

DU Ad Platform_SDK for Android接入手册

Version: DUAd_SDK_CW1.2.7.4

DU Ad Platform_SDK for Android接入手册

1. 概述
2. 接入流程
3. 获取身份
 - 3.1 APP_ID
 - 3.2 广告位 ID
4. 加载与配置
 - 4.1 加载 DU Ad Platform_SDK 压缩包
 - 4.2 配置 AndroidManifest.xml
 - 4.3 混淆代码
 - 4.4 Kotlin 加载（可选）
5. 初始化
6. 控制用户信息获取许可状态
 - 6.1 用户信息获取许可状态的设置接口
 - 6.2 用户信息获取许可状态的获取接口
7. 获取原生广告数据
 - 7.1 构造原生广告数据类接口
 - 7.2 注册广告数据监听回调接口
 - 7.3 获取广告数据接口
 - 7.4 销毁原生广告对象
8. 原生广告数据介绍
 - 8.1 构成元素
 - 8.2 数据获取接口
9. 注册原生广告 View 监听
10. 原生广告 List 使用
 - 10.1 构造原生广告 List 使用类
 - 10.2 构造子原生广告类
 - 10.3 注册原生广告 List 监听接口
 - 10.4 注册原生广告子类监听接口
 - 10.5 获取广告数据接口
 - 10.6 原生广告数据获取
 - 10.7 销毁原生广告 List 对象
11. 广告墙使用
 - 11.1 注册广告墙 Activity
 - 11.2 设置相关参数
12. 插屏广告使用
 - 12.1 构造插屏广告对象
 - 12.2 注册插屏广告回调
 - 12.3 使用插屏广告
13. 横幅广告使用

- 13.1 构造横幅广告对象
- 13.2 注册横幅广告回调
- 13.3 设置相关参数
- 13.4 添加横幅到布局中
- 13.5 使用横幅广告
- 14. 常见问题
 - 14.1 SDK接入
 - 14.2 平台
 - 14.3 广告
 - 14.4 其他

1. 概述

本文档描述如何在安卓应用中接入来自DAP开发者平台的 DU Ad Platform_SDK 产品。

[DAP开发者平台](#)可以为应用提供广告服务。DU Ad Platform_SDK 是DAP开发者平台中用来提供原生广告的一款产品。

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：

1. 申请广告 ID
2. 导入 DU Ad Platform_SDK 工程包
3. 初始化 DU Ad Platform_SDK
4. 广告接入
5. 完成接入

3. 获取身份

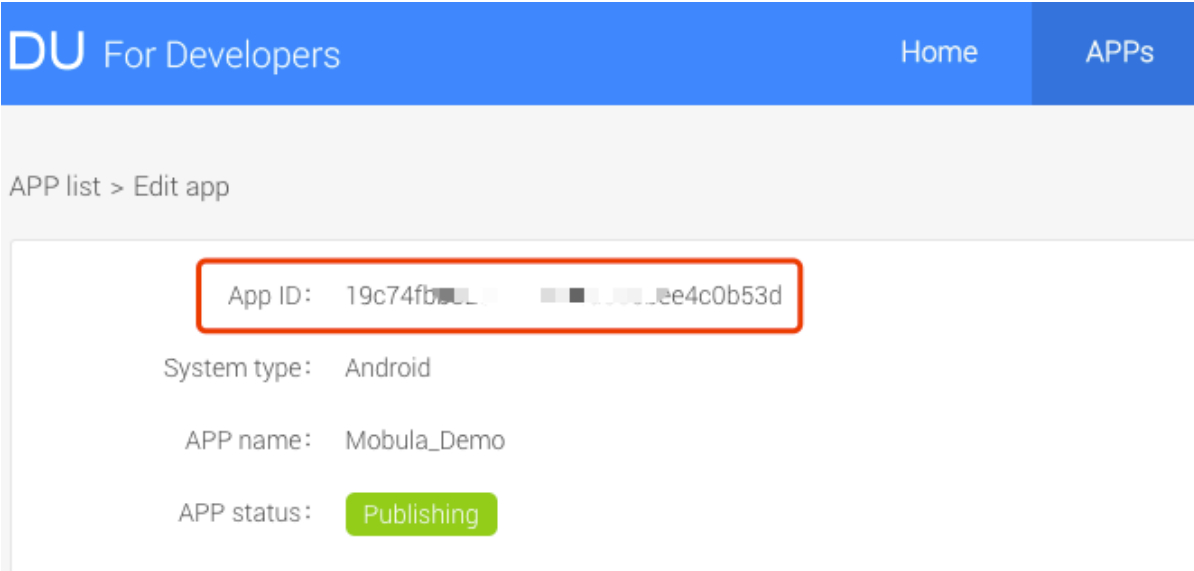
本章描述 DU Ad Platform_SDK 接入过程中需要的四个身份：APP_ID，广告位 ID。

