DU Ad Platform SDK for Android Access Guide

DUAd_SDK_CW1.0.6

Baidu Online Network Technology (Beijing) Co., Ltd

No.	DUAd10120150810
Date	2016-08-19
Ver.	1.0.6
Email	support_duad@baidu.com

Contents

1 Introduction

This document describes how to integrate **DU Ad Platform SDK** into Android apps. **DAP, short for DU Ad platform** offers advertising services for helping Android apps to monetize. This version of SDK provides native ad, interstitial ad and advertising wall.

1.1 Target Audience

This document is for Android app developers.

1.2 Prerequisites

DU Ad Platform SDK currently supports Android 2.3 API level 9 (included) plus system versions.

1.3 Document Conventions

The document conventions are listed below:

Element	**Description**	**Example**
Folder name	.zip, aar etc.	DuappsAd_CW_xxxx.aar
System elements	Path, Parameters	${ android-sdk }/tools/proguard/
Reference	Sytle: Italic	*See 3.1*
Code		DuAdNetwork.init(this, keyJson);
CTA Button	Sytle: Bold	**Download， Install Now**
Note	points for attention	*** Note:**

2 Integration Workflow

This section describes the integration workflow of **DU Ad Platform SDK**.

- The integration workflow for **Single Du Native Ad**:

  1. Apply for App_ID and DAP Placement_ID and Facebook Placement ID. *See Section 3.*
  2. Load **DU Ad Platform SDK** package; configure Androidmanifest.xml. *See Section 4.*
  3. Initialize **DU Ad Platform SDK** *See Section 5.*
  4. Access single Du native ad. See *Section 6. Section 7. Section 8.*

- The access workflow of **DU interstitial ad**:

  1. Apply for App_ID and DAP Placement_ID and Facebook Placement ID. *See Section 3.*
  2. Load **DU Ad Platform SDK** work package; configure Androidmanifest.xml. *See Section 4.*
  3. Initialize **DU Ad Platform SDK**. *See Section 5.*
  4. Access Du Interstitial Ad. See *Section 9.*

- The access workflow of **DU advertising wall**:

  1. Apply for App_ID and DAP Placement_ID and Facebook Placement ID. *See Section 3.*
  2. Load **DU Ad Platform SDK** work package; configure Androidmanifest.xml. *See Section 4.*
  3. Initialize **DU Ad Platform SDK**. *See Section 5.*
  4. Access Du advertising wall. *See Section 10.*

3 Obtain Identity

This section describes the three IDs needed during **DU Ad Platform SDK** integration:

APP ID, DAP Placement ID and Facebook Placement ID.

3.1 APP ID

A. Definition
APP ID is a unique identifier of a developer's APP on **Du Ad Platform**. Each app has its own App ID.

B. Obtain method
Visit our official website http://ad.duapps.com and register your app on **Du Ad Platform**, the APP ID will be generated automatically.

C. Code
`app_license`

3.2 DAP Placement ID

A. Definition
DAP Placement ID is a unique identifier of an ad slot on **DAP (Du Ad platform).** Developers can create multiple DAP Placement IDs for one app.

B. Obtain method
Visit our official website http://ad.duapps.com and after registered your app, you can create the placement for your app.

C. Code
`Pid`

3.3 Facebook Placement ID

A. Definition
Facebook Placement ID is the unique identifier of an ad slot on Facebook audience network.

B. Obtain method

Visit Facebook Developers https://developers.facebook.com to apply it.

C. Code
`fbids`

4 Load SDK and Configuration

This section describes how to load the **DU Ad Platform SDK** into your android project, how to configure the *AndroidManifest.xml* file and how to obfuscate code

against project needs.

4.1 Load DU Ad Platform SDK

A. **Download** the DU Ad Platform SDK package.

- Package name: *DuAD_SDK_CWxxxx.zip*

B. **Unzip** the package

After unzipping the package, two folders are available in the subdirectory:

- **DUAd_SDK**

  This folder stores **DU Ad Platform SDK** aar: *DuappsAd_CW_xxxx.aar*

- **DUAd_SDK_DEMO**

  This folder stores a sample program, which integrates **DU Ad Platform SDK**. All interfaces in this document can be found in corresponding usage in this sample program.

C. **Load** DU Ad Platform SDK

- **When using Android Studio:**

1) Copy the **SDK** aar to your Android Project, under the *libs* directory in root directory.

