

DU Ad Platform SDK for Android Access Guide

DUAd_SDK_CW1.0.9.7

Baidu Online Network Technology (Beijing) Co., Ltd

No.	DUAd10120150810
Date	2017-07-06
Ver.	1.0.9.7
Email	support_duad@baidu.com

Contents

1	Introduction	1
1.1	Prerequisites	1
2	Integration Workflow	1
3	Obtain Identity	2
3.1	APP ID.....	2
3.2	DAP Placement ID	2
3.3	Facebook Placement ID	3
4	Load SDK and Configuration	3
4.1	Load DU Ad Platform SDK	3
4.2	Configure AndroidManifest.xml	4
4.3	Obfuscate Code.....	5
5	Initialization	6
6	Request single native ad	9
6.1	Construction	9
6.2	Pre-cache the native ad.....	10
6.3	Retrieve native ad	10
6.4	Get cached native ad.....	11
7.	Native ad properties	12
7.1	Introduction of ad properties	12
7.2	Get the ad properties	12
8.	Register the native ad's View	14
9.	Request native ad list	15
9.1	Construct Manager Class of Native Ad List	15
9.2	Pre-cache native ad list	16
9.3	Retrieve native ad list	16
9.4	Register a listener for Manager Class of Native Ad List (DuNativeAdsManager).....	16
9.5	Register a listener for each single ad in ad List	17
9.6	Destroy the object and listener interface of Native Ad list.....	18
10.	Request interstitial ad	19
10.1	Constructor of interstitial ad	20
10.2	Set listener for interstitial ad	20
10.3	Pre-load interstitial ad	20
10.4	Retrieve interstitial ad	21
10.5	Show interstitial ad.....	21
10.6	Close interstitial ad	21
10.7	Destroy interstitial ad	21
11.	Request offer wall	22
11.1	Mandatory parameter	22
11.2	Optional parameter	22

12.	Request Banner ad	24
12.1	Constructor of Banner ad	24
12.2	Set background color of Banner ad	25
12.3	Set close button of Banner ad	25
12.4	Add Banner ad to custom layout	25
12.5	Load Banner ad	26
12.6	Destroy Banner ad	26
13.	Request Trigger ad	26
13.1	Trigger Ad configuration	27
13.2	Trigger configuration	27

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, offer wall, banner ad and trigger ad.

1.1 Prerequisites

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

2 Integration Workflow

- 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 integration workflow for **Du Native Ad List**:

Du Native Ad List is for showing multiple ads in one page at the same time. (**Please note that** Du Native Ad List has relatively poor monetization efficiency compared with single Du Native Ad. Please use this according to your situation.)

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 Du Native ad list. See [Section 9](#).

- 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** package; configure Androidmanifest.xml. See [Section 4](#).
3. Initialize **DU Ad Platform SDK**. See [Section 5](#).
4. Access Du Interstitial Ad. See [Section 10](#).

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

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 Du offer wall. See [Section 11](#).

- The access workflow of **DU Banner 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 DU Banner ad. See [Section 12](#).

- The access workflow of **DU Trigger 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 DU Trigger ad. See [Section 13](#).

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_CW1.0.9.7.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-xxx-release.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-xxx-release', ext: 'aar')
}
```

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

- **When using Eclipse:**

- 1) Create a new Eclipse Android library.
- 2) Change the **suffix** of *DuappsAd-CW-xxx-release.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** 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-xxx-release.aar*
- 5) Replace the res folder in the new created Android library with the **res** folder in unzipped *DuappsAd-CW-xxx-release.aar*.

4.2 Configure AndroidManifest.xml

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="com.mobula.sample.MobulaApplication"
  android:label="@string/app_name"
  . . . >
  <meta-data
    android:name="app_license"
    android:value="@string/Your_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="Your_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>;}
-keep class com.duapps.ad.banner.BannerListener { *; }
```

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

C: If accessing Facebook ads, please add the below class to proguard configuration.

```
-keep class com.facebook.ads.NativeAd
```

D: If accessing Admob ads, please add the below class to proguard configuration.

```
-keep class com.google.android.gms.ads.formats.NativeContentAd
```

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.

