess(List<INativeAdData> iNativeAdDataList) | 当广告请求成功时回调 |
| onAdFailed(NativeAdError nativeAdError) | 当广告请求失败时回调 |
| onAdError(NativeAdError nativeAdError, INativeAdData iNativeAdData) | 调用广告统计方法出错时回调 |

* **com.oppo.mobad.api.params.INativeAdData:**

| **方法名** | **方法介绍** |
| --- | --- |
| getTitle() | 获取广告标题 |
| getDesc() | 获取广告描述 |
| getIconFiles() | 获取推广应用的Icon图标 |
| getImgFiles() | 获取广告图片 |
| onAdShow(View view) | 在广告曝光时调用 |
| onAdClick(View view) | 在广告点击时调用，需要在onAdShow调用后再调用，否则视为无效点击 |
| getCreativeType() | 获取创意类型，取值说明：  0：无  1：纯文字  2：图片  3：图文混合  4：视频 |
| getInteractionType() | 获取广告点击之后的交互类型，取值说明：  0：无  1：浏览类  2：下载类  3：浏览器（下载中间页广告）  4：打开应用首页  5：打开应用详情页 |
| getLogoFile() | 获取“广告”标签图片 |
| isAdValid() | 广告是否有效，在广告曝光前onAdShow()前调用，判断广告是否合法，不合法的广告，onAdShow调用无效 |
| getExtra() | 获取额外信息，预留字段 |
| getClickBnText() | 获取点击按钮文本描述 |

* **com.oppo.mobad.api.params.INativeAdFile:**

| **方法名** | **方法介绍** |
| --- | --- |
| getUrl() | 获取文件下载地址 |
| getMd5() | 获取文件的Md5值，可能为空 |

* **com.oppo.mobad.api.params.NativeAdError:**

| **字段** | **字段介绍** |
| --- | --- |
| code | 错误码，请参照《错误码》 |
| msg | 错误描述 |

## 原生模板广告

### 广告接入代码示例

(详细内容请参考压缩包中的代码示例)

public class NativeTempletNormalActivity extends Activity implements INativeTempletAdListener {  
 private static final String TAG = "NativeTempletNormalActivity";  
 public static final String EXTRA\_KEY\_POS\_ID = "posId";  
 /\*\*  
 \* 广告容器  
 \*/  
 private ViewGroup mAdContainer;  
  
 private EditText mAdWidthET;  
 private EditText mAdHeightET;  
 /\*\*  
 \* 原生模板广告对象  
 \*/  
 private NativeTempletAd mNativeTempletAd;  
 /\*\*  
 \* 原生模板广告View对象，在请求广告成功以后返回  
 \*/  
 private INativeTempletAdView mINativeTempletAdView;  
 //default =0.  
 private int mWidth = 0;  
 private int mHeight = 0;  
 //  
 private String mPosId;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.activity\_native\_templet\_normal);  
 initView();  
 initData();  
 }  
  
 @Override  
 protected void onDestroy() {  
 destroyAd();  
 super.onDestroy();  
 }  
  
 private void initView() {  
 mAdContainer = (ViewGroup) findViewById(R.id.ad\_container);  
 mAdWidthET = (EditText) findViewById(R.id.width\_et);  
 mAdHeightET = (EditText) findViewById(R.id.height\_et);  
 }  
  
 private void initData() {  
 mPosId = getIntent().getStringExtra(EXTRA\_KEY\_POS\_ID);  
 }  
  
 public void onBnClick(View view) {  
 switch (view.getId()) {  
 case R.id.load\_ad\_bn:  
 loadAd();  
 break;  
 case R.id.destroy\_ad\_bn:  
 destroyAd();  
 break;  
 }  
 }  
  
 private void loadAd() {  
 if (null == mNativeTempletAd || checkEditTextChanged()) {  
 if (!checkEditTextEmpty()) {  
 mWidth = Integer.valueOf(mAdWidthET.getText().toString());  
 mHeight = Integer.valueOf(mAdHeightET.getText().toString());  
 }  
 /\*\*  
 \* 通过构造NativeAdSize对象，在NativeTempletAd初始化时传入、可以指定原生模板广告的大小，单位为dp  
 \* 在接入、调试模板广告的过程中，可以利用这个字段调整广告View的大小，找到与自己的App需求适合的最佳广告位尺寸  
 \* 确定最佳尺寸后，应该把这个ADSize固定下来，并在构造NativeTempletAd的时候传入  
 \* 也可以传入null，展示默认的大小  
 \*/  
 NativeAdSize nativeAdSize = new NativeAdSize.Builder()  
 .setWidthInDp(mWidth)  
 .setHeightInDp(mHeight)  
 .build();  
 mNativeTempletAd = new NativeTempletAd(this, mPosId, nativeAdSize, this);  
 }  
 /\*\*  
 \*调用loadAd方法请求原生模板广告  
 \*/  
 mNativeTempletAd.loadAd();  
 hideSoftInput();  
 }  
  