2) Then configure build.gradle:

```
repositories {
    flatDir {
        dirs 'libs'
    }
}
dependencies {
    compile fileTree(include: ['*.jar'], dir: 'libs')
    compile(name:  '  DuappsAd_CW_Online_v1.0.6',  ext:
'aar')
}
```

***Note:** The assigned directory of flatDir is where the aar file is placed.

- **When using Eclipse:**

1) Create a new Android library.

2) Change the **suffix** of *DuappsAd_CW_Online_xxxx.**aar*** *to **zip** and unzip it, then you will get a **classes.jar**, an **AndroidManifest.xml*** and a ***res*** folder.

3) Copy the ***classes.jar*** and a universal-image-loader-1.9.3.jar to the new created Android library, under the ***libs*** directory in root directory.

4) Replace the AndroidManifest.xml in the new created Android library with the ***AndroidManifest.xml*** in unzipped DuappsAd_CW_Online_xxxx.**aar**

5) Replace the res folder in the new created Android library with the **res**

folder in unzipped DuappsAd_CW_Online_xxxx.**aar.**

4.2 Configure AndroidManifest.xml

Update your *Android Manifest*:

A. Add a `user-permission` element to the manifest. Least Privilege of **DU Ad Platform SDK** is shown below:

```
    <uses-permission
android:name="android.permission.INTERNET" />
    <uses-permission
android:name="android.permission.ACCESS_NETWORK_STATE" />
```

B. Add a `meta-data` element to the `application` element, and fill your DAP App ID (See ***3.1***.) as the value of "app_license".

```
    <application
        android:name="Your_PackageName.MobulaApplication"
        android:label="@string/app_name"
        . . . >
        <meta-data
            android:name="app_license"
            android:value="@string/DAP_APP_ID" />
    </application>
```

C. Declare the *com.duapps.ad.stats.DuAdCacheProvider* in the manifest. Replace the below *packagename* with your app's full package name. Please make sure the package name at here is exactly the same as the package name you filled on DAP when registering you app. Otherwise, it will fail to get ad from DAP.

```
<provider
   android:name="com.duapps.ad.stats.DuAdCacheProvider"
   android:authorities="packagename.DuAdCacheProvider"
   android:exported="false">
 </provider>
```

D. Register the BroadcastReceiver for receiving app install event.

Solution 1: Statically register the PACKAGE_ADDED Receiver in AndroidManifest.xml.

```
    <receiver
android:name="com.duapps.ad.base.PackageAddReceiver" >
    <intent-filter>
```

```
      <action
android:name="android.intent.action.PACKAGE_ADDED" />
      <data android:scheme="package" />
     </intent-filter>
  </receiver>
```

Solution 2: Dynamically register the BroadcastReceiver for PACKAGE_ADDED.

If developers had registered their own BroadcastReceiver for PACKAGE_ADDED in AndroidManifest.xml, they should use the below interface to pass the broadcast of APP install event to Du Ad platform SDK. **This interface can be used repeatedly.**

- **Interface Instruction:**
  DuAdNetwork.onPackageAddReceived(Context context, Intent intent);

Parameters	Description
Context context	Application context
Intent intent	Broadcast intent

- **Code Sample:**

```
public class MyBroadcast extends BroadcastReceiver{

@Override
public void onReceive(Context context, Intent intent) {
       DuAdNetwork.onPackageAddReceived(context, intent);
    }
}
```

* Note： The above “MyBroadcast” is the developer’s own BroadcastReceiver for PACKAGE_ADDED.

4.3 Obfuscate Code

Please follow the below rules to obfuscate code. Otherwise, there might be exceptions at run time.

A: Exclude classes of **DU Ad Platform SDK** when obfuscating;

B: Below classes can add to `proguard` configuration:

```
-keep class com.dianxinos.DXStatService.stat.TokenManager {
public static java.lang.String getToken(android.content.Context);
}
-keep public class * extends android.content.BroadcastReceiver
```

```
-keep public class * extends android.app.Activity
-keep public class * extends android.app.Application
-keep public class * extends android.content.ContentProvider

-keepnames @com.google.android.gms.common.annotation.KeepName class *
-keepclassmembernames class * {
       @com.google.android.gms.common.annotation.KeepName *;}
-keep class com.google.android.gms.common.GooglePlayServicesUtil {
     public <methods>;}

-keep class com.google.android.gms.ads.identifier.AdvertisingIdClient {
     public <methods>;}
-keep class
com.google.android.gms.ads.identifier.AdvertisingIdClient$Info {
     public <methods>;}
```

* **Note**: For more about obfuscation methods, please refer to the official Android obfuscation document at: *${ android-sdk }/tools/proguard/*

5 Initialization

This section describes how to initialize DAP SDK. You need to initialize DAP SDK before you can use it.