Also, you can add a `DuAdNetwork.setLaunchChannel()` from `onCreate` in your Application class, to distinguish your data by your app distribution channel. It's optional.

```
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()));
        // Distinguish your data by your app distribution channel. It's
        optional.
        DuAdNetwork.setLaunchChannel("YOUR_APP_CHANNEL");
    }
}
```

- **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": "YOUR_DAP_PLACEMENT_ID",
      "fbids": ["YOUR_FACEBOOK_PLACEMENT_ID"]
    }, {
      "pid": "YOUR_DAP_PLACEMENT_ID",
      "fbids": ["YOUR_FACEBOOK_PLACEMENT_ID"]
    }
  ],
  "list": [{
    "pid": "YOUR_DAP_PLACEMENT_ID",
    "fbids": "YOUR_FACEBOOK_PLACEMENT_ID"
  }, {
    "pid": "YOUR_DAP_PLACEMENT_ID",
    "fbids": "YOUR_FACEBOOK_PLACEMENT_ID"
  }],
  "offerwall": [
    {
      "pid": "YOUR_DAP_PLACEMENT_ID",
      "fbids": "YOUR_FACEBOOK_PLACEMENT_ID"
    }
  ]
}
```

***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.

***Note:** For the pid under "offerwall", please make sure it's Ad Format is selected as 『Offerwall』 during creation. Otherwise, it will fail to get ad.

***Note:** If FBID is bound, please make sure the facebook sdk version is not lower than fb 4.20.0.

- **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.

- **Interface Instruction:**

public static void *setLaunchChannel* (String channelName);

Parameters	Description
String channelName	To distinguish your data by you app distribution channel. It's optional.

- **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 cache size to 1 or 2 or don't set cachesize. The default value will be 1.

- **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().

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. <pre>public interface DuAdListener { public void onError(DuNativeAd ad, AdError error); public void onAdLoaded(DuNativeAd ad); public void onClick(DuNativeAd ad); }</pre>

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
<i>NETWORK_ERROR_CODE</i>	1000	Client network error
<i>NO_FILL_ERROR_CODE</i>	1001	No Ad data retrieved
<i>LOAD_TOO_FREQUENTLY_ERROR_CODE</i>	1002	Too many interface requests
<i>SERVER_ERROR_CODE</i>	2000	Server error
<i>INTERNAL_ERROR_CODE</i>	2001	Network error
<i>TIME_OUT_CODE</i>	3000	Retrieve Ad data timed out
<i>UNKNOW_ERROR_CODE</i>	3001	Unknown error

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

6.4 Get cached native ad

6.4.1 Check if there is cached ad

- **Interface Instruction:**

```
public boolean isHasCached();
```

6.4.2 Get cached native ad

- **Interface Instruction:**

```
public DuNativeAd getCacheAd();
```

This interface is for getting the cached ad for the current ad object. It could be traversed until the number of cached ads goes to 0. Please make sure the cache pool is not null before showing ad.

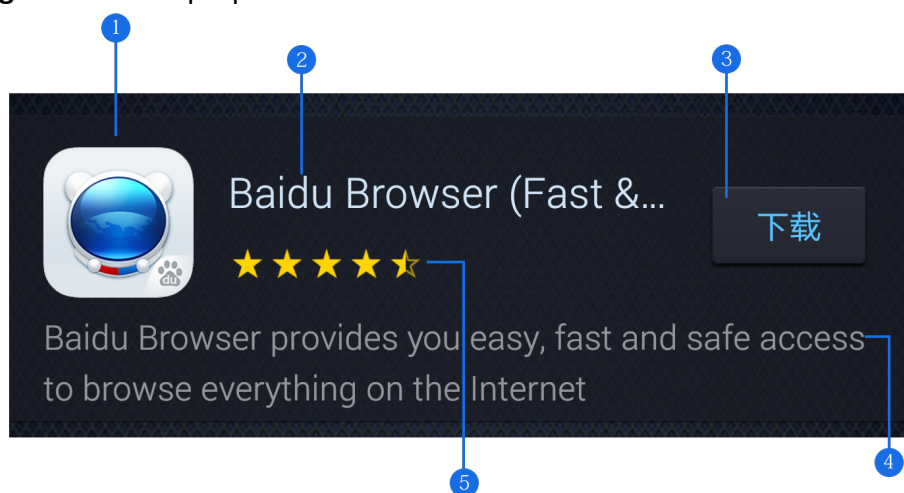
7. Native ad properties

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 characters' 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 characters' 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: 1200x628 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);

Parameter	Description
DuAdChoicesView choicesView	DuAdChoicesView object

- **Code Sample:**

