Avazu Native Ad SDK Setup for Android

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Version | Description | Developer |
| 2017.04.26 | v2.2.5 | Improve Appwall | 卢红伍 |
| 2017.07.04 | v2.2.6 | Support native ad | 卢红伍 |
| 2017.08.14 | v2.2.7 | Support advanced native ad | 卢红伍 |
| 2017.08.30 | v2.2.7 | Support reward ad | 卢红伍 |

目录

[1. 简介 3](#_Toc492302215)

[1.1 SDK功能介绍 3](#_Toc492302216)

[1.2 兼容性 3](#_Toc492302217)

[2. 集成准备 3](#_Toc492302218)

[2.1 APP ID 3](#_Toc492302219)

[2.2 Unit ID 3](#_Toc492302220)

[3. 初始化SDK 3](#_Toc492302221)

[3.1 AndroidMenifest.xml 配置 （可选） 5](#_Toc492302222)

[3.2添加权限（可选） 8](#_Toc492302223)

[3.3初始化 8](#_Toc492302224)

[4. 广告墙接入 8](#_Toc492302225)

[4.1 配置 9](#_Toc492302226)

[4.2 预加载应用墙数据 12](#_Toc492302227)

[4.3 启动market页面 13](#_Toc492302228)

[5. 原生广告接入 14](#_Toc492302229)

[5.1接入流程 14](#_Toc492302230)

[6. 高级原生广告接入 15](#_Toc492302231)

[6.1 接入流程 15](#_Toc492302232)

[7.激励视频接入 17](#_Toc492302233)

[7.1 接入流程 17](#_Toc492302234)



1. 简介

1.1 SDK功能介绍

Avazu ADSDK 能够提供以下功能：

* AppWall （广告墙）
* Native (原生广告)
* Reward (激励视频)

1.2 兼容性

支持Android 2.3（API level 9）及以上系统

2. 集成准备

2.1 APP ID

确保您已注册Avazu APX账号且已有可用的APP ID 用来展示广告

（参照aNative用户使用手册）

2.2 Unit ID

在Avazu APX创建一个广告位后，系统会自动生成Unit id

（参照aNative用户使用手册）

1. 初始化SDK

Avazu SDK可以从 Avazu 运营获取，SDK包含多个aar包

**aar包名** **作用** **是否必须**

* anative\_common.aar 公共基础包 是
* anative\_appwall.aar appwall广告的功能包 否
* anaitve\_native.aar 原生广告功能包 否
* anative\_reward.aar 激励视频功能包 否

将aar文件保存到本地工程中:

* 将所需要的aar和jar包拷贝到项目的libs文件夹中，并作为项目的依赖库
* 在项目的build.gradle文件中的dependencies添加：

|  |
| --- |
| compile 'com.google.code.gson:gson:2.8.0' // 必须  compile(name: anative\_common, ext: 'aar') // 必须，基础包  compile(name: anative\_appwall, ext: 'aar') // 可选，用于应用墙  compile(name: anative\_native, ext: 'aar') // 可选，用于原生广告  compile(name: anative\_reward, ext: 'aar') // 可选，用于激励视频  compile 'com.google.android.gms:play-services-ads:8.4.0' // 可选，Admob广告，用于原生广告或激励视频  compile 'com.facebook.android:audience-network-sdk:4.23.0' // 可选，Fb 广告，用于原生广告  compile 'com.adcolony:sdk:3.1.2' // 可选，Adcolony广告，用于激励视频  compile(name: 'unity-ads', ext: 'aar') // 可选，Unity 广告，用于激励视频 |

* 在项目的build.gradle文件中添加(可选， 第三方sdk)：

|  |
| --- |
| repositories {  flatDir {  dirs 'libs'  }  maven {  url <https://adcolony.bintray.com/AdColony> // 可选，adcolony广告，用于激励视频  }  } |

* 在项目的libs目录添加(可选，第三方sdk)：

|  |
| --- |
| applovin-sdk-6.4.2.jar // 可选，Applovin 广告，用于激励视频  dagger-2.7.jar // 可选，Vungle 广告，用于激励视频  javax.inject-1.jar // 可选，Vungle 广告，用于激励视频  publisher-sdk-android-5.1.0.jar // 可选，Vungle 广告，用于激励视频  rxjava-1.2.0.jar // 可选，Vungle 广告，用于激励视频  eventbus-2.2.1.jar // 可选，Vungle 广告，用于激励视频 |

