Avazu Native Ad SDK Setup for Android

Date	Version	Description	Developer
2015.07.17	v1.3.1	Add Function	曹诗聪
2015.08.27	v1.3.3	Add new parameter for C2S API	曹诗聪

1.Introduction

1.1 What the SDK can provide

Avazu ADSDK can provide following functions in your mobile Android App:

- Show Single Banner (Recommended size: 320 * 100)
- Show Transparent Banner (Recommended size: 410* 100)
- Show Banner App Wall
- Show Single Line Button APP Wall
- Show Multiple Line Button APP Wall
- Get Ad raw data by C2S Interface(Including Google Play market Ad, DDL Ad and full screen Ad)
- Random jump to a ad's Google Play Landing page through DirectToMarket Interface

For the ad types, the appearance and size of the ad are fully customizable with our function provided in the SDK.

1.2 Compatibility

The minimum runtime OS requirement is Android 2.3, or higher

2. Configuration

2.1 Traffic Source ID

Make sure you are have registered Avazu APX account and have a valid Traffic Source ID for display.

2.2 Adding the SDK to your project

Copy the following file to the “libs” directory of your project.

- adsdk_1.3.1.jar

Import appcompat-v7 and google play service to your project

Note: If you forget to import these files, the app will crash. And do not import a lib twice.

2.3 Configure AndroidManifest.xml

Add the following permissions to “AndroidManifest.xml”:

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

3. Display Webview Ad

AvazuADSDK provide 5 different types of ads, which include single banner, banner app wall, transparent banner, single line button App wall and multiple-line button App wall.

You can display webview Ad by following the instruction:

3.1 SDK Initialization

Call the initialization function at the startup of your application:

```
Adsdk.initialize(Context context,String sourceId);
```

sourceId: a appropriate Traffic Source ID, and you will use it to create the instance every time , so make sure the it is valid, or SDK will fail to load the ads

3.2 Create AdView

1. Instantiate AdViewSettings

You can use below function:

```
AdViewSettings(int width_dip, int height_dip,
int show_type, boolean should_show_loading)
```

Parameter Description:

• width_dip: Ad width in dip

• height_dip: Ad height in dip

• show_type: One of the following ad display types:

Display Format	Variable Name
Single banner	AdViewSettings.TYPE_BANNER_SINGLE
Banner App Wall	AdViewSettings.TYPE_BANNER_WALL
Transparent Banner	AdViewSettings.TYPE_BANNER_TRANSPARENT
Single Line Button App Wall	AdViewSettings.TYPE_RECT_SINGLE
Multiple Line Button App Wall	AdViewSettings.TYPE_RECT_SINGLE

• should_show_loading: The switch to show loading effect or not when the AdView is loading

2. Generating AvazuAdView

You can generate AvazuAdView with the instantiated AdViewSettings object

```
AvazuAdView adView =new AvazuAdView(Context context,
 AdViewController controller);
```

Then you can place the generated AdView to the proper position in your application.

3. Loading AvazuAdView

Using the function when you need to load ad:

```
adView.loadWebviewAd();
```

4. Setting Listener for Loading Result

```
adView.setAdViewStateListener(AdViewStateListener l);
```

Description of Callback function:

public void onLoadAdStart(AdView view): This method will be called when the AdView starts to load

public void onLoadAdFinish(AdView view, int adCount): This method will be called when the AdView finishes loading

public void onLoadAdError(AdView view, String error): This method will be called when any error occurs on loading. The string "error" contains the error message.

3.3 Customized AdView

AvazuADSDK support fully customization for different AD types, to apply the customized settings, you just to need one line code like:

```
AdViewSettings.setNeedIcon(boolean needIcon);
```

1.Setting customized Ad Elements

Function Name	Variable Type	Variable Name	Usage
setNeedIcon	boolean	needIcon	Show app's icon or not in adview
setNeedTitle	boolean	needTitle	Show app's title or not in adview
setNeedCat	boolean	needCat	Show app's category or not in adview
setNeedSize	boolean	needSize	Show app's size or not in adview
setNeedRating	boolean	needRating	Show app's rating or not in adview
setNeedBtn	boolean	needBtn	Show install button or not in adview
setNeedReviewNum	boolean	needReviewNum	Show app's review number or not in adview
setNeedInstalls	boolean	needInstalls	Show app's install number or not in adview

2. Setting customized Ad Color

Function Name	Variable Type	Variable Name	Usage
setBlockBackColor	String	blockBackColor	set the background color of ad block
setAppTitleColor	String	appTitleColor	set the font color for the app's title
setButtonBackColor	String	buttonBackColor	set the background color of install button
setButtonTextColor	String	buttonTextColor	set the font color of install button
setMainBackColor	String	mainBackColor	set the background color of appeal

Note: When setting AD color, All color value must be a NSString started with "#" followed by 6 hexadecimal RGB digitals. For example, #FFFFFF represents for white color while #000000 represents for color black.

