聚效广告平台Android SDK

接入说明文档 V1.13

版本号	修改时间	内容	修改人
1.0	2014-08-07	初稿	聚效产品部
1.1	2014-08-15	增加广告位尺寸说明	聚效产品部
1.2	2014-09-01	简化添加横幅和插屏广告代码	聚效产品部
1.3	2014-09-17	增加显示广告接口、插屏刷新接口、移除系统事件注册接口，更新混淆配置	聚效产品部
1.4	2014-10-29	修改嵌入广告位代码及追踪广告状态代码、修改接口代码	聚效产品部
1.5	2014-11-13	修改添加权限许可代码	聚效产品部
1.6	2014-12-01	增加浮动窗广告接口、浮动窗关闭接口、Activity活动销毁接口	聚效产品部
1.7	2015-02-05	优化内开浏览器，增加Service来实现APP下载	聚效产品部
1.8	2015-03-27	引入原生广告	聚效产品部
1.9	2015-06-02	全屏广告显示效果优化、优化内开浏览器、广告相关Bug修复	聚效产品部
1.10	2015-06-26	支持自定义内开浏览器	聚效产品部
1.11	2015-07-31	支持自定义内开浏览器下载APP	聚效产品部
1.11	2015-09-01	增加NativeBannerAd接口	聚效产品部

1.12	2015-09-18	增加开屏广告接口	聚效产品部
1.13	2016-03-03	增加原生广告回调	聚效产品部

上海聚效广告有限公司

广告形式

本SDK版本支持下列广告形式：

横幅广告：广告置于开发者提供的容器中，由SDK负责渲染。

浮动横幅广告：广告位于屏幕的顶部或底部，由SDK负责渲染。

插屏广告：广告位于屏幕的中间，由SDK负责渲染。

开屏广告：广告置于开发者提供的容器中，支持倒计时功能，由SDK负责渲染。

嵌入式广告：广告置于开发者提供的容器中，且能展示丰富广告素材，由SDK负责渲染。

原生广告：SDK提供广告素材（JSON）给开发者，由开发者负责渲染。

注意事项

1. 请务必在主线程中调用SDK，子线程调用会导致SDK崩溃（SDK会主动抛出异常）。
2. SDK仅支持Android 2.2及以上的版本调用。
3. 请务必为SDK添加其所需权限、注册Activity（MvActivity）及Service（MvService）。
4. SDK的jar包中包含assets目录，如果你使用打包脚本请确保打包脚本里抽取SDK下的assets目录。
5. 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（或其他混淆文件）尾部添加如下配置：

   ```
   -keep class com.mediav.** {*;}
   ```

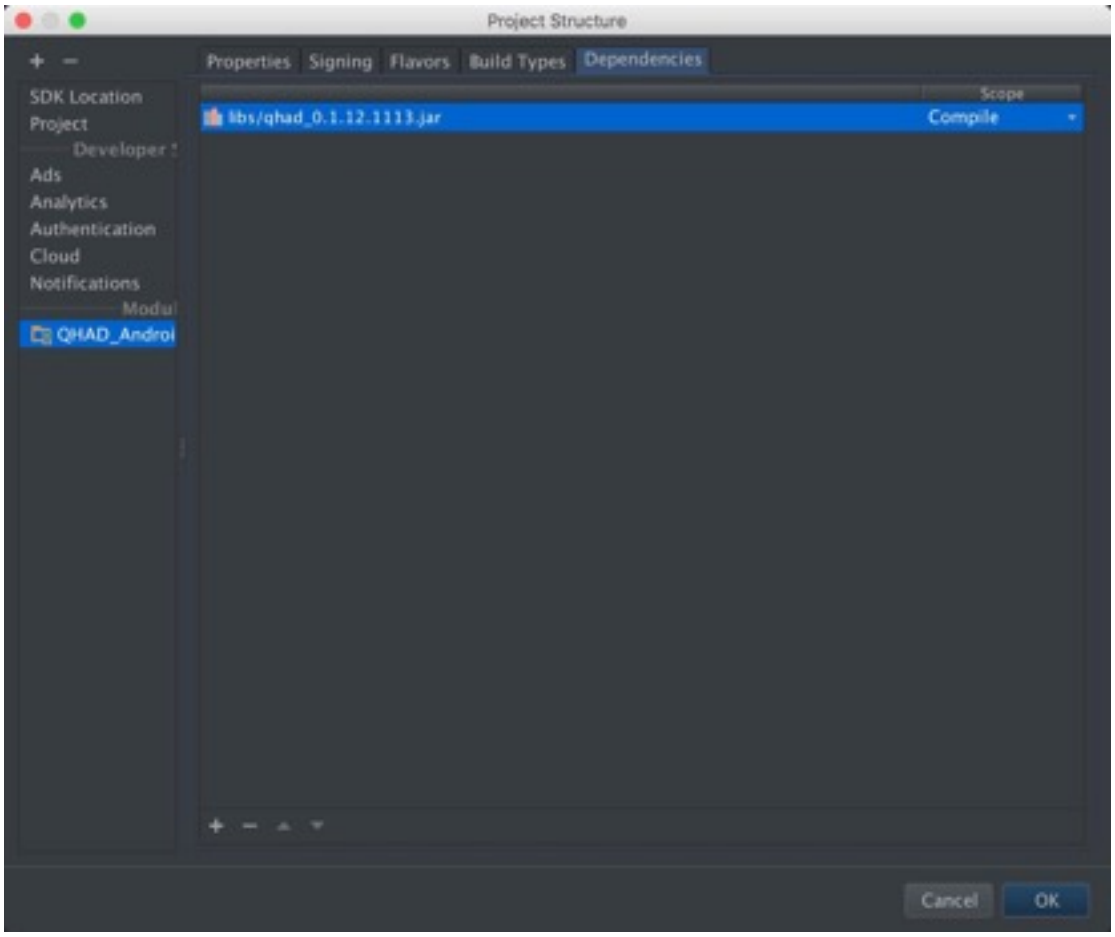
6. 在代码混淆时务必不要混淆android-support-v4.jar中的类。
7. 确保所使用的android-support-v4.jar包中的android.support.v4.app.NotificationCompat.Builder类包含setProgress方法，如果不包含此方法请升级android开发套件。

