DU Ad Platform SDK for Android Access Guide

DuAD_SDK_CW1.0.5

Baidu Online Network Technology (Beijing) Co., Ltd

No.	DUAd10120150810
Date	2016-06-02
Ver.	1.0.5
Email	support_duad@baidu.com

Contents

1 Introduction

This document describes how to access **DU Ad Platform SDK** for Android apps of Baidu developers.
Baidu (http://ad.duapps.com) offers advertising services for Android apps. For example, DU Ad Platform SDK is a product that provides native ads.

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 package, etc.**	**DUAd_CWxxxx.zip**
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 Access Workflow

The access workflow of **DU Ad Platform SDK** to Android app is as below:

- The access workflow of DU Native Ad:

1. Apply for Location_ID, App_ID and Placement_ID. *See Section 3.*
2. Load **DU Ad Platform SDK** work package; configure Androidmanifest.xml. *See Section 4.*
3. **DU Ad Platform SDK** initialization. *See Section 5.*
4. Access Du Native Ad. See *Section 7. Section 8.*

- The access workflow of DU advertising wall:

1. Apply for Location_ID, App_ID and Placement_ID. *See Section 3.*
2. Load **DU Ad Platform SDK** work package; configure Androidmanifest.xml. *See Section 4.*
3. **DU Ad Platform SDK** initialization. *See Section 5.*
4. Access Du advertising wall. *See Section 9.*

- The access workflow of DU Interstitial Ad:

5. Apply for Location_ID, App_ID and Placement_ID. *See Section 3.*
6. Load **DU Ad Platform SDK** work package; configure Androidmanifest.xml. *See Section 4.*
7. **DU Ad Platform SDK** initialization. *See Section 5.*
8. Access Du Interstitial Ad. See *Section10.*

3 Obtain Identity

This section describes the three IDs needed during **DU Ad Platform_SDK** access: APP_ID, Location_ID and Placement_ID.

3.1 APP_ID

A. Definition
APP_ID is the unique identifier of a developer's APP at Baidu Developer Platform. Each app will have its own App_ID.

B. Obtain method
Visit Baidu Developer Platform http://ad.duapps.com to apply.

C. Code
`app_license`

3.2 Location_ID

A. Definition
Location_ID is the identifier of an ad's location at Baidu Developers Platform. Developers can create multiple Ad locations.
* **Note:** Developers can create multiple Ad locations.

B. Obtain method
Visit Baidu Developer http://ad.duapps.com to apply.

C. Code
`Pid`

3.3 Placement_ID

A. Definition
Placement_ID is the identifier of an ad's location at Facebook.

B. Obtain method
Visit Facebook Developers https://developers.facebook.com to apply.

C. Code
`fbids`

4 Loading and Configuration

This section describes how to load the **DU Ad Platform SDK** package, how to configure the *AndroidManifest.xml* file, and how to obfuscate code against project needs.

4.1 Load and Unzip the Package

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

- Package name: *DuAD_SDK_CW1.0.5.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 file:

  DuappsAd_CW_Online_v1.0.5.aar

- **DUAd_SDK_DEMO**

  This folder stores the example programs that use **DU Ad Platform SDK**. All interfaces in this document can be found in corresponding usage examples in this folder.

C. **Load** DU Ad Platform SDK

- **When using Android Studio:**

1) Copy the **SDK** aar file 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_v1.0.5', ext:
'aar')
}
```

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

- **When using Eclipse:**

1) Change the suffix of *DuappsAd_CW_Online_v1.0.5.aar* to zip and unzip it.

2) Configure the unzipped *DuappsAd_CW_Online_v1.0.5* to project file and import it into Eclipse.

4.2 Configure AndroidManifest.xml

In Android Project, open *AndroidManifest.xml* and perform the following actions to finish configuration:

A. Add permissions. Least Privilege of **DU Ad Platform SDK** as shown below:

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

B. Go to *app_license*, find `value`, and fill in the supplied App_ID as shown below. For more info about App_ID, *See* 3.1