## 3.1 AndroidMenifest.xml 配置 （可选）

|  |
| --- |
| ***<!—admob， 若集成paly-services-ads, 则添加-->***  <activity  android:name="com.google.android.gms.ads.AdActivity"  android:configChanges="keyboard|keyboardHidden|orientation|screenLayout|uiMode|screenSize|smallestScreenSize"  android:theme="@android:style/Theme.Translucent" />  <meta-data  android:name="com.google.android.gms.version"  android:value="@integer/google\_play\_services\_version" />  ***<!—facebook，若集成 audience-network-sdk , 则添加-->***  <activity  android:name="com.facebook.ads.InterstitialAdActivity"  android:configChanges="keyboardHidden|orientation|screenSize"  android:excludeFromRecents="true"  android:exported="true"  android:noHistory="true"/>  ***<!—adcolony, 若集成adcolony sdk, 则添加-->***  <activity  android:name="com.adcolony.sdk.AdColonyInterstitialActivity"  android:configChanges="keyboardHidden|orientation|screenSize"  android:hardwareAccelerated="true"/>  <activity  android:name="com.adcolony.sdk.AdColonyAdViewActivity"  android:configChanges="keyboardHidden|orientation|screenSize"  android:hardwareAccelerated="true"/>  ***<!--vungle若集成vungle sdk, 则添加-->***  <activity  android:name="com.vungle.publisher.VideoFullScreenAdActivity"  android:configChanges="keyboardHidden|orientation|screenSize|screenLayout|smallestScreenSize"  android:theme="@android:style/Theme.NoTitleBar.Fullscreen"/>  <activity  android:name="com.vungle.publisher.MraidFullScreenAdActivity"  android:configChanges="keyboardHidden|orientation|screenSize|screenLayout|smallestScreenSize"  android:theme="@android:style/Theme.Translucent.NoTitleBar.Fullscreen"/>  <activity  android:name="com.vungle.publisher.FlexViewAdActivity"  android:configChanges="keyboardHidden|orientation|screenSize|screenLayout|smallestScreenSize"  android:theme="@android:style/Theme.Translucent.NoTitleBar.Fullscreen"/>  ***<!-- AppLovin: 若集成applovin sdk, 则添加-->***  <activity  android:name="com.applovin.adview.AppLovinInterstitialActivity"  android:configChanges="orientation|screenSize"/>  <activity  android:name="com.applovin.adview.AppLovinConfirmationActivity"  android:configChanges="orientation|screenSize"/> |

## 3.2添加权限（可选）

部分广告主需要SDK回传device id，建议开发者添加 android.persmission.READ\_PHONE\_STATE 便于SDK获取设备id。

## 3.3初始化

在Application的onCreate里面调用该初始化方法:

|  |
| --- |
| Adsdk.initialize(Context context, String appId, IAdSdkListener listener); |

参数说明：

* context: Application 上下文
* appId: 申请的应用ID
* listener: IAdSdkLitener 用于监听初始化是否成功，建议成功后再调用后续接口，如设置应用墙样式。

注意：初始化接口必须在进程的主线程里面初始化！

# 4. 广告墙接入

App Market接口能够通过简单的设定，获取一个完整的广告商店的Activity或者Fragment，其界面如下：



4.1 配置

您可以根据需求配置App Market

1. 设置商店名称：

|  |
| --- |
| public static void setAppMarketName(Context context, String name) |

参数说明：

context: 上下文

name: 您所希望的商店名，将显示在商店的左上角，若不调用该接口，默认商店名为：“Market”

2. 配置广告墙样式（建议在sdk初始化成功回调中调用）

|  |
| --- |
| public static void setMarketStyle (Context context, HashMap<String, Integer> marketStyle) |

