DU Ad Platform_SDK for Android 接入手册

DUAd_SDK_HW1.1.1.2

百度在线网络技术（北京）有限公司

文档编号：DUAd10120150810
发布日期：2017-10-31
版本：1.1.1.2

联系方式：support_duad@baidu.com

目 录

1. 概述

本文档描述如何在安卓应用中接入来自百度开发者平台的 DU Ad Platform_SDK 产品。百度开发者平台（http://ad.duapps.com）可以为安卓应用提供广告服务。DU Ad Platform_SDK 是百度开发者平台中用来提供原生广告的一款产品。

1.1 读者对象

本文档面向的读者是安卓应用的开发者。

1.2 前提

DU Ad Platform_SDK 目前支持 Android2.3 API Level9（含）以上的系统版本。

2. 接入流程

DU Ad Platform_SDK 的接入流程如下：

- **原生广告接入流程:**

1. 申请广告 ID *参见第 3 章;*
2. 导入 DU Ad Platform_SDK 工程包 *参见第 4 章;*
3. 初始化 DU Ad Platform_SDK *参见第 5 章;*
4. 原生广告接入 *参见第 7 章,第 8 章;*
5. 完成接入.

- **原生广告 LIST 接入流程:**

1. 申请广告 ID *参见第 3 章;*
2. 导入 DU Ad Platform_SDK 工程包 *参见第 4 章;*
3. 初始化 DU Ad Platform_SDK *参见第 5 章;*
4. 原生广告 LIST 接入 *参见第 9 章*
5. 完成接入.

3. 获取身份

本章描述 DU Ad Platform_SDK 接入过程中需要的三个身份：APP_ID，广告位 ID，Placement_ID。

3.1 APP_ID

A. 定义
APP_ID 是开发者的应用在广告平台的唯一标识。

B. 获取方式
访问百度开发者平台 http://duad.baidu.com/进行申请。

C. 代码
app_license

3.2 广告位 ID

A. 定义
广告位 ID 是开发者平台上广告所在的广告位置的标识。开发者可以创建多个广告位。

B. 获取方式
访问百度开发者平台 http://duad.baidu.com/进行申请。

C. 代码
pid

3.3 Placement_ID

A. 定义
Placement_ID 是 Facebook 广告所在广告位置的标识。

B. 获取方式
访问 Facebook 开发者平台 https://developers.facebook.com 进行申请。

C. 代码
fbids

4. 加载与配置

本章描述在安卓应用中如何加载 DU Ad Platform_SDK 的压缩包，如何配置 AndroidManifest.xml，以及根据项目需要配置混淆代码。

请严格按照本章进行配置，否则有可能会出现运行异常。

4.1 加载 DU Ad Platform_SDK 压缩包

1． 下载 DU Ad Platform_SDK 的压缩包。
名称：*DuAD_SDK_HW_xxxx.zip*

2． 解压 DU Ad Platform_SDK 的压缩包。解压后有两个子目录文件夹，名称和内容如下：

- DUAd_SDK：
  该文件夹存放 DU Ad Platform_SDK 的 aar 包：*DuappsAd-HW-xxxx.aar*

- DUAd_SDK_DEMO
  该文件夹存放使用 DU Ad Platform_SDK 过程中的示例程序。本文档中所有接口都可以在 DUAd_SDK_DEMO 中找到对应的使用示例

3． 加载 DU Ad Platform_SDK：

- Android Studio 导入:
  1). 拷贝 SDK aar 包放到你的安卓工程文件根目录的 *libs* 目录下。
  2). 然后配置
  build.gradle :

```
repositories {
    flatDir {
        dirs 'libs'
    }
}
dependencies {
    compile fileTree(include: ['*.jar'], dir: 'libs')
    compile(name: 'DuappsAd-HW-xxx-release', ext: 'aar')
}
```

***注: flatDir 指定的位置 即为 aar 存放的位置**

- Eclipse 导入:

1) 新建一个 Eclipse android library.
2) 将 DuappsAd-HW-xxx.aar 后缀改成 zip 解压.
3) 将 classes.jar 拷进 libs 目录下
4) 用解压出的 AndroidManifest.xml 替换新建 Eclipse android library 中的 manifest.
5) 用解压出的 res 文件夹替换新建 Eclipse android library 中的 res 文件夹.