使用步骤

步骤一：添加SDK到工程中

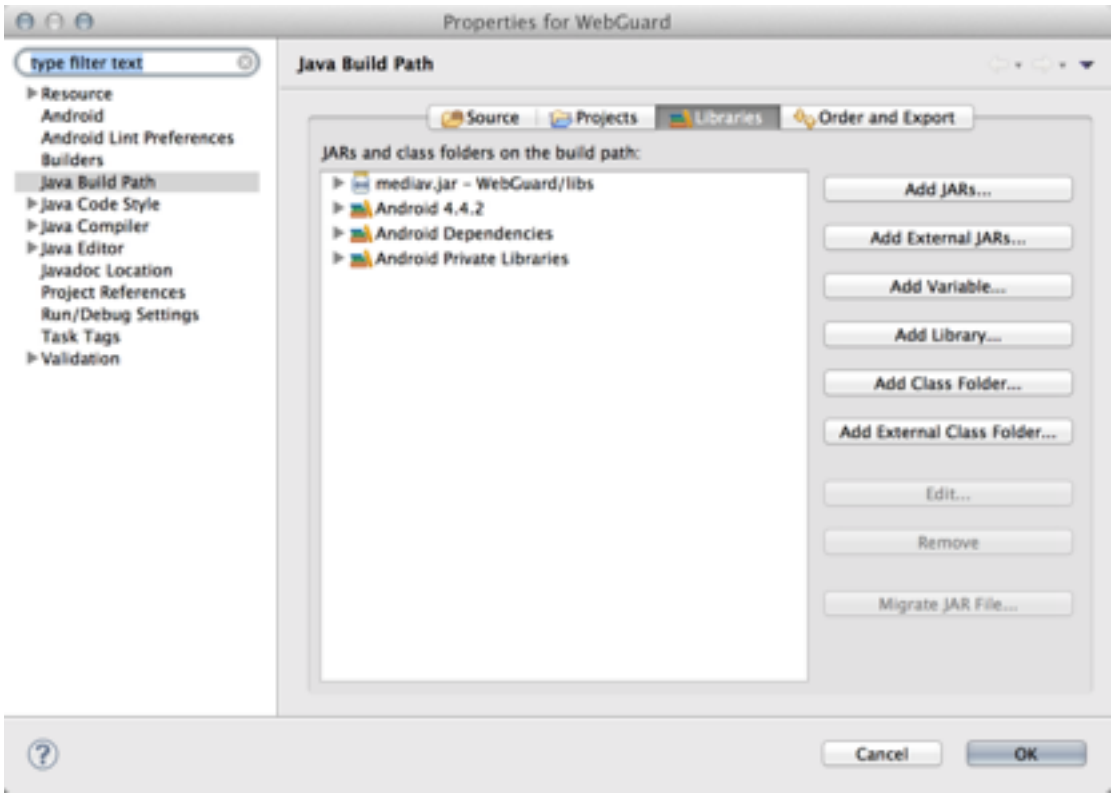
请在工程文件根目录下创建一个名为 libs的子目录，并将SDK的JAR包（mvad_*.jar，*为具体版本号）拷贝到 libs 目录下。

