Avazu Android Ad SDK Integration Documentation

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Version | Description | Developer |
| 2017.04.26 | v2.2.5 | Improved Appwall | Hongwu Lu |
| 2017.07.04 | v2.2.6 | Supported native ad | Hongwu Lu |
| 2017.08.14 | V2.2.7 | Support advanced native ad | Hongwu Lu |

Content

[1. Introduction 3](#_Toc493499159)

[1.1. SDK Introduction 3](#_Toc493499160)

[1.2. Requirements 3](#_Toc493499161)

[2. Apply App ID and Ad Unit ID 3](#_Toc493499162)

[2.1. App ID 3](#_Toc493499166)

[2.2. Ad Unit ID 3](#_Toc493499167)

[3. Initialization SDK 3](#_Toc493499168)

[3.1. Obfuscated code configuration 4](#_Toc493499170)

[3.2. Initialization 4](#_Toc493499171)

[3.3. Modify your AndroidMenifest.xml 5](#_Toc493499172)

[3.4 Add Permissions [Optional] 5](#_Toc493499173)

[4. Integration with Appwall 5](#_Toc493499174)

[4.1 Configration 6](#_Toc493499179)

[4.2 Preload market data 8](#_Toc493499180)

[4.3 Start App Market 9](#_Toc493499181)

[5. Integration with Native Ads 9](#_Toc493499182)

[5.1 Configuration 10](#_Toc493499185)

[6. Integration with Advanced Native Ads 11](#_Toc493499186)

[6.1 Configuration 11](#_Toc493499187)

[7. Reward Video AD 12](#_Toc493499188)

[7.1 Integrated with reward video ads 12](#_Toc493499189)



# Introduction

## SDK Introduction

Use Avazu Android Ad SDK to maximize your app’s revenue streams and save time, support below ad formats:

* Appwall
* Native Ad

## Requirements

Android 2.3 (API Version 9) and up.

# Apply App ID and Ad Unit ID



## App ID

Make sure you have registered on Avazu APX, you will get an App ID after creating your own app.

## Ad Unit ID

You can create an ad unit under the app, you will get an Ad Unit ID after creating the ad unit.

# Initialization SDK

Below are the aar packages in the SDK:

**aar package** **Function** **Required**

* avazu\_common.aar public base package Yes
* avazu\_appwall.aar appwall ads package No
* avazu\_native.aar native ads package No

Add aar packages to your local project:

Add necessary aar packages under the ‘libs‘ folder of your project, and they will be served as a dependency library for the project.

* Some code show blow should be added into the dependencies of the project’s build.gradle file:

compile 'com.google.code.gson:gson:2.8.0' // Required

compile(name: avazu\_common, ext: 'aar') // Required

compile(name: avazu\_appwall, ext: 'aar') // Optional, used in Appwall

compile(name: avazu\_native, ext: 'aar') // Optional, used in Native Ad

compile 'com.google.android.gms:play-services-ads:8.4.0' // Optional, Admob Native Ad compile 'com.facebook.android:audience-network-sdk:4.23.0' // Optional, Facebook Native Ad



## Obfuscated code configuration

Some code should be added in your profuard file according to the integrated package:

# Facebook Native Ad, Optional

-keep class com.facebook.\*\* {\*;}

-dontwarn com.facebook.\*\*

# Admob Native Ad, Optional

-keep class com.google.android.gms.\*\* {\*;}

-dontwarn com.google.android.gms.\*\*

## Initialization

Call the initialization method in Application’s onCreate:

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

Parameter description:

* context: Application Context.
* appId: App ID that generated after creating your App.
* listener: IAdSdkLitener for listening init result.

NOTICE：Adsdk.initialize() must be called in main thread!

## Modify your AndroidMenifest.xml

If you had integrated with paly-services-ads, please add below configuration in your AndroidMenifest.xml:

|  |
| --- |
| <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" /> |

## 3.4 Add Permissions [Optional]

Some advertisers need sdk feedback the device id, so we recommond developers add permission: android.persmission.READ\_PHONE\_STATE in your convenience.

# Integration with Appwall

A complete ad market’s Activity or Fragment can be returned via the App Market interface with simple settings, below is the sample UI:





## 4.1 Configration

You can customize App Market according to your product requirement:

1. Set market name：

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

Parameter description：

* context: Application context
* name: the name of your App Market page (default value is ：“Market”)

2. Configurate App Market style(suggest configurate in the callback of initialize):

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

Parameter description：

* context: Application context
* marketStyle : support to config below style, **value is resource id.** If set to null, mean default value：
* TITLE\_BACKGROUND\_COLOR
* TITLE\_TEXT\_COLOR
* TITLE\_TEXT\_SIZE
* TITLE\_BAR\_HEIGHT
* TITLE\_BACK\_DRAWABLE (default: 20x36 pixel)
* TABLE\_BACKGROUND\_COLOR
* TABLE\_TEXT\_COLOR
* TABLE\_TEXT\_SIZE
* TABLE\_BAR\_HEIGHT
* TABLE\_INDICATOR\_COLOR
* DK\_BUTTON\_BACKGROUND\_COLOR
* DK\_BUTTON\_TEXT\_COLOR
* INSTALL\_TEXT\_BACKGROUND\_DRAWABLE (default：140x60 pixel）
* INSTALL\_TEXT\_COLOR
* STATUS\_COLOR
* NAVIGATION\_COLOR
* WALL\_BACKGROUND\_COLOR
* AD\_TITLE\_TEXT\_COLOR
* AD\_DESCRIPTION\_TEXT\_COLOR
* CATEGORY\_TEXT\_COLOR
* CATEGORY\_TEXT\_SIZE
* CATEGORY\_OF\_RECOMMEND 
* CATEGORY\_OF\_POPULAR 
* CATEGORY\_OF\_LIKE 
* AD\_CLICK\_COVER\_LAYER\_TRANS\_BACKGROUND