4.2 配置 AndroidManifest.xml

在安卓工程目录下，打开 AndroidManifest.xml，配置以下内容：

A. 添加权限。DU Ad Platform_SDK 使用的最低权限如下：

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

B. 在 app_license 的 value 中填入已申请的 APP_ID。APP_ID 的申请，*参见 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>
```

*注： “packagename”为开发者 APP 的包名全称。

C． 注册 APP 安装广播监听

```
<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>
```

4.3 混淆代码

请务必按如下混淆规则对应用代码进行混淆,否则有可能会出现运行异常:

A 把 DU Ad Platform_SDK 中的类排除在混淆之外；

```
-keep class com.duapps.ad.**{*;}
-dontwarn com.duapps.ad.**
```

B 将以下类添加到 proguard 配置：

```
-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.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>;}
```

***注：混淆方法参见 Android 官方混淆文档：**${ android-sdk }/tools/proguard/

C 如接入 Facebook 广告，须将下类可以添加到 proguard 配置：

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

D 如接入 Admob 广告，须将下类可以添加到 proguard 配置：

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

5. 初始化

在完成 DU Ad Platform_SDK 接入操作之前，安卓应用首先需要对 DU Ad Platform_SDK 做初始化。

- **方法：**

  在 application 的 OnCreate 方法中使用 *DuAdNetwork.init()*。

  ***注：请按上述要求使用该接口.否则可能导致初始化无效。**

  开发者可以在 application 的 OnCreate 方法中使用 DuAdNetwork.setLaunchChannel(); 用以区分不同 app 投放渠道的数据。此接口可选择使用，不是必须。

- **操作：**

  在 *init* 中，按照下面的 *json* 格式填写 String 数据。将 Placement_ID 与广告位 ID 建立对应关系。具体如下：

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

  ***注：**

  a) 如果开发者某广告位不需要 Facebook 广告，请将该广告位对应的"fbids"部分删除。该广告位就不会去请求所删除渠道的广告。

  b) 如需绑定 FBID，请确保 Facebook Audience Network 版本不低于 FB 4.23.0

- **接口说明：**

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

参数	说明
Context context	ACTIVITY CONTEXT
String pidsJson	Placement_ID 与广告位 ID 的对应关系

- **接口说明：**

  public static void setLaunchChannel (String channelName);

参数	说明
String channelName	此接口可以帮你根据你的 APP 投放渠道，区分数据。此接口可选择使用，不是必须。

- **代码示例：**

```
/*** 从assets中读取txt*/
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. 获取原生广告数据

本章描述如何获取广告数据。包括构造广告数据类接口，填充广告缓存接口，和获取广告数据接口三个部分。

6.1 构造原生广告数据类接口

步骤如下：

1. **构造原生广告类**
   创建原生广告对象必须指定对应的广告位 ID。不同的广告位会获取到不同的广告数据。

2. **设置广告缓存个数**
   广告缓存个数可以设置 1-5 个。
   推荐不设置广告缓存个数。如果不设置或者设置无效值，会使用默认缓存：1 个。

3. **使用原生广告相关接口**
   接口说明：
   public DuNativeAd (Context context, **int** pid)
   public DuNativeAd (Context context, **int** pid, **int** cacheSize)

参数	说明
Context context	ACTIVITY CONTEXT
int pid	广告位 ID
int cacheSize	缓存广告个数

4. **设置 Facebook ID ,支持动态传入 ID 的接口**
   接口说明：
   public void setFbids (List<String> fbids);

参数	说明
List<String> fbids	Facebook 的 placementID

*注: 如果需要使用此接口，必须在 Json 中配置该 pid 对应的 fbids（参见第五章），然后参数 List<String> fbids 会覆盖前面 Json 里配置的该 pid 对应的 fbids.

6.2 填充原生广告缓存接口

开发者可根据自己产品的需求，选择时机使用填充广告缓存接口。

- 调用 fill()接口可以提前缓存广告，在 load()广告时可以更快展示。
  建议：在广告展示的前置场景调用 fill()。
  ***注：**广告数据会缓存到客户端内存中，**不会缓存广告的图片数据，只会缓存图片的 Url 地址，缓存数据量小。**

- **接口说明：**
  public void fill();

6.3 获取原生广告数据接口

获取广告数据，首先要注册接收广告数据的回调，然后获取广告数据接口。
广告数据获取成功或失败，点击事件的响应是通过回调接口返回的。此过程与广告数据获取过程是异步的，不会阻塞开发者的线程。

6.3.1 注册接收广告数据回调接口

接口说明：

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

参数	说明

DuAdListener	回调函数返回获取广告错误，获取广告的数据，广告点击事件。
	public interface DuAdListener {
	public void onError(DuNativeAd ad, AdError error);
	public void onAdLoaded(DuNativeAd ad);
	public void onClick(DuNativeAd ad);
	}

6.3.2 获取广告数据接口

使用 load 方法后，DU Ad Platform_SDK 会在回调函数中通知开发者获取广告数据的结果。结果有三类，具体如下：

a) **获取广告成功。**DU Ad Platform_SDK 会回调 onAdLoaded 方法，通过 DuNativeAd 的对象，开发者可以用 get 方法得到具体的广告数据内容。*参见 7.2。*

b) **获取广告失败。**DU Ad Platform_SDK 会回调 onError 方法，开发者可以通过 onError 的对象获取到具体错误信息。获取广告数据失败的错误码及含义，*参见表 2。*