对于 Android Studio工程:
1. 点击菜单栏“File”，选择“Project Structure”（快捷键：Ctrl+Alt+Shift+S）
2. 选择"Dependencies"，点击加号“+”，选择“File Dependency"
3. 选择您拷贝到 libs 目录下的 jar包
4. 点击 “OK” 完成



对于 Eclipse 工程:
1. 在 “Package Explorer” 页签中右击你的工程并选择 “Properties”
2. 在左侧面板中选择 “Java Build Path”

3. 在主窗口中选择 “Libraries” 页签
4. 点击 “Add JARs...” 按钮
5. 选择您拷贝到 libs 目录下的 jar包
6. 点击 “OK” 完成



步骤二：修改AndroidManifest.xml

1. 添加权限许可

在AndroidManifest.xml文件中的<application>标签之前，请为SDK添加以下权限许可代码（如果App本身没有的话）：

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" /
>
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<uses-permission android:name="android.permission.GET_TASKS"/>
```

注册MvActivity：

```
<activity
            android:name=" com.mediav.ads.sdk.adcore.MvActivity "
            android:configChanges="orientation|screenSize|keyboardHidden" >
</activity>
```

注册MvService服务：

```
<service android:name=" com.mediav.ads.sdk.service.MvService "
                android:enabled="true"
                android:exported="false">
</service>
```

请确保android:exported属性值为false

当您完成了以上的步骤设置后，请参考以下完整的AndroidManifest.xml范例：

```
    ……
     <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission
android:name="android.permission.ACCESS_NETWORK_STATE" />   <uses-
permission android:name="android.permission.READ_PHONE_STATE" />
    <uses-permission
android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    <uses-permission
android:name="android.permission.ACCESS_FINE_LOCATION" />   <uses-
permission android:name="android.permission.ACCESS_WIFI_STATE" />
    <uses-permission android:name="android.permission.GET_TASKS"/>

    <application ……>
    ……
         <activity
            android:name=" com.mediav.ads.sdk.adcore.MvActivity"
            android:configChanges="orientation|screenSize|keyboardHidden" >
         </activity>
          <service android:name="
com.mediav.ads.sdk.service.MvService"
                android:enabled="true"
                android:exported="false">
         </service>
     ……
     </application>
```

步骤三：添加广告位

1. 添加横幅广告

接口定义

public static IMvBannerAd showBanner(ViewGroup adContainer, Activity activity, String adSpaceid, Boolean isTest)

参数说明

adContainer 广告容器
activity 当前活动实例
adSpaceid 广告位标识
isTest 是否测试

实际调用代码

请求广告

```
final String adSpaceid = "网站上获取此ID";
IMvBannerAd bannerad = Mvad.showBanner(adContainer, this, adSpaceid, true);
```

关闭广告

```
bannerad.closeAd(); //设置广告不可见且不会占用空间，不会销毁广告
```

显示广告

```
bannerad.showAd(this); //设置广告为可见，不会重新请求广告
```

设置广告状态监听

```
bannerad. setAdEventListener(IMvAdEventListener listener); //详见步骤四
```

回收广告资源

```
Mvad.activityDestroy(this); //务必在当前activity的onDestroy方法中调用，确保广告资源被回收。
```

注意：

a、IMvBannerAd的LayoutParams为ViewGroup.LayoutParams.MATCH_PARENT, ViewGroup.LayoutParams.WRAP_CONTENT请根据此设置合理设置广告容器

b、横幅广告加载完成后立即显示，广告30s自动刷新。

c、您可以自定义接收广告的容器的位置和尺寸，系统会根据您的设置自动匹配适合的尺寸并显示广告。

2. 添加浮动横幅广告

接口定义

public static IMvFloatbannerAd showFloatbannerAd(Activity activity, String adSpaceid, Boolean isTest, FLOAT_BANNER_SIZE size, FLOAT_LOCATION location)

参数说明

activity 当前活动实例
adSpaceid 广告位标识
isTest 是否测试

size　　广告大小
location　广告位置