- **Method:**
  Add a call to DuAdNetwork.init() from onCreate in your Application class.

```
import com.duapps.ad.base.DuAdNetwork;
public class MobulaApplication extends Application {
    @Override
    public void onCreate() {
        super.onCreate();
       // Initialize the DAP SDK before executing any other
operations
        DuAdNetwork.init(this,
getConfigJSON(getApplicationContext()));
    }
```

- **Operation:**
  Go to *init* and use JSON format to write **String** data with mappings for the **DAP Placement ID (pid)** and **Facebook Placement ID (fbids)**

```
{
```

```
    "native": [
       {
       "pid": "10032",
       "fbids": [
          "xxxxxxxxx_xxxxxxxxx",
          "xxxxxxxxx_xxxxxxxxx" ]
       },{
       "pid": "xxxxx",
       "fbids": ["xxxxxxxxx_xxxxxxxxx"]
       }
    ],
    "offerwall": [
       {
          "pid": "xxxxx",
          "fbids":"xxxxxxxxx_xxxxxxxxx"
       }
     ]
}
```

***Note:** If some of the DAP placements (pid) don't need ads from Facebook, the `"fbids"` part for that "pid" (DAP placement) could be removed.

- **Interface Instruction:**

  public static void *init*(Context context， String pidsJson);

Parameters	Description
Context context	Activity Context
String pidsJson	The relationship between DAP Placement ID and Facebook Placement ID.

- **Code Sample:**

```
/*** read the json.txt from assets */

private String getConfigJSON(Context context) {
   BufferedInputStream bis = null;
```

```
    ByteArrayOutputStream bos = new ByteArrayOutputStream();
    try {
        bis                          =                          new
BufferedInputStream(context.getAssets().open("json.txt"));
        byte[] buffer = new byte[4096];
        int readLen = -1;
        while ((readLen = bis.read(buffer)) > 0) {
            bos.write(buffer, 0, readLen);
        }
    } catch (IOException e) {
        Log.e("", "IOException :" + e.getMessage());
    } finally {
        closeQuietly(bis);
    }

    return bos.toString();
}

private void closeQuietly(Closeable closeable) {
    if (closeable == null) {
        return;
    }
    try {
        closeable.close();
    } catch (IOException e) {
        // empty
    }
}
```

6 Request single native ad

6.1 Construction

6.1.1 Constructor of Du Native Ad

- **Interface Instructions:**

public DuNativeAd (Context context, **int** pid)
public DuNativeAd (Context context, **int** pid, **int** cacheSize)

Parameters	Description
Context context	Activity Context
int pid	DAP placement ID, see 3.2
int cacheSize	ad cache size. Recommend to set cachesize to 1 or don't set cachesize.

- **Code Sample:**

```
private DuNativeAd nativeAd;

nativeAd = new DuNativeAd(this, "your_DAP_placement_ID",
"Your_cache_size");
```

6.1.2 Correlate Facebook Placement ID dynamically

- **Interface Instructions:**

public void setFbids (List<String> fbids);

Parameters	Description
List<String> fbids	Facebook Placement ID, see 3.3

*__Note__: For using this interface, a default correlated fbids need to be configured in Json (see *chapter 5*). Then the new parameter (List<String> fbids) will cover the corresponding fbids configured in Json.

6.2 Pre-cache the native ad

- **Interface Instruction:**

public void fill();

Use the fill() to pre-cache ad in advance for faster loading the ad when using load(). Developers can select the timing for pre-cache native ad.
Suggestion: Use the fill() at the page before the ad showing page.
***Note:** Ad data will be cached in client device's memory. Since SDK only caches the image's URL address not the image data, the cache size is small.

6.3 Retrieve native ad

Please register a callback interface for receiving the native ad data. The ad retrieving process is asynchronous, so it will not block developers' threads.