 private void destroyAd() {  
 hideSoftInput();  
 /\*\*  
 \* 每一个mINativeTempletAdView使用完以后都要调用destroy释放  
 \*/  
 if (null != mINativeTempletAdView) {  
 mINativeTempletAdView.destroy();  
 }  
 /\*\*  
 \* mNativeTempletAd调用destroyAd方法释放相关资源  
 \*/  
 if (null != mNativeTempletAd) {  
 mNativeTempletAd.destroyAd();  
 }  
 }  
  
 @Override  
 public void onAdSuccess(List<INativeTempletAdView> iNativeTempletAdViewList) {  
 Log.d(TAG, "onAdSuccess size=" + (null != iNativeTempletAdViewList ? iNativeTempletAdViewList.size() : "null"));  
 if (null != iNativeTempletAdViewList && iNativeTempletAdViewList.size() > 0) {  
 /\*\*  
 \* 释放前一个INativeTempletAdView对对象资源  
 \*/  
 if (null != mINativeTempletAdView) {  
 mINativeTempletAdView.destroy();  
 }  
 /\*\*  
 \* 将广告“容器”置于可见状态，否则将无法产生有效曝光  
 \*/  
 if (View.VISIBLE != mAdContainer.getVisibility()) {  
 mAdContainer.setVisibility(View.VISIBLE);  
 }  
 /\*\*  
 \* 如果容器已经有广告、则移除  
 \*/  
 if (mAdContainer.getChildCount() > 0) {  
 mAdContainer.removeAllViews();  
 }  
 /\*\*  
 \* 获取广告View  
 \*/  
 mINativeTempletAdView = iNativeTempletAdViewList.get(0);  
 View adView = mINativeTempletAdView.getAdView();  
 if (null != adView) {  
 /\*\*  
 \* 添加广告View到广告“容器”  
 \*/  
 mAdContainer.addView(adView);  
 /\*\*  
 \* 调用render方法渲染广告  
 \*/  
 mINativeTempletAdView.render();  
 }  
 }  
 }  
  
 @Override  
 public void onAdFailed(NativeAdError nativeAdError) {  
 Log.d(TAG, "onAdFailed nativeAdError=" + (null != nativeAdError ? nativeAdError.toString() : "null"));  
 Toast.makeText(NativeTempletNormalActivity.this, "load native templet ad error,error msg：" + (null != nativeAdError ? nativeAdError.toString() : "null"), Toast.LENGTH\_LONG).show();  
 }  
  
 @Override  
 public void onAdClick(INativeTempletAdView iNativeTempletAdView) {  
 Log.d(TAG, "onAdClick iNativeTempletAdView=" + (null != iNativeTempletAdView ? iNativeTempletAdView : "null"));  
 }  
  
 @Override  
 public void onAdShow(INativeTempletAdView iNativeTempletAdView) {  
 Log.d(TAG, "onAdShow iNativeTempletAdView=" + (null != iNativeTempletAdView ? iNativeTempletAdView : "null"));  
 }  
  
 @Override  
 public void onAdClose(INativeTempletAdView iNativeTempletAdView) {  
 Log.d(TAG, "onAdClose iNativeTempletAdView=" + (null != iNativeTempletAdView ? iNativeTempletAdView : "null"));  
 /\*\*  
 \*当广告模板中的关闭按钮被点击时，广告将不再展示。INativeTempletAdView也会被Destroy，不再可用  
 \*/  
 if (null != mAdContainer && mAdContainer.getChildCount() > 0) {  
 mAdContainer.removeAllViews();  
 mAdContainer.setVisibility(View.GONE);  
 }  
 }  
  