```
if (adType == DuNativeAd.CHANNEL_TYPE_FB)
{
    DuAdChoicesView choicesView = new
    DuAdChoicesView(FullSDKAct.this, ad, true);
    adChoicesContainer.addView(choicesView);
    bigImgView.setVisibility(View.GONE);
    mMediaView.setVisibility(View.VISIBLE);
    mMediaView.setAutoPlay(true);
    mMediaView.setNativeAd(ad.getRealSource());
}
```

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)

Parameter	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 native ad list

Du Native Ad List is for showing multiple ads in one page at the same time. (Please note that Du Native Ad List has relatively poor monetization efficiency compared with single Du Native Ad. Please use this according to your situation.)

The whole workflow of getting the Ad is done in AsyncTask. Please use this function in the main thread.

9.1 Construct Manager Class of Native Ad List

- **Interface Instruction:**

public DuNativeAdsManager(Context context, int pid, int cacheSize);

Parameter	Description
Context context	ACTIVITY CONTEXT
int pid	DAP Placement ID
int cacheSize	Native ad list cache size

- **Code Sample:**

```
DuNativeAdsManager adsManager = new DuNativeAdsManager
(getApplicationContext(),PID,cacheSize);
```

Note: The PID used for interstitial Ad need to be added into "list" part JSON during initialization, see [Section 5](#).

```
{
  "list": [
    {
      "pid": "YOUR_PLACEMENT_ID",
      "fbids": "YOUR_FACEBOOK_PLACEMENT_ID"
    }
  ]
}
```

```
    ]
}
```

9.2 Pre-cache native ad list

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.

- **Interface Instruction:**

```
public void fill();
```

- **Code Sample:**

```
adsManager.fill();
```

9.3 Retrieve native ad list

Use `load()` to retrieve the native ad list. After called `load()`, three types of results could be returned, see [9.4](#), [9.5](#).

- **Interface Instruction:**

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

- **Code Sample:**

```
Private void showNativeAdList(){
    . . .
    //see 9.4 for listener
    adsManager.setListener(listener);
    adsManager.load();
}
```

9.4 Register a listener for Manager Class of Native Ad List (DuNativeAdsManager)

- **Interface Instruction:**

```
public void setListener (AdListArrivalListener listener);
```

Parameter	Description
-----------	-------------

AdListArrivalListener listenerListener for Native Ad list, see [9.3](#)

- **Code Sample:**

```
adsManager.setListener(listener);
```

```
AdListArrivalListener listener = new AdListArrivalListener()
{
    @Override
    public void onAdLoaded(List arg0) {
        . . .
    }
    @Override
    public void onAdError(AdError arg0) {
    }
};
```

***Note:** There is no onAdclick() callback for Manager Class of Native Ad List.

9.5 Register a listener for each single ad in ad List

- **Interface Instruction:**

```
public void setMobulaAdListener (DuAdDataCallBack callback);
```

Parameters	Description
DuAdDataCallBack	Callback function returns: click event. <pre>public interface DuAdDataCallBack { public void onAdClick (); }</pre>

- **Code Sample:**

```
AdListArrivalListener listener = new AdListArrivalListener()
{
    @Override
    public void onAdLoaded(List arg0)
    {
        loadBtn.setEnabled(true);
        rootContainer.removeAllViews();
        Log.d(TAG, "-----start to fill view-----");
        for (int i = 0; i < arg0.size(); i++)
        {
```

```

        //get a single NativeAd object
        NativeAd ad = (NativeAd) arg0.get(i);
        rootContainer.addView(createItem(ad));