6.3.1 Set listener for native ad

- **Interface Instruction:**

public void setMobulaAdListener (**DuAdListener** adListener);

Parameters	Description

DuAdListener	Callback function returns: ad error, ad data, and ad click event.
	public interface DuAdListener {
	public void onError(DuNativeAd ad, AdError error);
	public void onAdLoaded(DuNativeAd ad);
	public void onClick(DuNativeAd ad);
	}

6.3.2 Retrieve ad

- **Interface Instruction:**

public void load();

- **Code Sample:**

```
if (nativeAd != null) {
          nativeAd.setMobulaAdListener (mListener);
          nativeAd.load();
 }
```

```
DuAdListener mListener = new DuAdListener () {
@Override
public void onError (DuNativeAd ad, AdError error) {
       }
@Override
public void onClick (DuNativeAd ad) {
       }
@Override
public void onAdLoaded (final DuNativeAd ad) {
       }
};
```

After called load(), three types of results could be returned:

a) **Retrieve ad successfully.** Modify the *onAdLoaded* function above to retrieve the ad properties. *See 7.2*.

b) **Get an error.** Get specific error information in *onError* function above. Error code and description are shown in Table 2.

Table2 Error Code

Constants	Error Code	Description
NETWORK_ERROR_CODE	1000	Client network error
NO_FILL_ERROR_CODE	1001	No Ad data retrieved
LOAD_TOO_FREQUENTLY_ERROR_CODE	1002	Too many interface requests
SERVER_ERROR_CODE	2000	Server error
INTERNAL_ERROR_CODE	2001	Network error
TIME_OUT_CODE	3000	Retrieve Ad data timed out
UNKNOW_ERROR_CODE	3001	Unknown error

c) **Retrieve a ad click event.** Get informed when an ad is clicked in *onClick* function.

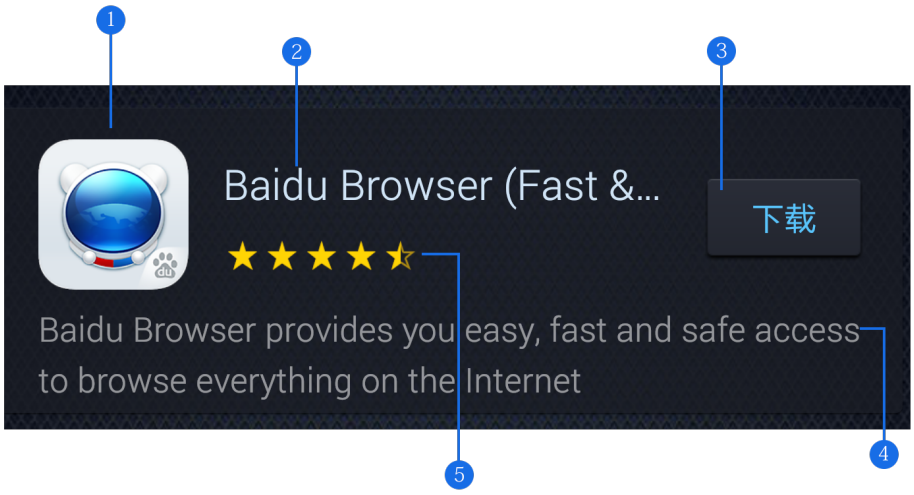
7. Native ad properties

When using the Native Ad, instead of receiving an ad ready to be displayed, you will receive a group of ad properties such as a title, an image, a call to action, and you will have to use them to construct a custom view where the ad is shown. This section describes the ad properties and how to get them.

7.1 Introduction of ad properties

Ad properties include: Icon, title, Call to action (CTA) button, short description, rating, promotion image, etc. *See Figure 1*.

Figure 1 Ad properties



① Icon ② Title ③ CTA button ④ Short description ⑤ Rating

7.2 Get the ad properties

The interfaces for retrieving the ad properties as shown below:

- Get **Icon**

Interface Instruction:
public String getIconUrl();

Return Value	Description
String iconUrl	The URL address of icon

- Get **Title**

Please reserve at least 20 charactors' space to display the title.
An ellipsis (…) can be used to indicate truncated text.
Please note the ad title must be included in your native ad design.

Interface Instruction:
public String getTitle();

Return Value	Description
String title	The title of ad

- **Get Call to Action (CTA) button**

Advertisers can specify the text of CTA button, e.g. **Install Now**. Please do not shorten or change the text.
For CTA button with promotion image, the **max** character length is **25.** For CTA button without image, the text is usually defined as **Download**.
Please note the CTA button must be included in your native ad design.