```
public enum FLOAT_BANNER_SIZE{
    SIZE_DEFAULT,           //默认
    SIZE_MATCH_PARENT,      //适应屏幕宽
    SIZE_640X100,           //640 x 100
    SIZE_936x120,           //936 x 120
    SIZE_728x90             //728 x 90
}

public enum FLOAT_LOCATION{
    TOP,                    //屏幕顶部
    BOTTOM                  //屏幕底部
}
```

实际调用代码

请求广告

final String adSpaceid = "网站上获取此ID";

IMvFloatbannerAd floatBanner = Mvad.showFloatbannerAd(this, adSpaceid, true, Mvad.FLOAT_BANNER_SIZE.SIZE_DEFAULT, Mvad.FLOAT_LOCATION.BOTTOM);

关闭广告

Mvad.closeFloatbannerAd(this) 或 floatBanner.closeAd();　//关闭并销毁广告

设置广告状态监听

floatBanner. setAdEventListener(IMvAdEventListener listener); //详见步骤四

回收广告资源

Mvad.activityDestroy(this); //务必在当前activity的onDestroy方法中调用，确保广告资源被回收。

注意：浮动窗广告加载完成后立即显示，广告30s自动刷新。

3. 添加插屏广告

接口定义

public static IMvInterstitialAd showInterstitial(Activity activity, String adSpaceid, Boolean isTest)

参数说明

activity　当前活动实例
adSpaceid　广告位标识
isTest　　是否测试

实际调用代码

请求广告

final String adSpaceid = "网站上获取此ID";

IMvInterstitialAd interstitialAd = Mvad.showInterstitial(this, adSpaceid, true);

关闭广告

Mvad. closeInterstitial(this) 或 interstitialAd.closeAd(); //关闭并销毁广告

显示广告

interstitialAd.showAd(this); //重新请求广告

设置广告状态监听

interstitialAd. setAdEventListener(IMvAdEventListener listener); //详见步骤四

回收广告资源

Mvad.activityDestroy(this); //务必在当前activity的onDestroy方法中调用，确保广告资源被回收。

注意：

a. 插屏广告加载完成后立即显示。
b. 插屏广告默认不自动刷新，点击广告关闭按钮只是把广告隐藏，当需要对插屏进行刷新时，可以先调用关闭接口再显示广告。

4. 添加开屏广告

接口定义

public static void showSplashAd(ViewGroup adContainer, Activity activity, String adSpaceid, IMvAdEventListener listener, Boolean showCountdown, Boolean isTest)

参数说明

adContainer 广告容器
activity 当前活动实例
adSpaceid 广告位标识
listener 广告状态监听
showCountdown 是否显示倒计时
isTest 是否测试

实际调用代码

请求广告

```
final String adSpaceid = "网站上获取此ID";
Mvad.showSplashAd (adContainer, this, adSpaceid, new IMvadEventListener () {

    //获取广告成功
    public void onAdviewGotAdSucceed();

   //获取广告失败
    public void onAdviewGotAdFail();

   //广告渲染完成
    public void onAdviewRendered();

    //进入落地页
    public void onAdviewIntoLandpage();
```

```
        //离开落地页
        public void onAdviewDismissedLandpage();

        //广告被点击
        public void onAdviewClicked();

        //广告被关闭，开屏广告结束时会回调此方法
        public void onAdviewClosed();

        //当广告实例被销毁
        public void onAdviewDestroyed();
}, true, true);
```

回收广告资源

Mvad.activityDestroy(this); //务必在当前activity的onDestroy方法中调用，确保广告资源被回收。

注意：

a. 开屏广告加载完成后立即显示。
b. 开屏广告不支持开发者自定义倒计时时间。
c. 开屏广告倒计时结束会回调listener的onAdviewClosed方法，请复写此方法执行自定义操作。

5. 添加嵌入式广告

接口定义

public static IMvNativeBannerAd showNativeBanner(ViewGroup adContainer, Activity activity, String adSpaceid, Boolean isTest)

参数说明

adContainer 广告容器
activity 当前活动实例
adSpaceid 广告位标识
isTest 是否测试

实际调用代码

请求广告

```
final String adSpaceid = "网站上获取此ID";
IMvNativeBannerAd nativebannerad = Mvad. showNativeBanner (adContainer, Activity.this, adSpaceid, true);
```

关闭广告