参数说明：

context: 上下文

marketStyle : 广告墙样式配置，**参数为resource id**, 可以配置项如下（若某一项设置为null，表示该项用默认配置）：

TITLE\_BACKGROUND\_COLOR 标题栏背景色

TITLE\_TEXT\_COLOR 标题栏字体颜色

TITLE\_TEXT\_SIZE 标题栏字体大小

TITLE\_BAR\_HEIGHT 标题栏高度

TITLE\_BACK\_DRAWABLE 标题栏返回按钮图片 （默认图片大小 20x36 pixel）

TABLE\_BACKGROUND\_COLOR Table栏背景色

TABLE\_TEXT\_COLOR Table栏字体颜色

TABLE\_TEXT\_SIZE Table栏字体大小

TABLE\_BAR\_HEIGHT Table栏高度

TABLE\_INDICATOR\_COLOR Table栏指示器颜色

DK\_BUTTON\_BACKGROUND\_COLOR 大卡广告按钮背景色

DK\_BUTTON\_TEXT\_COLOR 大卡广告按钮字体颜色

INSTALL\_TEXT\_BACKGROUND\_DRAWABLE 安装按钮背景图片 （默认：140x60 pixel）

INSTALL\_TEXT\_COLOR 安装按钮字体颜色

STATUS\_COLOR 状态栏颜色

NAVIGATION\_COLOR 导航栏颜色

WALL\_BACKGROUND\_COLOR 应用墙内容背景色

AD\_TITLE\_TEXT\_COLOR 广告标题字体颜色

AD\_DESCRIPTION\_TEXT\_COLOR 广告描述字体颜色

CATEGORY\_TEXT\_COLOR 广告分类字体颜色

CATEGORY\_TEXT\_SIZE 广告分类字体大小

CATEGORY\_OF\_RECOMMEND  推荐分类名称

CATEGORY\_OF\_POPULAR  流行分类名称

CATEGORY\_OF\_LIKE  喜欢分类名称

AD\_CLICK\_COVER\_LAYER\_TRANS\_BACKGROUND 广告点击时界面蒙层颜色



使用示例（在sdk初始化成功回调中调用）：

|  |
| --- |
| public class MyApplication extends Application {  @Override  public void onCreate() {  AdSdk.initialize(context, appid, new IAdSdkListener() {  @Override  public void onInitSuccess() {  // set market style  setMarketStyle();  }  @Override  public void onInitFailed(String msg) {  }  });  }  private void setMarketStyle() {  marketStyle.put(Constants.MarketStyle.TITLE\_BACKGROUND\_COLOR, R.color.white);  marketStyle.put(Constants.MarketStyle.TITLE\_TEXT\_COLOR, R.color.gray);  marketStyle.put(Constants.MarketStyle.TITLE\_BAR\_HEIGHT, R.dimen.title\_bar\_height);  marketStyle.put(Constants.MarketStyle.TITLE\_TEXT\_SIZE, R.dimen.anative\_appwall\_title\_text\_size);  marketStyle.put(Constants.MarketStyle.TITLE\_BACK\_DRAWABLE, R.drawable.apx\_appwall\_adrss\_ic\_back);  marketStyle.put(Constants.MarketStyle.TABLE\_BACKGROUND\_COLOR, R.color.white);  marketStyle.put(Constants.MarketStyle.TABLE\_TEXT\_COLOR, R.color.gray);  marketStyle.put(Constants.MarketStyle.TABLE\_TEXT\_SIZE, R.dimen.table\_text\_size);  marketStyle.put(Constants.MarketStyle.TABLE\_INDICATOR\_COLOR, R.color.light\_gray);  marketStyle.put(Constants.MarketStyle.TABLE\_BAR\_HEIGHT, R.dimen.table\_bar\_height);  marketStyle.put(Constants.MarketStyle.DK\_BUTTON\_BACKGROUND\_COLOR, R.color.black);  marketStyle.put(Constants.MarketStyle.DK\_BUTTON\_TEXT\_COLOR, R.color.white);  marketStyle.put(Constants.MarketStyle.INSTALL\_TEXT\_BACKGROUND\_DRAWABLE, R.drawable.anative\_appwall\_adress\_button\_type1);  marketStyle.put(Constants.MarketStyle.INSTALL\_TEXT\_COLOR, R.color.gray);  marketStyle.put(Constants.MarketStyle.STATUS\_COLOR, null);  marketStyle.put(Constants.MarketStyle.NAVIGATION\_COLOR, null);  marketStyle.put(Constants.MarketStyle.WALL\_BACKGROUND\_COLOR, null);  marketStyle.put(Constants.MarketStyle.AD\_TITLE\_TEXT\_COLOR, R.color.gray);  marketStyle.put(Constants.MarketStyle.AD\_DESCRIPTION\_TEXT\_COLOR, R.color.light\_gray);  marketStyle.put(Constants.MarketStyle.AD\_CLICK\_COVER\_LAYER\_TRANS\_BACKGROUND, R.color.cover\_layer\_background);  marketStyle.put(Constants.MarketStyle.CATEGORY\_TEXT\_COLOR, R.color.gray);  marketStyle.put(Constants.MarketStyle.CATEGORY\_TEXT\_SIZE, R.dimen.category\_text\_size);  marketStyle.put(Constants.MarketStyle.CATEGORY\_OF\_RECOMMEND, R.string.recommend);  marketStyle.put(Constants.MarketStyle.CATEGORY\_OF\_POPULAR, R.string.popular);  marketStyle.put(Constants.MarketStyle.CATEGORY\_OF\_LIKE, R.string.like);  AdSdk.setMarketStyle(this, marketStyle);  }  } |