Example：

|  |
| --- |
| 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);  }  } |

## Preload market data

In order to optimize AD fill, conversion and improve the speed of loading market page ，**strongly suggest developers to preload appwall data. It will have benefit on revenue**：

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

Parameter description：

* context: ApplicationContext

Example：

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

## Start App Market

We can launch App Marktet via Activity or Fragment

1. Launch with Activity：

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

2. Launch with Fragment：

Call below API at first:

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

Parameter description：

* context: Appliction Context
* isFragmentMode: true for fragment mode

Use below method to get fragment instance when you need

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

# Integration with Native Ads

Sample UI:





## 5.1 Configuration

Copy avazu\_common.aar, avazu\_native.aar to your project

1. Integration with 3rd party ad sources, please add the related dependencies in build.gradle file:

compile 'com.google.android.gms:play-services-ads:8.4.0' // AdMob Native Ad

compile 'com.facebook.android:audience-network-sdk:4.23.0' // Facebook Native Ad

1. Initialization and Preload

Build a NativeAd object and pass into the unitId (the Ad Uni ID that was generated after creating the Ad Unit in APX). Set up ad load listening. Pulling the ads to the client side before showing them will enhance the user experience and maximum your revenue.

|  |
| --- |
| INativeAd nativeAd = new NativeAd(Context activityContext, String unitId);  nativeAd.setNativeAdListener(new NativeAdListener() {  @Override  public void onAdLoaded() { // Ad load successfully callback  }  @Override  public void onError(String error) {  }  })  mNativeAd.load(); // load ad |

1. Show ad

Call the interface to show the ad when it is preloaded successfully. The adContainer is the container used to display the native ad.

|  |
| --- |
| if (nativeAd.isLoaded()) {  nativeAd.show(ViewGroup adContainer);  } |

Ad container configuration reference:

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

# 

# 6. Integration with Advanced Native Ads

The Definition of Advanced Native Ads：Similar to Admob Advanced Native Ads, our sdk will provide interface for developers to get the original data of ads. You can take full advantage of this technology to implement highly custom renderings.

## 6.1 Configuration

1. Initialization and Preload

Build a AdvancedNativeAd object and pass into the unitId (the Ad Unit ID that was generated after creating the Ad Unit in APX). Set up ad load listening and the number of ads that you need to load. Note that the number of ads should be set up between one and ten. Pulling the ads to the client side before showing them will enhance the user experience and maximum your revenue.

|  |
| --- |
| mAdvancedNativeAd = new AdvancedNativeAd(this, unitId);  mAdvancedNativeAd.setAdListener(new INativeAdLoadListener() {  @Override  public void onAdListLoaded(List<INativeAd> ads) {  // Note: If you set up to load more than one ad, the number of return ads may be less than you set, these ads come from one same source.  }  @Override  public void onError(String error) { // Error load  }  });  mAdvancedNativeAd.load(1); // the number should in the range [1, 10] |

3. Show ad (Single)

Ads from different ad sources will be displayed by different codes, including APX, AdMob and Facebook. Please refer to source codes of the demo for details.

|  |
| --- |
| private void showAd() {  switch (mNativeAd.getAdType()) {  case Constants.NativeAdType.AD\_SOURCE\_APX:  inflateApxAdView(mNativeAd);  break;  case Constants.NativeAdType.AD\_SOURCE\_ADMOB\_INSTALL:  inflateAdmobInstallAdView(mNativeAd);  break;  case Constants.NativeAdType.AD\_SOURCE\_ADMOB\_CONTENT:  inflateAdmobContentAdView(mNativeAd);  break;  case Constants.NativeAdType.AD\_SOURCE\_FACEBOOK:  inflateFbNativeAdView(mNativeAd);  break;  }  } |

4. Show ad (Multiple)

After getting multiple ads from the same ad source, developers can customize format that are seamless with content. Carousel is universally used in multiple native ads, please refer to the carousel format of native ads in demo for details.

1. Reward Video AD

Reward video AD is sdk mediation for video ads, include platform: Vungle, Unity, Admob, Applovin, Adcolony. Sample UI:



7.1 Integrated with reward video ads

Copy anative\_common.aar, anative\_reward.aar to your project, and add dependency in build.gradle or libs directory, then configure the AndroidMenifest.xml.

Add dependency in build.gradle:

|  |
| --- |
| 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  } |

Add dependency in libs directory(get source files in demo APP):

|  |
| --- |
| 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. initilize and preload

Build a RewardedVideo object and pass into the unitId(the Ad Unit ID that was generated after creating the Ad Unit in APX). Set up ad load listening and pulling the ads to the client side before showing them will enhance the user experience and maximum your revenue.

|  |
| --- |
| rewardedVideoAd = new RewardedVideoAd(this, unitId, new RewardedVideoAdListener() {  @Override  public void onInitSuccess() {  // init succeed callback, should call loadAd() to load AD here  rewardedVideoAd.loadAd();  }  @Override  public void onInitFailed() {  }  @Override  public void onRewardedVideoAdLoaded() {  // AD load succeed callback, should call show() to display AD  }  @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. show reward video AD

Call show() in onRewardedVideoAdLoaded() to display reward video AD.

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