```
nativebannerad.closeAd(); //设置广告不可见且不会占用空间，不会销毁广告
```

显示广告

```
nativebannerad.showAd(this); //设置广告为可见，不会重新请求广告
```

设置广告状态监听

nativebannerad. setAdEventListener(IMvAdEventListener listener); //详见步骤四

回收广告资源

Mvad.activityDestroy(this); //务必在当前activity的onDestroy方法中调用，确保广告资源被回收。

注意：

a、IMvNativeBannerAd的LayoutParams为ViewGroup.LayoutParams.MATCH_PARENT, ViewGroup.LayoutParams.MATCH_PARENT

b、嵌入式广告加载完成后立即显示。

c、您可以自定义接收广告的容器的位置和尺寸，系统会根据您的设置自动匹配适合的尺寸并显示广告。

6. 添加原生广告

接口定义

```
public static IMvNativeAdLoader initNativeAdLoader(Activity activity, String
adSpaceid, IMvNativeAdListener listener, Boolean isTest)
```

参数说明

activity　当前活动实例

adSpaceid　广告位标识

listener　原生广告状态监听

isTest　是否测试

初始化原生广告加载器

```
final String adSpaceid = "网站上获取此ID";
IMvNativeAdLoader MvNativeLoader = Mvad.initNativeAdLoader(
this, adSpaceid, new IMvNativeAdListener() {
            @Override
            public void onNativeAdLoadSucceeded(ArrayList<IMvNativeAd> nativeAds)
{
                // 原生广告请求成功的回调在此执行
                MvNativeAds.addAll(nativeAds);
            }

            @Override
            public void onNativeAdLoadFailed() {
                // 原生广告请求失败的回调在此执行
            }
        }, true);
```

注意：初始化之后并不会请求广告，要请求广告请参考下一步：请求原生广告。

请求原生广告

```
IMvNativeAdLoader.loadAds();
```

调用此方法即可请求广告，请求完成之后由初始化传入的回调方法进行处理。如果要请求更多的广告，只需要再执行一次loadAds方法即可，不用重复初始化。

如果要设置原生广告请求的关键词信息，在请求广告之前可以调用IMvNativeAdLoader. setKeywords方法，完整代码实例如下：

```
HashSet<String> keywordList = new HashSet<String>();
keywordList.add("新闻");
keywordList.add("国内新闻");
nativeAdLoader.setKeywords(keywordList);
```

如果要清除设置的关键词信息，可以调用IMvNativeAdLoader. clearKeywords方法。

注意：setKeywords和clearKeywords方法都应该在调用loadAds之前执行。

如果要设置其他参数，在请求广告之前可以调用IMvNativeAdLoader. setAdAttributes()方法，完整实例代码如下：

```
MvProductAdAttributes attr = new MvProductAdAttributes();
attr.setCategory("3C", 0);
attr.setPrice(100);
nativeAdLoader.setAdAttributes(attr);
```

目前只支持通过MvProductAdAttributes类设置商品参数，可以传递商品的类目和价格。
如果要清除设置的参数，可以调用IMvNativeAdLoader. clearAdAttributes()方法。

注意：setAdAttributes和clearAdAttributes方法都应该在调用loadAds之前执行。

获取广告内容

IMvNativeAd.getContent();

调用此方法即获得广告内容。获取的广告内容为JSONObject对象，具体的信息包括：

```
{
    "logo": "http://material.mediav.com/mba/ad/3.png",
    "title": "别克君威",
    "desc": "精准操控，动感在握！",
    "contentimg": "http://material.mediav.com/mba/ad/4.jpg",
    "btntext": "试驾",
    "ext_text": "扩展字段"
}
```

logo:LOGO图片地址
title：标题
desc：描述
contentimg：广告图片地址
btntext：按钮文字
ext_text：扩展字段，目前多用来表示副标题

广告曝光

```
IMvNativeAd.onAdShowed();
```

广告展示之后调用此方法进行上报曝光，否则广告记录不到曝光；

广告点击

```
IMvNativeAd.onAdClicked();
```

广告被点击之后调用方法进行点击处理。

步骤四：追踪广告状态（可选）

每个返回当前广告实例的广告类型可以为其设置状态监听，如果需要追踪广告：加载成功、加载失败、被点击、被关闭（插屏）、进入落地页、离开落地页等事件请添加以下代码。例如：

```
IMvFloatbannerAd mvadAdView = Mvad.showFloatbannerAd(this, adSpaceid,
true,    Mvad.FLOAT_BANNER_SIZE.SIZE_DEFAULT,
Mvad.FLOAT_LOCATION.BOTTOM);