4.2 预加载应用墙数据

为提高market页面的加载速度，以及优化广告的填充和转化，强烈建议开发者预加载应用墙的数据：

|  |
| --- |
| public static void preloadMarketData (Context context) |

参数说明：

* context: 上下文，建议使用ApplicationContext

使用示例：

|  |
| --- |
| @Override  protected void onResume() {  super.onResume();  AdSdk.preloadMarketData(this.getApplicationContext());  }  } |

4.3 启动market页面

有两种方式启动market页面：

1. 用Activity启动：

在需要打开App market处调用该接口

|  |
| --- |
| public static void showAppMarket(Context context) |

2. 用Fragment启动：

在打开market fragment前需要调用

|  |
| --- |
| public static void setMarketFragmentMode(Context context, boolean isFragmentMode) |

参数说明：

context: 上下文

isFragmentMode: 设为true则为fragment接入模式

在需要获取Fragment对象的地方用如下方法获取

|  |
| --- |
| Fragment fr = AdSdk.getFeatureFragment(Context context); |

5. 原生广告接入

原生广告按尺寸分为大中小三种样式，其中大图样式又包含三种形式（大图，视频，轮播），若开发者同时设置可以展示大图的三种形式，那么出大图广告优先级将会是：轮播 > 视频 >大图。集成后的可能效果如下：



## 5.1接入流程

请将anative\_common.aar, anative\_native.aar复制到相应位置

1. 若集成第三方广告源， 请在build.gradle文件中增加相应依赖并配置AndroidMenifest.xml

2. 初始化和预加载

构建一个NativeAd 对象，并传入该广告位unit ID，设置广告加载监听；展示广告之前，提前将广告拉到本地，提升用户体验，提高广告收入

|  |
| --- |
| INativeAd nativeAd = new NativeAd(Context activityContext, String unitId);  nativeAd.setNativeAdListener(new NativeAdListener() {  @Override  public void onAdLoaded() { // 广告加载成功回调  }  @Override  public void onError(String error) {  }  @Override  public void onClicked() { //只支持记录APX广告点击，不支持记录第三方广告点击  }  })  mNativeAd.load(); // 加载广告 |

3. 显示广告

当广告预加载成功后，调用接口show展示广告；adContainer 是用来展示原生广告的容器。

|  |
| --- |
| if (nativeAd.isLoaded()) {  nativeAd.show(ViewGroup adContainer); // 显示广告  } |

广告容器配置参考：

|  |
| --- |
| <FrameLayout  android:id="@+id/native\_container"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"/> |

# 6. 高级原生广告接入

高级原生定义： sdk 提供获取广告原始数据的接口，由开发者自定义UI并绑定数据。

## 6.1 接入流程

1. 所需aar包和普通原生广告一致

2. 初始化和预加载

构建一个AdvancedNativeAd 对象，并传入该广告位unit ID，设置广告加载监听和需要加载广告的数量，最少1个，最多10个；展示广告之前，提前将广告拉到本地，提升用户体验，提高广告收入。

|  |
| --- |
| mAdvancedNativeAd = new AdvancedNativeAd(this, unitId);  mAdvancedNativeAd.setAdListener(new INativeAdLoadListener() {  @Override  public void onAdListLoaded(List<INativeAd> ads) {  // 注意： 若load 时设置的加载数量>1, 实际返回的数量可能会小于请求的  数量；多个广告的来源相同, 比如都来源于APX  }  @Override  public void onError(String error) { //加载错误  }  });  mAdvancedNativeAd.load(1); // 加载的广告数量范围必须是[1, 10] |