3. Setting App count

Function Name	Variable Type	Variable Name	Usage
setAppCount	int	appCount	Set the number of apps in ad

Note: appCount is required for these ad types:
Banner App Wall, Single Line Button App Wall and Multiple Line Button App Wall.

4. Setting Transparency Ratio For Transparent Banner

Function Name	Variable Type	Variable Name	Usage
setAlpha	int	alpha	Set the transparent ratio for ad

Note: transparentBannerAlpha is required only for Transparent Banner, and the value should be set between 0 (completely transparent) and 100 (completely opaque).

Below you can find how to customize a banner ad:

```
AdViewSettings adSettings = new AdViewSettings(400, 110,
AdViewSettings.TYPE_BANNER_TRANSPARENT, false);
adSettings.setNeedIcon(true);
adSettings.setNeedBtn(true);
adSettings.setNeedCat(true);
adSettings.setNeedInstalls(true);
adSettings.setNeedRating(true);
adSettings.setNeedReviewNum(true);
adSettings.setNeedTitle(true);
adSettings.setAppTitleColor("#FFFFFF");
adSettings.setMainBackColor("#f5f5f5");
adSettings.setBlockBackColor("#080807");
adSettings.setAlpha(80);
AvazuAdView adView = new AvazuAdView(this, adSettings);
adView.loadWebviewAd();
```

4. C2S Interface

You can get access to C2S interface by:

```
AdSDK.getAdRawData(Context context, final String sourceId,
String excludePackages, int limitNumber,int creatives, String market,
FetchRawDataListener listener)
```

Parameter Description:

- Source ID: Traffic Source ID
- excludePackages: The campaign_id of ad which you want exclude, it is useful to avoid displaying repeat ads when page changing . You can input "" when no exclude needs. If there are multiple ads needs to be excluded, you are required to divide the campaign by ",".For example: "6184, 3241".
- limitNumber: The maximum ad number you want to get
- market: you can choose the Ad type of the raw data as below, the default value is "google"

Input String	Ad Type
google	The click url can direct to Google Play Market
ddl	The click url can direct to download APK
optin	The click url can direct to a full screen Ad

- creatives: If specified (i.e., creatives=1), images of various sizes will be returned. Default is "0", meaning no image will be returned.

Below you can find the available image size:

Size	Parameter Name
320x50	banner
320x480	phone_fullscreen
480x320	phone_fullscreen_landscape
1024x768	tablet_fullscreen
768x1024	tablet_fullscreen_landscape
300x250	medium
728x90	leaderboard
160x600	skyscraper
1200x627	content_stream_image

• listener： Get the callback of RawData, return the data of Ads, it has three override function:

public void onLoadRawDataStart():This will be called when the RawData starts to load. Please execute this method in the main thread.
public void onLoadRawDataSuccess(List<FetchAdResult.Ad> data):
This will be called when the RawData finishes loading. Please execute this method in the main thread.
public void onLoadRawDataFail(Error mError):This will be called when fetching Rawdata failed. Please execute this method in the main thread.

Below is the data structure of FetchAdResult.Ad.

Key	Description
campaignid	campaignid The campaign ID of the ad
payout	The payout of one app installation for this ad in USD
pkgname	The package name of the app
title	The title of the app
description	The description of the app
icon	The URL of the app icon (a png image with the size of 100x100)
appcategory	The category of the app
apprating	The rating of the app (0-5 stars)
appreviewnum	The review number of the app
appinstalls	The installation number of the app
appsize	The package file size of the app
creatives	The sizes and URLs of ad images
clkurl	The click URL of the ad

Note： You need to check if it is null before attempt to get the ad picture in your desired size

```
if (datadata.tablet_fullscreen != null) {
                    L.d(datadata.tablet_fullscreen);
}
```

5. DirectToMarket

DirectToMarket is a simple API call that takes the user directly to a popular app on Google Play.

You can get access to the API by:

```
public static void directToMarket(Context context, String souceId,
DirectToMarketManager.DirectToMarketListener listener)
```

Parameter Description:

- Source ID: Traffic Source ID
- listener: Listen to the result of Google Play landing page jumping

public void DirectToMarketSuccess(): Jump to Google Play landing page Successfully

public void DirectMarketToFail(): Failed to jump to Google Play landing page