        //Set Ad listener for a single Native Ad object
        ad.setMobulaAdListener(callback);
    }
    Log.d(TAG, "-----end to fill view-----");
}
. . .
};

```

```

DuAdDataCallBack callback = new DuAdDataCallBack() {
    @Override
    public void onAdLoaded(NativeAd data) {
    }
    @Override
    public void onAdError(AdError error) {
    }
    @Override
    public void onAdClick() {
        Log.d(TAG, "ad is click");
    }
};

```

*Note: There is no onAdLoaded() callback and onAdError() callback for single ad in ad list.

9.6 Destroy the object and listener interface of Native Ad list

When exiting the native ad list showing page, the object(DuNativeAdsManager) and listener(AdListArrivalListener) must be destroyed.

- **Code Sample:**

```

protected void onDestroy()
{
    super.onDestroy();
    adsManager.setListener(null);
    adsManager.destroy();
}

```

10. Request interstitial ad

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

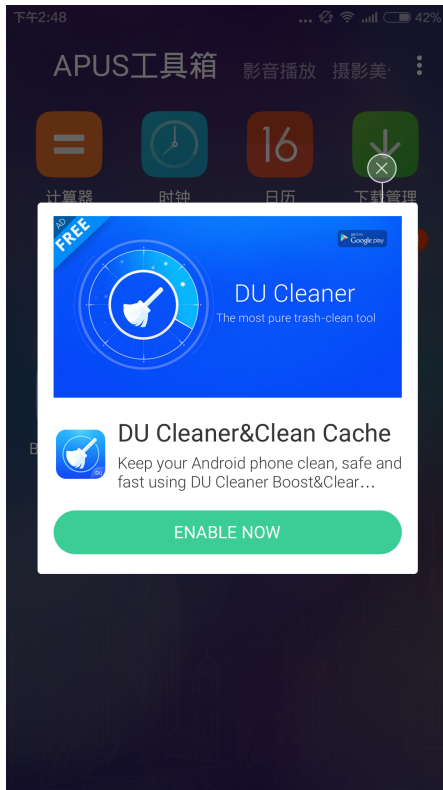
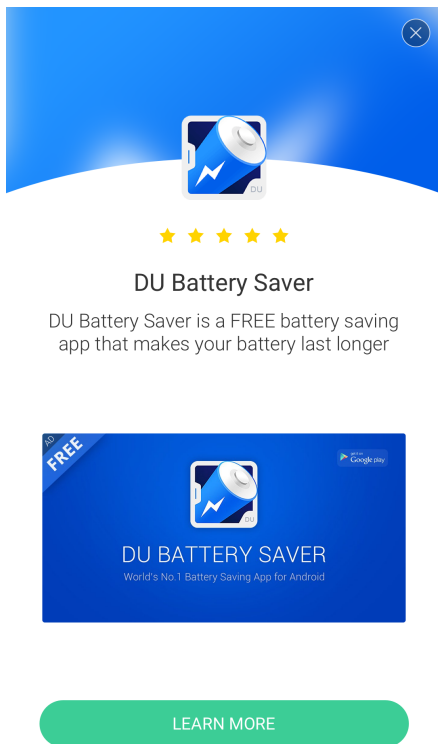


Figure 3 A sample of full screen interstitial ad



10.1 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 PID used for interstitial Ad need to be added into “native” part JSON during initialization, see [Section 5](#).