表 2　　获取广告数据失败的错误码（AdError）

常量	错误码	说明
NETWORK_ERROR_CODE	1000	客户端网络错误
NO_FILL_ERROR_CODE	1001	没有获取到广告数据
LOAD_TOO_FREQUENTLY_ERROR_CODE	1002	请求接口过频繁
IMPRESSION_LIMIT_ERROR_CODE	1003	展示超出限制
SERVER_ERROR_CODE	2000	服务器错误
INTERNAL_ERROR_CODE	2001	服务器网络错误
TIME_OUT_CODE	3000	获取广告数据等待时间超时
UNKNOW_ERROR_CODE	3001	未知错误

c) **获取广告点击事件。**DU Ad Platform_SDK 会回调 onClick 方法，通知开发者该 DuNativeAd 的对象的广告被点击。

- **接口说明：**
  `public void load();`

- **代码示例：**

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

```
DuAdListener mListener = new DuAdListener () {
        @Override
        public void onError (DuNativeAd ad, AdError error) {
                Log.d(TAG, "onError : " + error.getErrorCode());
        }

        @Override
        public void onClick (DuNativeAd ad) {
        }

        @Override
        public void onAdLoaded (final DuNativeAd ad) {
        }
  };
```

6.4 获取原生广告缓存接口

6.4.1 判断缓存广告接口

- **接口说明：**

  public Boolean isHasCached ();

  此接口可以获取到当前是否有广告缓存。

6.4.2 获取缓存广告接口

- **接口说明：**

  public DuNativeAd getCacheAd();

此接口可以直接获取到当前广告对象的广告缓存。可以循环拿取,一直到广告缓存为 0。

*注： 在使用该接口展示广告时,请务必做非空判断,避免缓存池为空,导致空指针。

6.5 销毁原生广告对象

- **接口说明：**

  public void destory ();

  在退出原生广告展示界面时， 建议销毁原生广告对象。

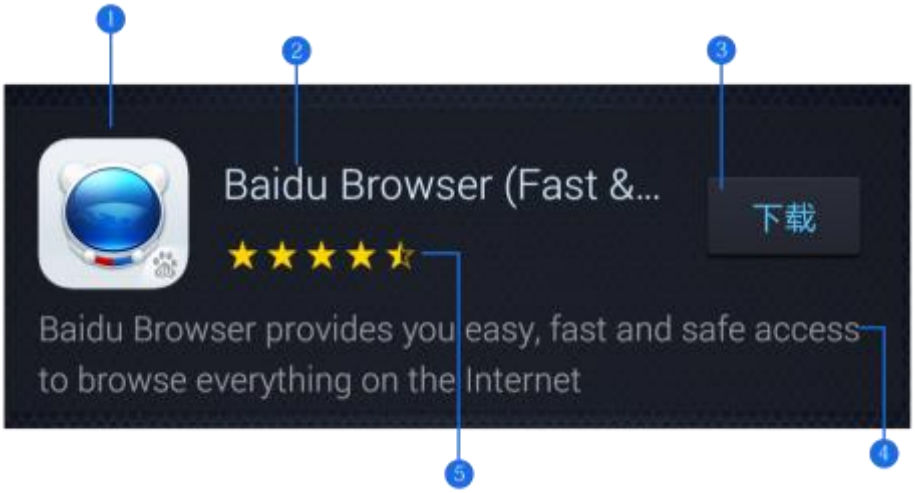
7. 原生广告数据介绍

本章描述广告数据的构成元素及构成元素的获取接口。

7.1 构成元素

广告数据的构成元素包括图标，标题，CTA 按钮，宣传文案，评价，和宣传图等。

图 2 广告数据的构成元素



*注： ① 图标 ② 标题 ③ CTA 按钮 ④ 宣传文案 ⑤ 评级

7.2 获取接口

广告数据构成元素的获取接口如下：

- **图标**获取接口

接口说明：

public String getIconUrl();

返回值	说明
String iconUrl	广告图标的 Url 地址。

- **标题**获取接口

***注：**广告中**必须包含**一个**标题**。
保留**至少 20 个字符**的空间用来显示标题，可以用省略号代替超出的文本。

接口说明：

public String getTitle();

返回值	说明
String title	广告标题。

- **CTA 按钮**获取接口

*注：广告中**必须包含**一个**触发按钮**。
广告商可以指定按钮的文案，如 **Install Now**。不要缩短或改变广告的按钮文案。
有宣传图的按钮文案的最大字符长度个数：25。
无宣传图的按钮文案通常为：**Download**。

接口说明：
public String getCallToAction();