3. 显示广告（单个广告）

根据广告来源的不同（Apx, Admob, Facebook）分别处理，详情参考demo 源码。

|  |
| --- |
| private void showAd() {  switch (ad.getAdType()) {  case Constants.NativeAdType.AD\_SOURCE\_APX:  inflateApxAdView(ad);  break;  case Constants.NativeAdType.AD\_SOURCE\_ADMOB\_INSTALL:  inflateAdmobInstallAdView(ad);  break;  case Constants.NativeAdType.AD\_SOURCE\_ADMOB\_CONTENT:  inflateAdmobContentAdView(ad);  break;  case Constants.NativeAdType.AD\_SOURCE\_FACEBOOK:  inflateFbNativeAdView(ad);  break;  }  }  // 显示APX 广告  private void inflateApxAdView(INativeAd ad) {  View adView = LayoutInflater.from(mContext).inflate(R.layout.advanced\_native\_ad\_apx, null);  View transitionView = LayoutInflater.from(mContext).inflate(R.layout.anative\_native\_ad\_transition\_view, null);  ApxMediaView nativeAdMedia = (ApxMediaView) adView.findViewById(R.id.anative\_mediaview);  nativeAdMedia.setNativeAd((AdInfo) ad.getAdObject());  nativeAdMedia.setAutoPlay(true);  BasicLazyLoadImageView iconView = (BasicLazyLoadImageView) adView.findViewById(R.id.anative\_ad\_icon\_image);  iconView.requestDisplayURL(ad.getIconImageUrl());  TextView titleView = (TextView) adView.findViewById(R.id.anative\_ad\_title);  titleView.setText(ad.getTitle());  TextView subtitleView = (TextView) adView.findViewById(R.id.anative\_ad\_subtitle\_text);  subtitleView.setText(ad.getBody());  TextView ctaView = (TextView) adView.findViewById(R.id.anative\_ad\_cta\_text);  ctaView.setText(ad.getCallToActionText());  BasicLazyLoadImageView choiceIconImage = (BasicLazyLoadImageView) adView.findViewById(R.id.anative\_ad\_choices\_image);  choiceIconImage.setImageResource(R.drawable.apx\_native\_ad\_choices);  ad.registerViewForInteraction(mAdContainer);  ad.registerTransitionViewForAdClick(transitionView);  mAdContainer.removeAllViews();  mAdContainer.addView(adView);  }  // admob 安装类广告  private void inflateAdmobInstallAdView(ad) {}  // admob 内容类广告  private void inflateAdmobContentAdView(ad) {}  // facebook 广告  private void inflateFbNativeAdView(ad) {} |

4. 显示广告（多个广告）

如果成功获取到相同源的多个广告，开发者可以自定义展现形式，比较常见的是轮播广告，可以参考demo中普通原生广告中轮播样式实践。

# 7.激励视频接入

激励视频目前聚合了主流的几家平台，包括Vungle, Unity, Admob, Applovin, Adcolony。集成后效果如下：



## 7.1 接入流程

请将anative\_common.aar, anative\_reward.aar复制到相应位置，并在build.gradle文件中增加相应三方广告sdk依赖(或者libs里面添加), 并配置AndroidMenifest.xml

第三方平台sdk依赖(在build.gradle添加)：

|  |
| --- |
| dependencies {  compile 'com.google.android.gms:play-services-ads:8.4.0' // Admob  compile 'com.facebook.android:audience-network-sdk:4.23.0' // Facebook  compile 'com.adcolony:sdk:3.1.2' // Adcolony  compile(name: 'unity-ads', ext: 'aar') // Unity  } |

第三方平台sdk依赖(在libs目录添加, 源文件在demo里面获取)：

|  |
| --- |
| unity-ads.aar // Unity  applovin-sdk-6.4.2.jar // Applovin  dagger-2.7.jar // Vungle  javax.inject-1.jar // Vungle  publisher-sdk-android-5.1.0.jar // Vungle  rxjava-1.2.0.jar // Vungle  eventbus-2.2.1.jar // Vungle |

1. 初始化和预加载

构建一个RewardedVideoAd对象，并传入该广告位unit ID，设置广告加载监听, 展示广告之前，提前将广告拉到本地，提升用户体验，提高广告收入。

|  |
| --- |
| rewardedVideoAd = new RewardedVideoAd(this, unitId, new RewardedVideoAdListener() {  @Override  public void onInitSuccess() {  // 初始化成功回调，建议这里调用loadAd 接口加载广告  rewardedVideoAd.loadAd();  }  @Override  public void onInitFailed() {  }  @Override  public void onRewardedVideoAdLoaded() {  // 广告加载成功回调，建议这里调用show接口显示广告  }  @Override  public void onRewardedVideoAdOpened() {  }  @Override  public void onRewardedVideoStarted() {  }  @Override  public void onRewardedVideoAdClosed() {  }  @Override  public void onRewarded(RewardItem var1) {  }  @Override  public void onRewardedVideoAdLeftApplication() {  }  @Override  public void onRewardedVideoAdFailedToLoad(int var1) {  }  }); |

1. 显示激励广告

可在广告加载成功回调onRewardedVideoAdLoaded() 中调用show() 显示广告

|  |
| --- |
| public void onRewardedVideoAdLoaded() {  // AD load succeed callback, should call show() to display AD  if (rewardedVideoAd.isLoaded()) {  rewardedVideoAd.show();  }  } |