```
{
  "native": [
    {
      "pid": "YOUR_PLACEMENT_ID",
      "fbids": ["YOUR_FACEBOOK_PLACEMENT_ID"]
    }
  ]
}
```

10.2 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();

10.3 Pre-load interstitial 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.

10.4 Retrieve interstitial ad

- **Interface Instruction:**
`public void load ();`

*Note: Please set the listener of interstitial ad before calling `load()`.

10.5 Show interstitial ad

- **Interface Instruction:**
`public void show ();`

*Note: Please use this interface in `onAdReceive()` (see [10.2](#)).

10.6 Close interstitial ad

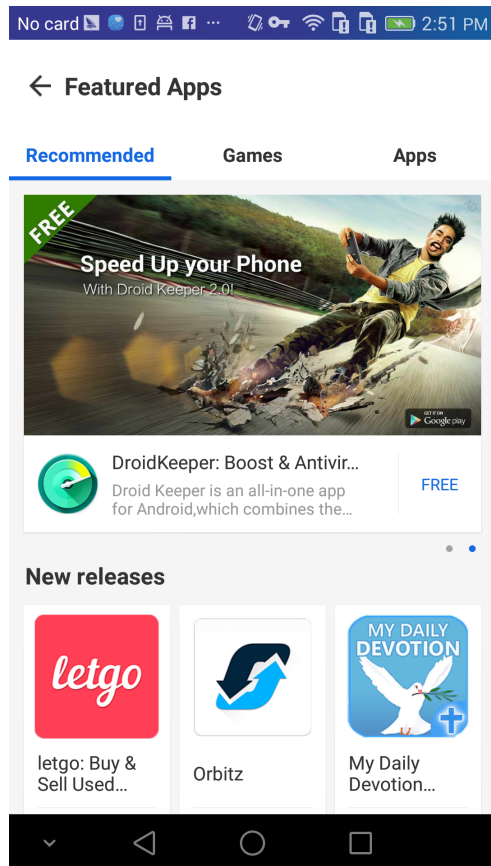
- **Interface Instruction:**
`public void close ();`

10.7 Destroy interstitial ad

- **Interface Instruction:**
`public void destroy ();`

11. Request offer wall

Figure 4 A sample of offer wall



11.1 Mandatory parameter

`OfferWallAct.class` is the encapsulated activity class of offer wall. Please fill in your DAP placement ID (see [3.2](#)) before using it.

Please pass your DAP placement ID for offerwall ad to `OfferWallAct.KEY_PID`

please make sure the Ad Format of this DAP placement ID is selected as 『Offerwall』 during creation on du ad platform. Otherwise, it will fail to get ad.

Create placement

* Placement name :

* Ad Format : Native Interstitial Offerwall

11.2 Optional parameter

- Customize your offerwall title text

key: OfferWallAct.KEY_TITLE_ID

value: The id of string defined in xml file.

- Customize the background color of title and tab

key: OfferWallAct.KEY_TAB_BACKGROUND_COLOR

value: color code, e.g. #ffffff.

- Customize the color of selected tab text

key: OfferWallAct.KEY_TAB_INDICATOR_COLOR

value: color code, e.g. #ffffff.

- Customize the color of title text and the color of unselected tab text

key: OfferWallAct.KEY_TAB_TEXT_COLOR

value: color code, e.g. #ffffff.

- **Code Sample:**

```
Intent intent = new Intent(ToolboxSampleMainActivity.this, OfferWallAct.class);
Bundle b = new Bundle();
b.putInt(OfferWallAct.KEY_PID, Your_DAP_placement_ID);
b.putInt(OfferWallAct.KEY_TITLE_ID, R.string.app_name); // Optional
b.putString(OfferWallAct.KEY_TAB_BACKGROUND_COLOR, "#EDED"); // Optional
b.putString(OfferWallAct.KEY_TAB_INDICATOR_COLOR, "#1C86EE"); // Optional
b.putString(OfferWallAct.KEY_TAB_TEXT_COLOR, "#000000"); // Optional
intent.putExtras(b);
startActivity(intent);
```

*Note: The PID used for offerwall Ad need to be added into “offerwall” part JSON during initialization, see [Section 5](#).