Interface Instruction:
public String getCallToAction();

Return Value	Description
String callToAction	The text of ad's CTA button

- Get **Short description**

Please reserve at least 72 charactors' space to display the short description.
If the space is not big enough, it is recommended to use scrolling text effects, or do not display the short description.

Interface Instruction:
public String getShortDesc();

Return Value	Description
String shortDesc	The short description of ad

- Get **Rating**

Interface Instruction:

public float getRatings();

Return Value	Description
float ratings	The ad's rating on Google Play.

- Get **Promotion Image**

A promotion image can increase user's desire to click the ad.

The image size is usually: 1200x627 pixels. You can zoom and cut part of the image, but do not distort or change it. **Please note** that not all ads have promotion images.

Interface Instruction:

public String getImageUrl();

Return Value	Description
String imageUrl	The URL address of ad's promotion image. When the image is not included in current ad, the returned value is NULL.

- **DuAdChoicesView**

This view is the AdChoices corner mark from by Facebook Native Ad. It's the mandatory element for Facebook native Ad. **Please Note that** the native ad which is not from Facebook doesn't have this.

Constructor: DuAdChoicesView choicesView = new DuAdChoicesView(....);

Usage: Create a View for AdChoices separately. It is different from Ad corner mark.

Interface Instruction:

public void addView(DuAdChoicesView choicesView);

Return Value	Description
DuAdChoicesView choicesView	DuAdChoicesView object

8. Register the native ad's View

The SDK will log the impression and handle the click automatically. Please note that you must register the ad's view with the DuNativeAd instance to enable that.

- **Interface Instruction:**

public void registerViewForInteraction(View view)
public void registerViewForInteraction(View view, List<View> views)

Return Value	Description
View view	Clickable View in Ad contents
List<View> views	More detailed sub-View

*** Note:** Don't recommend using this interface in multi-thread.

9. Request interstitial ad

Figure 2 A sample of half screen interstitial ad (vertical and horizontal)

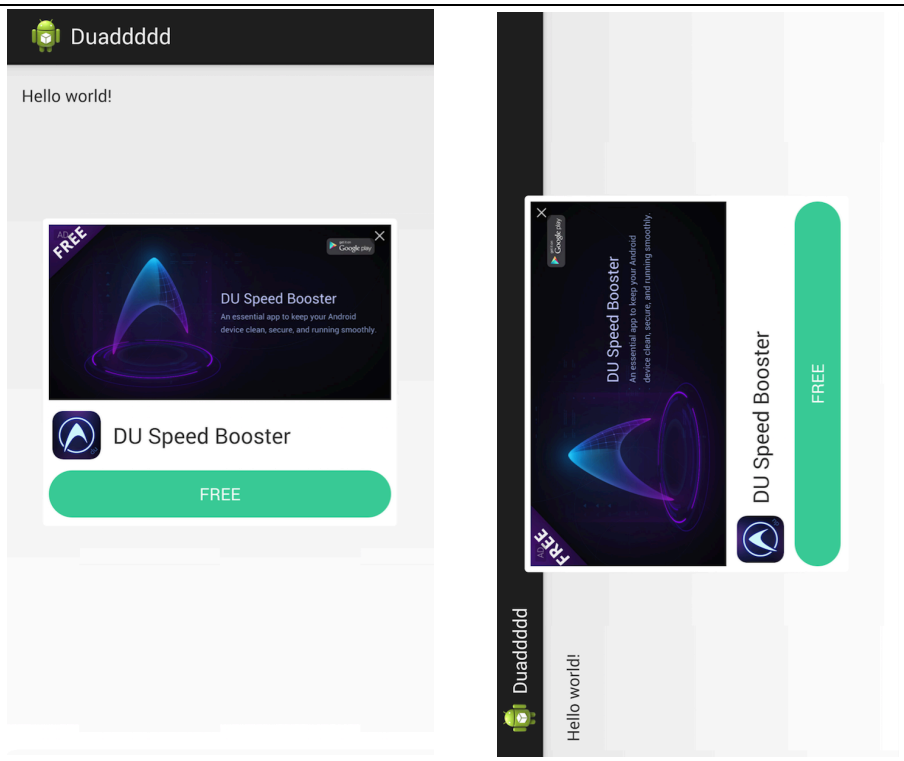
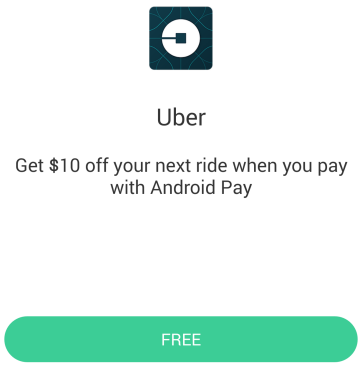