3.1 APP_ID

1. 定义

APP_ID 是开发者的应用在 DAP 广告平台的唯一标识。
2. 获取方式

访问[DAP开发者平台](#)进行申请。



3. 代码

app_license

3.2 广告位 ID

1. 定义

广告位 ID 是 DAP 开发者平台上广告所在的广告位置的标识。开发者可以为一个应用创建多个广告位。

2. 获取方式

访问[DAP开发者平台](#)进行申请。

Edit placement information Create placement

Name	Status	PID	Ad Format	The corresponding PID	Steps to trigger Ads	Creation time	Action
Mobula_Demo推广位	Running	71. 1132	Native			2017-05-19	
测试用-原生推广位	Running	1. 1179	6			2017-05-19	

3. 代码

pid

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。

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-CW-xxx.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:

拷贝 SDK aar 包放到你的安卓工程文件根目录的 libs 目录下，然后配置 build.gradle :

```
repositories {
    flatDir {
        dirs 'libs'
    }
}

dependencies {
    compile fileTree(include: ['*.jar'], dir: 'libs')
    compile(name: 'DuappsAd-CW-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-CW-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：

1. 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" />
```

2. 在 app_license 的 value 中填入已申请的 APP_ID。

```

<application
    android:name="com.mobula.sample.MobulaApplication"
    android:usesCleartextTraffic="true" //target SDK 28 要添加
    android:icon="@drawable/ic_launcher"
    android:label="@string/app_name"
    android:theme="@style/mobulaTheme" >
    <meta-data
        android:name="app_license"
        android:value="YOUR_APP_LICENSE" />
</application>

```

注：“applicationId”必须与平台注册的应用包名一致，否则将无法获得广告；“applicationId”在build.gradle中设置；平台的应用包名见下图。

Basic Information	Filter Settings
Edit app information * APP name: Mobula_Demo * APP type: TOOLS * Keywords: * Package name: com.mobula.sample APP description:	

3. 为了适配**Android 9.0 (target SDK 28)**，需要在Application标签下添加**android:usesCleartextTraffic="true"**来解析非https的广告链接

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:

1. 将以下类添加到 proguard 配置:

```

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

-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/](#)

4.4 Kotlin 加载（可选）

接入Kotlin应用的时候，除了上述步骤之外，还需要额外进行如下配置。

1. 在app下的build.gradle中加入：

```
apply plugin: 'kotlin-android'
apply plugin: 'kotlin-android-extensions'

.....

dependencies{
    .....
    implementation "org.jetbrains.kotlin:kotlin-stdlib-
jre7:$kotlin_version"
}
```

2. 在project下的build.gradle中加入：

```
buildscript {
    xt.kotlin_version = '1.2.51'
    .....
    dependencies {
        classpath "org.jetbrains.kotlin:kotlin-gradle-
plugin:$kotlin_version"
        .....
    }
}
```

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。

没有进行初始化的广告位 id 无法拉取广告。

1. 创建 json 文件，将 Placement_ID 与广告位 ID 建立对应关系。具体格式如下：