```
"offerwall": [
  {
    "pid": "YOUR_PLACEMENT_ID",
    "fbids": "YOUR_FACEBOOK_PLACEMENT_ID"
  }
]
```

12. Request Banner ad

Figure 5 A sample of banner ad (blue background)

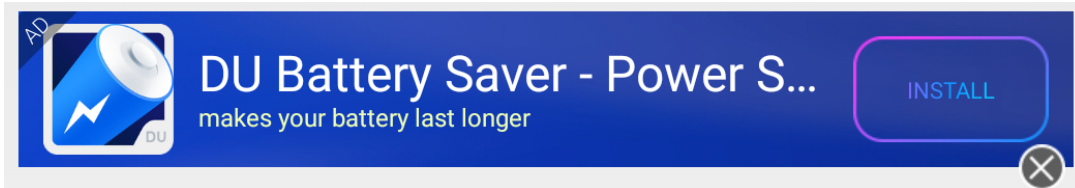
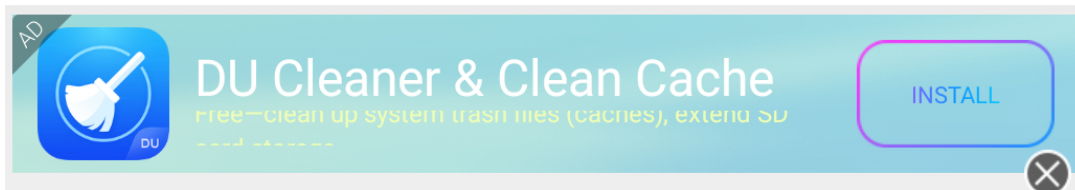


Figure 6 A sample of banner ad (green background)



12.1 Constructor of Banner ad

- **Interface Instruction:**

public BannerAdView (Context context, int pid, int cachesize, BannerListener listener)

Parameters	Description
Context context	ACTIVITY CONTEXT
int pid	DAP placement ID
int cachesize	The number of cached banner ads
BannerListener listener	Banner Ad listener (Only callback for the first time Load result for each BannerAdView)

* **Note:** The PID used for banner Ad need to be added into “native” part JSON during initialization, see [Section 5](#).

Add `-keep class com.duapps.ad.banner.BannerListener { *; }` to obfuscate code for BannerListener, see [4.3](#).

```
{
  "native": [
    {
      "pid": "YOUR_PLACEMENT_ID",
      "fbids": ["YOUR_FACEBOOK_PLACEMENT_ID"]
    }
  ]
}
```

- **Code sample:**

```
mBannerAdView = new BannerAdView(this, 10032, 5, new
BannerListener() {
    @Override
    public void onAdLoaded() {
        Log.d(TAG, "onAdLoaded");
    }

    @Override
    public void onError(String msg) {
        Log.d(TAG, "onError:" + msg);
    }
});
```

12.2 Set background color of Banner ad

- **Interface Instruction:**

public void setBgStyle(BannerStyle)

Parameters	Description
BannerStyle mBannerStyle	Set background color: BannerStyle.STYLE_BLUE: for blue background; BannerStyle.STYLE_GREEN: for green background;

12.3 Set close button of Banner ad

- **Interface Instruction:**

public void setCloseStyle(BannerCloseStyle.*STYLE_BOTTOM*);

Parameters	Description
BannerCloseStyle mBannerCloseStyle e	Set the position of close button: BannerCloseStyle.STYLE_BOTTOM: bottom right corner; BannerCloseStyle.STYLE_TOP: top right corner;

12.4 Add Banner ad to custom layout

- **Interface Instruction:**

Layout.addView(mBannerAdView);

Parameters	Description
BannerAdView mBannerAdView	Created Banner Note: layout is your app's custom layout

12.5 Load Banner ad

- **Interface Instruction:**

```
mBannerAdView.load();
```

12.6 Destroy Banner ad

- **Interface Instruction:**