IMvBannerAd  mvadAdView=Mvad.showBanner(adContainer,this, adSpaceid,
true);

IMvInterstitialAd mvadAdView = Mvad.showInterstitial(this, adSpaceid,
true);

mvadAdView. setAdEventListener (new IMvadEventListener () {

        //获取广告成功
        public void onAdviewGotAdSucceed();

       //获取广告失败
        public void onAdviewGotAdFail();

       //广告渲染完成
        public void onAdviewRendered();

        //进入落地页
        public void onAdviewIntoLandpage();

        //离开落地页
        public void onAdviewDismissedLandpage();

        //广告被点击
        public void onAdviewClicked();

        //广告被关闭
        public void onAdviewClosed();

        //当广告实例被销毁
        public void onAdviewDestroyed();
});
```

步骤五：测试广告

打开调试LOG

Mvad.setLogSwitch(Context context, boolean state);

说明：SDK的LOG默认是关闭的，通过此开关可以打开LOG展示。

您可以将isTest设置为true来测试广告效果及兼容性。在测试环境中，广告点击不扣费。当您在360移动开放平台上传应用时，需要将isTest设置为false，这样才能正常接收商业广告的投放。只有商业广告才能给您带来收益。

备注：isTest参数一般为广告接口中的最后一个参数

步骤六：扩展功能

1. 自定义内开落地页浏览器接口

针对内开落地页类型广告，SDK默认使用内置浏览器处理内开广告点击跳转。通过此接口开发者可用自定义的浏览器替换SDK的内置浏览器。

Mvad. setLandingPageView(Context context, IMvLandingPageView landingPageView);

在APP起始页的onCreate或Application的onCreate调用此方法设置SDK是否要使用APP的自定义浏览器实现。

context:上下文对象。

landingPageView：自定义落地页浏览器实现对象。传入null使用SDK默认的落地页浏览器。

IMvLandingPageView

开发者要实现的内开落地页浏览器接口

open(Context context,String url, IMvLandingPageListener listener)

当开发者通过Mvad.setLandingPageView方法设置了自定义落地页浏览器后，在点击内开浏览器广告时由SDK触发。

context：当前上下文。

url:要打开的落地页地址。如果此参数为null表示此功能被远程禁用，将使用内置浏览器打开落地页。

listener：浏览器事件通知回调。APP开发者通过调用该实例的方法通知SDK自定义浏览器的状态。

IMvLandingPageListener

浏览器事件通知回调。

onPageClose:在浏览器被关闭时触发，必须在主线程中调用。

onPageLoadFinished:在落地页加载完毕时触发，必须在主线程中调用。

onPageLoadFailed:在浏览器打开失败或落地页加载失败时触发，必须在主线程中调用。

onAppDownload:在浏览器处理App下载时触发，返回true表示SDK已处理该下载请求，返回false表示SDK未处理，必须在主线程中调用。

注：在使用自定义落地页功能时，请务必正确调用onAppDownload方法，否则会使SDK的App下载功能失效，造成大量广告收入损失

原生广告点击回调IMvNativeAdOnClickListener
onDownloadConfirmed：下载弹窗二次确认点击
onDownloadCancelled：下载弹窗取消
onLandingpageOpened：落地页打开
onLandingpageClosed：落地页关闭
onOutsideBrowserOpen：外开落地页打开

附录:

广告位尺寸说明

广告类型	宽高比	尺寸
横幅广告 Banner Ads	6.4：1	320×50
		480×75（HD1.5）
		640×100（HD2.0）
		1280×200（HD4.0）
	7.8：1	468×60
		702×90（HD1.5）
		936×120（HD2.0）
		1872×240（HD4.0）
	8.1：1	728×90
		1092×135（HD1.5）
		1456×180（HD2.0）
		2912×360（HD4.0）
插屏广告 Interstitial Ads	6：5	300×250
		450×375（HD1.5）
		600×500（HD2.0）
		1200×1000（HD4.0）
	9：16	720×1280
		1080×1920（HD1.5）
		1440×2560（HD2.0）
		2880×5120（HD4.0）
	3：5	480×800
		720×1200（HD1.5）
		960×1600（HD2.0）
		1920×2400（HD4.0）

Android M下的SDK使用

在AndroidMainfest.xml现有的基础上需要添加使用权限：
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION”/>
注：目前针对6.0的测试仅局限于模拟器测试。