```
    <application
        android:name="com.mobula.sample.MobulaApplication"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/mobulaTheme" >
    <meta-data
            android:name="app_license"
            android:value="xxxxxxxxxx" />
    <provider
         android:name="com.duapps.ad.stats.DuAdCacheProvider"
         android:authorities="packagename.DuAdCacheProvider"
         android:exported="false">
    </provider>
```

*** Note:** "packagename" is the full package name of developer's APP.

C. 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>
<activity android:name="com.duapps.ad.offerwall.ui.OfferWallAct"/>
```

Solution 2: Dynamically register the BroadcastReceiver for PACKAGE_ADDED.

If developers have 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 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:** "MyBroadcast" is the developer's own BroadcastReceiver for PACKAGE_ADDED.

4.3 Obfuscate Code

If you need to obfuscate code, the rules for obfuscating are shown below:

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.content.ContentProvider
```

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

5 Initialization

Before executing access, the Android App first needs to finish the initialization of **DU Ad Platform SDK**.

- **Method:**
  Go **Application Class** of **OnCreate** method, using *DuAdNetwork.init()*.
  * **Note:** Please use the interface according to the above requirements. Otherwise the initialization will be invalid.

- **Operation:**
  Go to *init* and use JSON format to write **String** data with mappings f or the **Placement_ID** and **Location_ID** as shown below:

```
{
   "native": [
      {
         "pid": "xxxxx",
         "fbids": [
         "xxxxxxxxxx ",
         ]
      }
   ],
   "offerwall": [
      {
         "fbids": "xxxxxxxxx",
         "pid": "xxxxx"
      }
    ]
}
```

***Note:** If developers don't have fbids, please leave the id blank. e.g. `"fbids": [ "" ]`.

- **Interface Instruction:**
  public static void *init*(Context context， String pidsJson);

Parameters	Description
Context context	Activity Context
String pidsJson	The relationship between Placement_ID a nd Location_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 Native Ad Data Obtainment

Obtaining Ad Data includes three parts: Constructing interfaces of Ad data classes, Populating Ad cache interfaces, and Retrieving Ad data interfaces.

6.1 Constructing Interfaces of Native Ad Data Classes

Steps are as shown below:

1) **Construct native Ad classes**

Create a native ad object that specifies the corresponding **Location_ID**. Different location can receive different data.

2) **Set Ad cache No.**

Developers can set Ad cache No.: 1–5;

Recommended Ad cache No.: 1–2;

If no Ad cache is set, or if an Ad cache is set with an invalid value, t he default cache of 2 will be used.

3) **Use native Ad related interfaces**

Interface Instructions:

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

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

Parameters	Description
Context context	Activity Context
int pid	Locaiton_ID
int cacheSize	Ad cache number

4) **The interface of setting Facebook ID for supporting passing the ID dynamically**

Interface Instructions:

public void setFbids (List<String> fbids);

Parameters	Description
List<String> fbids	Facebook's placementID

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

6.2 Populating Native Ad Cache Interfaces

According to their products' demands, developers can select the time to use Ad cache interfaces.

- Use the fill() to 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 can cache the data to client's memory. It **does not cache** Ad's **image data**. **It only caches the** image's URL address. The cached data is **small**.

- **Interface Instruction:**
  public void fill();

6.3 Retrieving Native Ad Data Interfaces

To retrieve Ad data, first register the callback interfaces of receiving the Ad data, and then retrieve Ad data interfaces.
The success or failure of Ad data retrieve, the respond of click is returned by callback interfaces. This process and Ad data retrieve are asynchronous, so as not to block developers' threads.

6.3.1 Register Callback Interfaces of Ad Data

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 Data Interfaces

Using *load* method, **DU Ad Platform SDK** can notify developers of Ad data result in callback function. Three types of results can be returned:

a) **Retrieve Ad successful. DU Ad Platform SDK** can callback *onAdLoaded* method. Through the object of **DuNativeAd**, developers can use *get* method to acquire specific Ad data contents. *See 7.2*

b) **Retrieve Ad error. DU Ad Platform SDK** can callback *onError* method. Developers can receive specific error information through the *onError* object. Error code and description of retrieve Ad error are shown in Table 2.

Table2 Error Code of Retrieve Ad Error（AdError）

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 Ad click event. DU Ad Platform SDK** can callback *onClick* method to inform developers that this DuNativeAd's object's Ad has been clicked.

- **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) {
      }
};
```