 @Override  
 public void onRenderSuccess(INativeTempletAdView iNativeTempletAdView) {  
 Log.d(TAG, "onRenderSuccess iNativeTempletAdView=" + (null != iNativeTempletAdView ? iNativeTempletAdView : "null"));  
 }  
  
 @Override  
 public void onRenderFailed(NativeAdError nativeAdError, INativeTempletAdView iNativeTempletAdView) {  
 Log.d(TAG, "onRenderFailed nativeAdError=" + (null != nativeAdError ? nativeAdError.toString() : "null") + ",iNativeTempletAdView=" + (null != iNativeTempletAdView ? iNativeTempletAdView : "null"));  
 Toast.makeText(NativeTempletNormalActivity.this, "render native templet ad error,error msg：" + (null != nativeAdError ? nativeAdError.toString() : "null"), Toast.LENGTH\_LONG).show();  
 }  
  
 private boolean checkEditTextEmpty() {  
 String width = mAdWidthET.getText().toString();  
 String height = mAdHeightET.getText().toString();  
 if (TextUtils.isEmpty(width) || TextUtils.isEmpty(height)) {  
 return true;  
 }  
 return false;  
 }  
  
 private boolean checkEditTextChanged() {  
 if (!checkEditTextEmpty()) {  
 return Integer.valueOf(mAdWidthET.getText().toString()) != mWidth  
 || Integer.valueOf(mAdHeightET.getText().toString()) != mHeight;  
 }  
 return false;  
 }  
  
 private void hideSoftInput() {  
 ((InputMethodManager) getSystemService(Context.INPUT\_METHOD\_SERVICE)).hideSoftInputFromWindow(  
 NativeTempletNormalActivity.this.getCurrentFocus().getWindowToken(),  
 InputMethodManager.HIDE\_NOT\_ALWAYS);  
 }  
}

### 原生模板广告主要API

* **com.oppo.mobad.api.ad.NativeTempletAd:**

| **方法名** | **方法介绍** |
| --- | --- |
| NativeTempletAd(Context context, String posId, NativeAdSize nativeAdSize, INativeTempletAdListener iNativeTempletAdListener) | 构造函数 |
| loadAd() | 加载广告 |
| loadAd(NativeAdParams nativeAdParams) | 加载广告，nativeAdParams请求广告的配置参数 |
| destroyAd() | 销毁广告,该广告被销毁后，不能再请求广告 |

* **com.oppo.mobad.api.listener.INativeTempletAdListener:**

| **方法名** | **方法介绍** |
| --- | --- |
| onAdSuccess(List<INativeTempletAdView> iNativeTempletAdViewList) | 当广告请求成功时回调 |
| onAdFailed(NativeAdError nativeAdError) | 当广告请求失败时回调 |
| onAdClick(INativeTempletAdView iNativeTempletAdView) | 当广告被点击时回调 |
| onAdShow(INativeTempletAdView iNativeTempletAdView) | 当广告曝光时回调 |
| onAdClose(INativeTempletAdView iNativeTempletAdView) | 当广告被关闭时回调 |
| onRenderSuccess(INativeTempletAdView iNativeTempletAdView) | 当广告被渲染成功时回调 |
| onRenderFailed(NativeAdError nativeAdError, INativeTempletAdView iNativeTempletAdView) | 当广告渲染失败时回调 |

* **com.oppo.mobad.api.params.NativeAdSize:**

| **字段** | **字段介绍** |
| --- | --- |
| widthInDp | 广告位宽度 |
| heightInDp | 广告位高度 |

* **com.oppo.mobad.api.params.NativeAdParams:**

| **字段** | **字段介绍** |
| --- | --- |
| fetchTimeout | 请求广告最大超时时间，单位ms |

* **com.oppo.mobad.api.params.NativeAdError:**

| **字段** | **字段介绍** |
| --- | --- |
| code | 错误码，请参照《错误码》 |
| msg | 错误描述 |

* **com.oppo.mobad.api.params.INativeTempletAdView:**

| **方法名** | **方法介绍** |
| --- | --- |
| getAdview() | 获取广告View |
| render() | 渲染广告View |
| Destroy() | 销毁广告View和资源 |

## 激励视频广告

### 广告接入代码示例

public class RewardVideoActivity extends Activity implements IRewardVideoAdListener {  
 private static final String TAG = "RewardVideoActivity";  
 private TextView mStatusTv;  
 private RewardVideoAd mRewardVideoAd;  
 private AlertDialog mAlertDialog;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.activity\_reward\_video);  
 initView();  
 initData();  
 }  
  