返回值	说明
String callToAction	广告的按钮文案。

- **宣传文案**获取接口

广告中可以添加宣传文案。需确保有 72 个字符可以被显示。
如果广告区域不足以显示 72 个字符，建议不要在广告中添加宣传文案，或者使用滚动文本效果，让全部宣传文案能够被显示。

接口说明：
public String getShortDesc();

返回值	说明
String shortDesc	广告的宣传文案。

- **评级**获取接口

接口说明：
public float getRatings();

返回值	说明
float ratings	返回的是该广告应用在 google play 上的评级。

- **宣传图**获取接口

广告中可以添加宣传图片，促进用户点击广告的欲望。可以缩放和裁剪宣传图的一部分，但请不要扭曲和改变它。宣传图的大小通常是：796*416 像素。
*注：部分广告没有宣传图。

接口说明：
public String getImageUrl();

返回值	说明
String imageUrl	广告宣传图的 Url 地址，当返回值为 NULL 时，当前广告数据中不含宣传图。

- **DuAdChoicesView**

该 View 是 Facebook 原生广告返回的 AdChoices 角标. Facebook 原生广告必须添加的元素.

*注: 非 Facebook 原生广告没有 AdChoices 角标。

构造方法:

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

使用方法: 给 AdChoices 创建单独 View 控件.它与 AD 角标不同.

接口说明：

public void addView(DuAdChoicesView choicesView);

返回值	说明
DuAdChoicesView choicesView	DuAdChoicesView 对象.

- **代码示例：**

```
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. 注册原生广告 View 监听

DU Ad Platform_SDK 会自动统计广告的展示和被点击次数，所以开发者必须注册广告可点击区域视图的监听。

接口说明：

public void registerViewForInteraction(View view)

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

返回值	说明
View view	广告内容中可点击的 view
List<View> views	更细致的子 view

* 注： 不建议在多线程使用此接口。

9. 原生广告 List 使用

*注:

a) 全部拉取广告已经在异步进行,请在主线程调用

b) 建议需要同时展示多条原生广告时使用该方法

9.1 创建原生广告 list 使用类

- 接口说明：

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

参数	说明
Context context	ACTIVITY CONTEXT
int pid	广告位 ID，该 pid 注册在 json 的 list 数组中
int cacheSize	原生广告 list 缓存广告个数

- **代码示例：**

```
DuNativeAdsManager adsManager = New DuNativeAdsManager(getApplicatio
nContext(),PID, cacheSize);
```

*注: 原生广告 list 管理类

9.2 创建原生广告 list 监听接口

- **代码示例：**

```
AdListArrivalListener listener = new AdListArrivalListener()
{
    //回调广告 list
    @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++)
        {
            //获取单个广告对象
            NativeAd ad = (NativeAd) arg0.get(i);
            rootContainer.addView(createItem(ad));
            //给单个广告对象添加广告监听（注册单个广告点击事件,
callback 参见 9.3)
            ad.setMobulaAdListener(callback);
        }
        Log.d(TAG, "------end to fill view-----");
    }
    //返回广告错误码
```

```
        @Override
        public void onAdError(AdError arg0) {
            Log.d(TAG, "onError : " + arg0.getErrorCode());
            }
    };
```

9.3 原生广告点击回调接口

此接口可以获得单个广告点击事件, onAdLoaded(),onAdError()无回调.

- **代码示例：**

```
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");
        }
};
```

9.4 注册广告监听接口

- 接口说明：

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

参数	说明
AdListArrivalListener listener	广告返回监听器

- **代码示例：**

```
DuNativeAdsManager adsManager = new DuNativeAdsManager(
        getApplicationContext(), PID, cacheSize);
//设置原生广告 list 接口监听,在回调里面获取广告数据, (listener 参见 9.2).
adsManager.setListener(listener);
```

9.5 拉取原生广告 list

9.5.1 广告填充接口

开发者可根据自己产品的需求，选择时机使用填充广告缓存接口。

- **接口说明：**

  public void fill();

  建议：在广告展示的前置场景调用 fill()。调用 fill()接口可以提前缓存广告，在 load()广告时可以更快展示。

  ***注：**广告数据会缓存到客户端内存中，**不会缓存广告的图片数据，只会缓存图片的 Url 地址，缓存数据量小。**

- **代码示例：**

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

9.5.2 广告拉取接口

使用 load 方法后，DU Ad Platform_SDK 会在回调函数（参见 9.2）中通知开发者获取广告数据的结果。

- 接口说明：

public void load();

- **代码示例：**

```
adsManager.load();
```

*注: 拉取广告接口,调用此接口去开始拉取原生广告 list.

9.6 销毁原生广告 list 对象与监听接口

在退出原生广告 list 展示界面的时候,需要销毁原生广告 list 对象(DuNativeAdsManager)和原生广告 list 监听(AdListArrivalListener).

- **代码示例：**

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