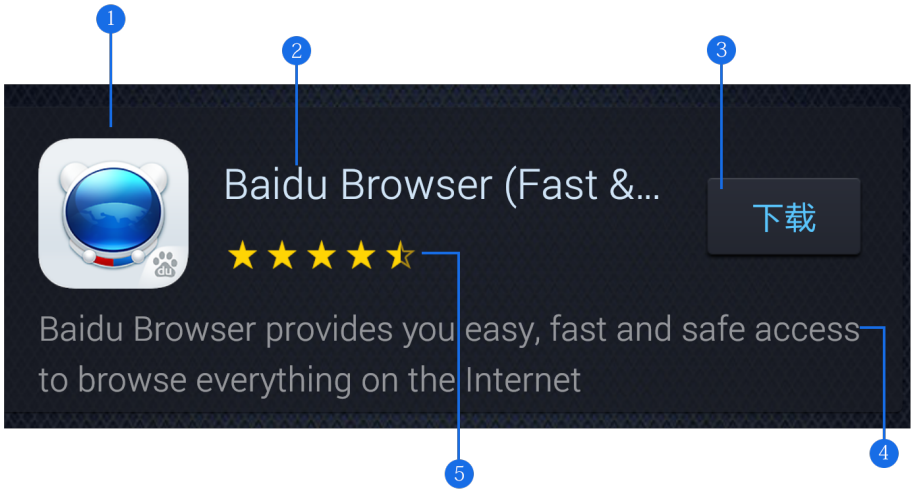
7. Native Ad Data Interface Introduction

This section describes Ad data's constituent elements and how to get the constituent elements' interfaces.

7.1 Constituent Elements

Ad data's constituent elements include: Logo, title, CTA button, promotion copy, rating, promotion image, etc. See Figure 3.

Figure 3 Constituent elements of Ad data



* ① Logo ② Title ③ CTA button ④ Promotion copy ⑤ Rating

7.2 Get Interfaces

Get interfaces of Ad data elements as shown below:

- Get **Logo** interface

  Interface Instruction:

  public String getIconUrl();

Return Value	Description
String iconUrl	Ad Logo's URL address

- Get **Title** interface

  Reserve space of at least 20 charactors to display the title. An ellipsis (…) can be used to indicate truncated text.

  * **Note:** Ad **must** include **one title.**

  Interface Instruction:

  public String getTitle();

Return Value	Description
String title	Ad's title

- Get **CTA button** interface

  * **Note:** Ad **must** include **one CTA button.**

  Advertisers can specify button copy, e.g. **Install Now**. Do not shorten or change the Ad's button copy.

  For button copy with promotion image, the **max** character length is **25**

  For button copy without image, the copy is usually defined as **Download**.

  Interface Instruction:

  public String getCallToAction();

Return Value	Description
String callToAction	Ad's CTA button copy

- Get **Promotion Copy** interface

  Ad can include promotion copy. Ensure that 72 characters can be displayed. If Ad space cannot display 72 characters, it is recommended that you do not include promotion copy in Ad or use scrolling text effects, unless all Ad copy can be displayed.

  Interface Instruction:

  public String getShortDesc();

Return Value	Description
String shortDesc	Ad promotion copy

- Get **Rating** interface

  Interface Instruction:

  public float getRatings();

Return Value	Description
float ratings	Returns the Ad's rating on Google Play.

- Get **Promotion Image** interface

  Promotion image can be included in Ad to increase user's desire to click the Ad.

  You can zoom and cut part of the image, but do not distort or change it.

  Promotion image size is usually: 1200x627 pixels.

  * **Note:** Not all Ads have promotion images.