 @Override  
 protected void onDestroy() {  
 destroyVideo();  
 super.onDestroy();  
 }  
  
 private void initView() {  
 mStatusTv = (TextView) findViewById(R.id.status\_tv);  
 mStatusTv.setMovementMethod(new ScrollingMovementMethod());  
 }  
  
 private void initData() {  
 /\*\*  
 \* 构造激励视频广告对象  
 \*/  
 mRewardVideoAd = new RewardVideoAd(this, Constants.REWARD\_VIDEO\_POS\_ID, this);  
 printStatusMsg("初始化视频广告.");  
 /\*\*  
 \*当请求视频广告成功时，用来展示视频播放入口的Dialog  
 \*/  
 mAlertDialog = new AlertDialog.Builder(this)  
 .setCancelable(false)  
 .setTitle("获取奖励")  
 .setMessage("观看视频、重新获取闯关机会？")  
 .setPositiveButton("确定", new DialogInterface.OnClickListener() {  
 @Override  
 public void onClick(DialogInterface dialog, int which) {  
 playVideo();  
 dialog.dismiss();  
 }  
 })  
 .setNegativeButton("取消", new DialogInterface.OnClickListener() {  
 @Override  
 public void onClick(DialogInterface dialog, int which) {  
 dialog.dismiss();  
 }  
 })  
 .create();  
 }  
  
 public void onBnClick(View view) {  
 switch (view.getId()) {  
 case R.id.req\_video\_ad:  
 loadVideo();  
 break;  
 }  
 }  
  
 private void loadVideo() {  
 /\*\*  
 \* 调用loadAd方法请求激励视频广告;通过RewardVideoAdParams.setFetchTimeout方法可以设置请求  
 \* 视频广告最大超时时间，单位毫秒；  
 \*/  
 RewardVideoAdParams rewardVideoAdParams=new RewardVideoAdParams.Builder()  
 .setFetchTimeout(3000)  
 .build();  
 mRewardVideoAd.loadAd(rewardVideoAdParams);  
 printStatusMsg("请求加载视频广告.");  
 }  
  
 private void playVideo() {  
 /\*\*  
 \* 调用showAd方法播放激励视频广告  
 \*/  
 mRewardVideoAd.showAd();  
 printStatusMsg("播放视频广告.");  
 }  
  
 private void destroyVideo() {  
 /\*\*  
 \* 销毁激励视频广告  
 \*/  
 mRewardVideoAd.destroyAd();  
 printStatusMsg("释放视频广告资源.");  
 }  
  
 private void printStatusMsg(String txt) {  
 if (null != txt) {  
 Log.d(TAG, txt);  
 mStatusTv.setText(mStatusTv.getText() + "\n" + txt);  
 }  
 }  
  
 @Override  
 public void onAdSuccess() {  
 /\*\*  
 \*请求视频广告成功、展示播放视频的入口Dialog  
 \*/  
 mAlertDialog.show();  
 printStatusMsg("请求视频广告成功.");  
 }  
  
 @Override  
 public void onAdFailed(String msg) {  
 /\*\*  
 \* 请求广告失败、不展示播放视频的入口Dialog  
 \*/  
 printStatusMsg("请求视频广告失败. msg=" + msg);  
 }  
  
 @Override  
 public void onAdClick(long currentPosition) {  
 printStatusMsg("视频广告被点击，当前播放进度 = " + currentPosition + " 秒");  
 }  
  
 @Override  
 public void onVideoPlayStart() {  
 printStatusMsg("视频开始播放.");  
 }  
  
 @Override  
 public void onVideoPlayComplete() {  
 /\*\*  
 \* TODO 用户完成激励视频观看，给予用户奖励。  
 \*/  
 printStatusMsg("视频广告播放完成,给用户发放奖励.");  
 }  
  
 @Override  
 public void onVideoPlayError(String msg) {  
 printStatusMsg("视频播放错误，错误信息=" + msg);  
 }  
  
 @Override  
 public void onVideoPlayClose(long currentPosition) {  
 printStatusMsg("视频广告被关闭，当前播放进度 = " + currentPosition + " 秒");  
 }  
  
 @Override  
 public void onLandingPageOpen() {  
 printStatusMsg("视频广告落地页打开.");  
 }  
  