```
mBannerAdView.onDestroy();
```

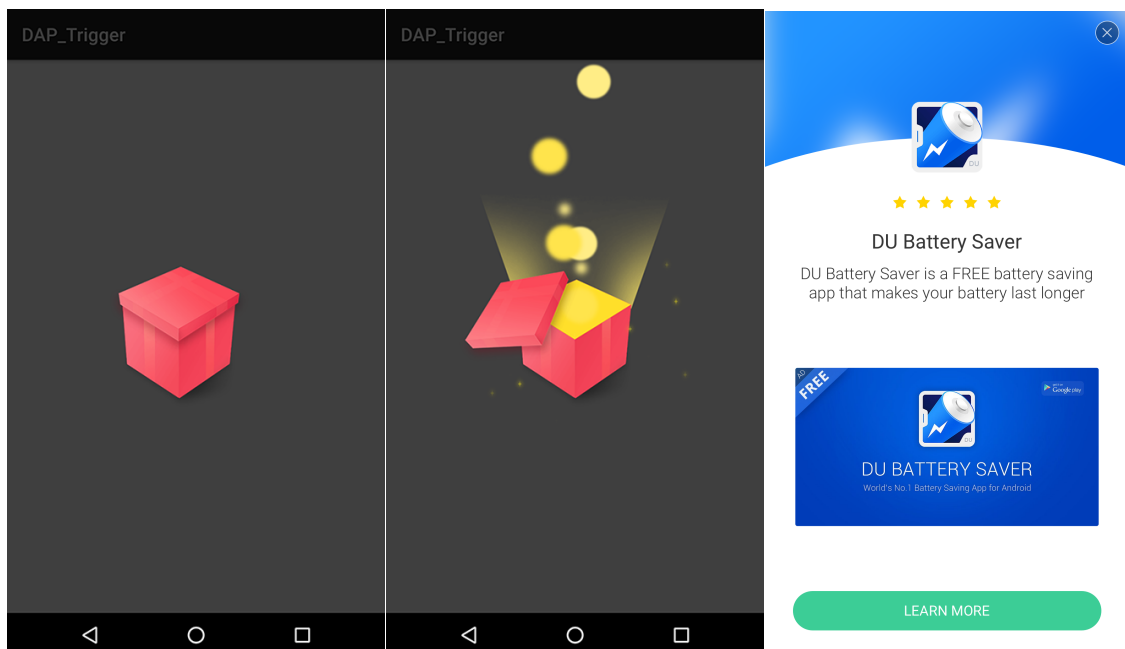
13. Request Trigger ad

Developer can find the best position in his app to add our trigger icon (with “zoom in” animation effects). After user click the trigger icon, a full screen interstitial ad will be shown.

Figure 7 A sample of trigger icon



Figure 8 A sample of trigger ad animation after clicking trigger icon



13.1 Trigger Ad configuration

1. Import DuappsAd-Trigger-xxx.aar, [see 4.1](#).

```
dependencies {  
  
    compile(name: 'DuappsAd-Trigger-xxx', ext: 'aar')  
  
    compile(name: 'DuappsAd-CW-xxx-release', ext: 'aar')  
  
}
```

2. Configure AndroidManifest.xml

```
<activity  
    android:name="com.trigger.ad.DuTriggerADActivity"  
    android:configChanges="keyboard|keyboardHidden|orientation|screenSi  
ze"  
    android:screenOrientation="portrait"  
    android:excludeFromRecents="true"  
    android:theme="@android:style/Theme.Translucent.NoTitleBar" />
```

13.2 Trigger configuration

Add a custom view for trigger icon in the xml of trigger icon page, and configure custom attributes, which include:

1. **Trigger:Pid**

Trigger:Pid="[Your_Placement_ID](#)"

* **Note:** **This attributes is the mandatory.** And the PID used for trigger Ad need to be added into “native” part JSON during initialization, see [Section 5](#).

```
{  
    "native": [  
        {  
            "pid": "YOUR\_PLACEMENT\_ID"  
        }  
    ]  
}
```

2. Trigger:AutoStart

Trigger:AutoStart="true"

If set it as "true", the trigger icon "zoom in" animation will be played automatically. The default value is "true".

Trigger:AutoStart="false"

If set it as "false". the trigger icon "zoom in" animation will not be played automatically. Then developer can call the below interface when need.

- **Interface Instruction:**

public void startAnim(int delay)

Parameters	Description
int delay	The delay time before playing animation (unit: ms)

- **Code sample:**

```
TriggerIconView icon = (TriggerIconView)
                    findViewById(R.id.trigger_icon);
icon.startAnim(500);
```

3. Trigger:Delay

Trigger:Delay=""

When Trigger:AutoStart= "true", developer can set the the delay time before playing animation, (unit: ms). The default value is 500ms.

- **Code sample:**

```
<com.trigger.view.TriggerIconView
xmlns:Trigger="http://schemas.android.com/apk/res-auto"
    android:id="@+id/trigger_icon"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    Trigger:Pid="Your_Placement_ID"
    Trigger:AutoStart="true"
    Trigger:Delay="500"/>
```