Figure 3　A sample of full screen interstitial ad



9.1　Constructor of interstitial ad

- **Interface Instruction:**

public InterstitialAd (Context context, int pid, int type)

Parameters	Description
Context context	ACTIVITY CONTEXT
int pid	DAP placement ID
int type	InterstitialAd.Type.SCREEN: is full screen Ad InterstitialAd.Type.NORMAL: is half screen Ad The default value is half screen

*** Note:** The initialization JSON for InterstitalAd need to be added into "Native" part, see *Section 5*.

```
{
    "native": [
        {
            "pid": "xxxxx",
            "fbids": ["xxxxxxxxxx_xxxxxxxxxx"]
        }
    ]
}
```

9.2 Set listener for interstitial ad

- **Interface Instruction:**

public void setInterstitialListener (AbsInterstitialListener mAbdl);

Parameters	Description
AbsInterstitialListener mAbdl	Abstract class of listener

Callback function returns:

1) Retrieve ad failed `onAdFail(int errcode);`
   ***Note:** see *Table 2* in *6.3.2* for error code
2) Retrieve ad successfully `onAdReceive();`
3) ad destroyed event `onAdDismissed();`
4) ad impression event `onAdPresent();`
5) ad click event `onAdClicked();`

9.3 Pre-cache interstitial ad

- **Interface Instruction:**

  public void fill();

Use the fill() to pre-cache ad in advance for faster loading the ad when using load().

Suggestion: Developers can use the fill() at the page before the ad showing page or use the fill() at least 4-5 seconds before using load().

***Note:** Ad data will be cached in client device's memory. Since SDK only caches the image's URL address not the image data, the cache size is small.

9.4 Retrieve interstitial ad

- **Interface Instruction:**

```
public void load ();
```

***Note:** Please Set the listener of Interstital Ads before loading the Ads.

9.5 Show interstitial ad

- **Interface Instruction:**

```
public void show ();
```

***Note:** Please use this interface in `onAdReceive()` (see 9.2).

9.6 Close interstitial ad

- **Interface Instruction:**

```
public void close ();
```

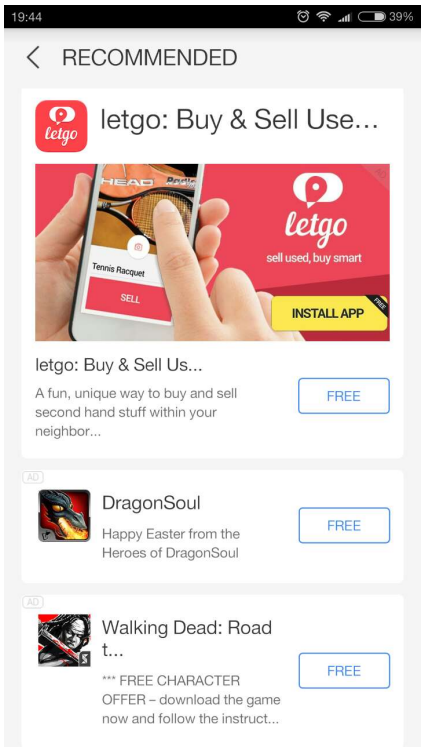
9.7 Destroy interstitial ad

- **Interface Instruction:**

```
public void destroy ();
```

10. Request advertising wall

Figure 4 A sample of advertising wall



Advertising wall is an encapsulated list Advertising (This is an Activity). For displaying the advertising wall, developers need to pass in the DAP placement ID when page jumping. Please take the below example for reference.

```
intent = new Intent
(ToolboxSampleMainActivity.this, OfferWallAct.class);

Bundle b = new Bundle();
b.putInt("pid", Your_DAP_placement_ID );

intent.putExtras(b);
startActivity(intent);
```

***Note:** `OfferWallAct.class` is the encapsulated activity class of advertising wall. Please fill in your DAP placement ID (see *3.2*), and please make sure the DAP placement ID had been registered in "`offerwall`" part during initialization (see *section 5*).

```
"offerwall": [
      {
        "pid": "xxxxx",
        "fbids":"xxxxxxxxx_xxxxxxxxx"
      }
]
```