 @Override  
 public void onLandingPageClose() {  
 printStatusMsg("视频广告落地页关闭.");  
 }  
}

### 激励视频广告主要API

* **com.oppo.mobad.api.ad.RewardVideoAd:**

| **方法名** | **方法介绍** |
| --- | --- |
| RewardVideoAd(Context context, String posId, IRewardVideoAdListener iRewardVideoAdListener) | 构造函数 |
| loadAd() | 加载广告 |
| loadAd(RewardVideoAdParams rewardVideoAdParams) | 加载广告，rewardVideoAdParams请求广告的配置参数 |
| showAd() | 展示广告 |
| destroyAd() | 销毁广告,该广告被销毁后，不能再请求广告 |

* **com.oppo.mobad.api.listener.IRewardVideoAdListener:**

| **方法名** | **方法介绍** |
| --- | --- |
| onAdSuccess() | 当广告请求成功时回调 |
| onAdFailed(String msg) | 当广告请求失败时回调 |
| onAdClick(long currentPosition) | 当广告被点击时回调 |
| onVideoPlayStart() | 当视频开始播放时回调 |
| onVideoPlayComplete() | 当视频播放完成时回调 |
| onVideoPlayError(String msg) | 当视频播放错误时回调 |
| onVideoPlayClose(long currentPosition) | 当视频播放过程中被关闭时回调 |
| onLandingPageOpen() | 当视频播放完毕落地页打开时回调 |
| onLandingPageClose() | 当视频落地页关闭时回调 |

* **com.oppo.mobad.api.params.RewardVideoParams:**

| **字段名称** | **字段介绍** |
| --- | --- |
| fetchTimeout | 请求广告超时时间、单位ms |

# SDK相关问题排查

如果根据正常的流程仍然无法在嵌入OPPO sdk的app中看到广告，可以尝试使用logcat排查问题。OPPO SDK默认设置下会对广告流程中的关键步骤打印日志，日志格式如下：

LogTag：mob\_ad

# 错误码

| **错误码** | **描述** |
| --- | --- |
| -1 | 未知错误 |
| 1 | 未知异常 |
| 1001 | 参数异常 |
| 1002 | 对象不存在 |
| 1003 | 无广告返回 |
| 1004 | 请求广告错误 |
| 1014 | 应用ID【appId】无效或者不存在 |
| 1015 | 应用ID【appId】和应用包名【pkgName】不匹配 |
| 1016 | 广告位ID【posId】无效或者不存在、或者审核未通过 |
| 10000 | 正常 |
| 10001 | 网络无响应 |
| 10002 | 广告列表为空 |
| 10003 | 广告过期 |
| 10004 | 返回数据为空 |
| 10005 | 原生模板广告View列表为空 |
| 10100 | 无网络 |
| 10101 | 请求广告网络异常 |
| 10102 | 解析广告数据异常 |
| 10200 | 原生广告重复曝光 |
| 10201 | 原生广告曝光过期 |
| 10202 | 原生广告在未调用曝光情况下调用点击 |
| 10203 | 重复点击 |
| 10204 | 点击过期 |
| 10205 | 闪屏广告不允许横屏显示 |
| 10206 | 闪屏广告必须全屏展示 |
| 11000 | SDK初始化失败 |
| 11001 | 广告已经销毁 |
| 11002 | 不能在非OPPO手机上展示广告 |
| 11003 | 请求广告太频繁 |
| 11004 | Banner广告不能展示在异形屏【刘海屏】手机的顶部 |
| 10300 | 原生模板广告渲染失败、广告过期 |
| 10301 | 原生模板渲染失败，广告信息为空 |
| 10302 | 原生模板渲染失败，广告物料为空 |
| 10303 | 原生模板渲染失败，未知的创意类型 |
| 10400 | 没有在线播放视频广告 |
| 10401 | 没有缓存播放视频广告 |
| 10402 | 没有视频广告 |
| 10403 | 无网络，不能播放广告 |
| 10404 | 视频过期 |
| 10405 | 手机系统版本太低，无法播放视频广告 |
| 10406 | 不是Wifi网络、无法播放在线视频广告 |
| 10407 | 不支持的播放模式 |
| 10408 | 视频广告未缓存，无法播放 |
| 10409 | 未知的创意类型 |
| 10410 | 视频播放源异常、无法播放视频 |
| 10411 | 视频渲染失败 |
| 10412 | 视频播放发生未知异常 |