  Interface Instruction:

  public String getImageUrl();

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

- **DuAdChoicesView**

  This view is the AdChoices corner mark from by facebook Native Ad. It's the mandatory element for facebook native Ad.

  * **Note:** The Native Ad which is not from facebook doesn't have the AdChoices corner mark.

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

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

8. Register the Listener of Native Ad View

DU Ad Platform SDK can automatically count the number of times an Ad is displayed and clicked. Therefore, developers must register a listener of View within the Ad's clickable region**.**

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. The use of advertising wall

Figure 4 A sample of Advertising wall

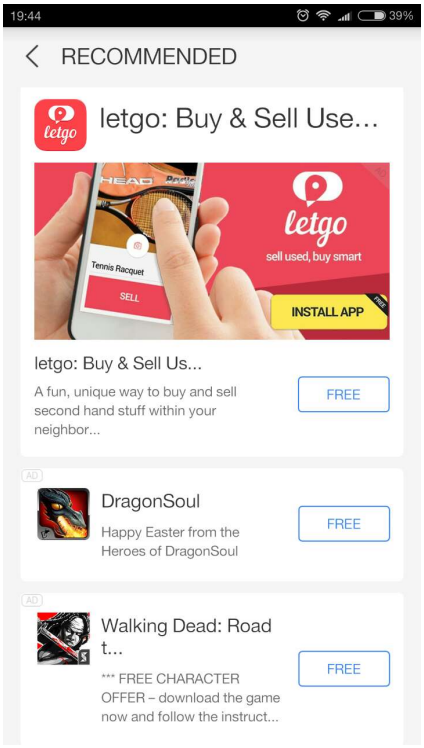


Figure 4 shows a sample of Advertising wall. Advertising wall is an encapsulated list Advertising (This is an Activity). For displaying the advertising wall, developers need to pass in the Location_ID when page jumping. For specific operation, please take the below example for reference.

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

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

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

The first red rectangle: The encapsulated activity class of advertising wall.

The second rectangle: The Location_ID needs to be passed in (This is the Location_ID applied via DU AD Platform).

10. The use of Interstitial Ads

Figure 5 A sample of half screen interstitial Ads

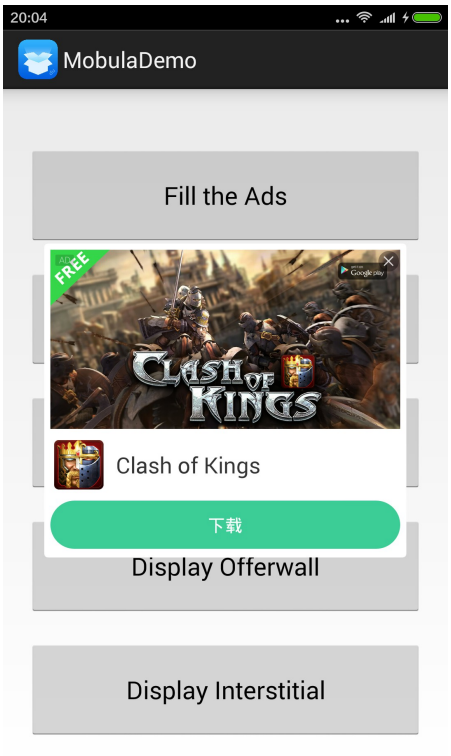


Figure 6 A sample of full screen interstitial Ads

10.1 Constructing Interfaces of Interstitial Ads Class

- **Constructor:**

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

Parameters	Description
Context context	ACTIVITY CONTEXT
int pid	Location_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 ",]
        }
    ],
    "offerwall": [ ]
}
```

10.2 Register Callback interfaces of Interstitial Ads

Interface Instruction:

public void setInterstitialListener (AbsInterstitialListener mAbdI);

Parameters	Description
AbsInterstitialListener mAbdI	Listener Abstract class

Callback function returns:

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

10.3 Load Interstital Ads

interface Instruction:
public void load ();

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

10.4 Show Interstital Ads

interface Instruction:
public void show ();

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