```

{
  "native": [
    {
      "pid": "YOUR_DAP_PLACEMENT_ID"
    },
    {
      "pid": "YOUR_DAP_PLACEMENT_ID"
    }
  ],
  "list": [
    {
      "pid": "YOUR_DAP_PLACEMENT_ID"
    }
  ],
  "offerwall": [
    {
      "pid": "YOUR_DAP_PLACEMENT_ID"
    }
  ]
}

```

注：如果不希望通过静态创建 json 文件的方式进行初始化，可以直接创建符合 json 格式的字符串并传值。

2. 在 application 的 `onCreate()` 方法中使用 `DuAdNetwork.init()`

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

Java代码示例：

```

public void onCreate() {
    super.onCreate();
    //初始化 SDK
    DuAdNetwork.init(this, getConfigJSON(getApplicationContext()));

    //DuAdNetwork.setLaunchChannel("YOUR_APP_CHANNEL");
}

//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
    }
}

```

Kotlin代码示例:

```

override fun onCreate() {
    super.onCreate()
    DuAdNetwork.setConsentStatus(this, true)
    DuAdNetwork.getConsentStatus(this)
    /**
     * the sdk initialization 初始化SDK
     */
    DuAdNetwork.init(this, getConfigJSON(getApplicationContext()))
}

fun getConfigJSON(context: Context): String {
    var bos = ByteArrayOutputStream()
    var bis = BufferedInputStream(context.assets.open("json.txt"))

    try {
        var readLen = -1
        while (bis.read().also { readLen = it } != -1) {
            bos.write(readLen)
        }
    }
}

```

```
    } catch (e: Exception) {
        Log.e("", "IOException :" + e.message)
    } finally {
        closeQuietly(bis)
    }
    return bos.toString()
}
```

3. 填写投放渠道用以区分不同 app 投放渠道的数据，此接口可选择使用，不是必需。

在 application 的 onCreate() 方法中使用 `DuAdNetwork.setLaunchChannel()`

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。

6. 控制用户信息获取许可状态

此配置为针对GDPR做出的修改，适用于需要进行用户信息获取许可状态配置的地区，为可选配置。

6.1 用户信息获取许可状态的设置接口

建议在初始化时调用该接口。

接口说明：

```
public static void setConsentStatus(Context context, boolean consentStatus)
```

参数	说明
Context context	ACTIVITY CONTEXT
boolean consentStatus	用户许可状态 true：获得了用户授权时传入，可以按正常方式请求和展示广告。 false：用户拒绝授权或收回授权时传入，广告请求直接返回4000（见7.2错误码）

6.2 用户信息获取许可状态的获取接口

接口说明：

```
public static boolean getConsentStatus(Context context)
```

获取当前用户信息获取许可状态，允许收集用户信息则返回 True，否则返回False。

7. 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。

7.1 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。

注：此方法只在通过 DU Ad Platform 聚合其他渠道时生效。

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，该 pid 注册在 Json 的 native 数组中
int cacheSize	缓存广告个数

7.2 注册广告数据监听回调接口

广告数据获取与点击事件的响应均通过回调接口返回。此过程与广告数据获取过程异步，不会阻塞开发者的线程。

接口说明：

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

参数	说明
DuAdListener adListener	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);
}
```

使用获取数据方法后，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。

- 获取广告成功

DU Ad Platform_SDK 会回调 `onAdLoaded()` 方法，通过 DuNativeAd 的对象开发者可以得到具体的广告数据内容。

- 获取广告失败

DU Ad Platform_SDK 会回调 `onError()` 方法，通过 AdError 对象开发者可以得到具体错误信息。获取广告数据失败的错误码及含义如下：

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
UNKNOWN_ERROR_CODE	3001	未知错误
NO_USER_CONSENT_ERROR_CODE	4000	用户信息获取未受到许可

- 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。

7.3 获取广告数据接口

开发者可根据自己产品的需求，选择时机获取广告数据。

接口说明：

```
public void fill()
```

调用 `fill()` 接口可以提前缓存广告，在 `load()` 广告时可以更快获取。建议在广告展示的前置场景调用该方法。

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。

```
public void load()
```

异步获取广告对象数据，没有缓存时会进行广告请求。

建议在使用 `load()` 后再次调用 `fill()` 接口进行广告缓存。

```
public DuNativeAd getCacheAd()
```

同步获取广告对象数据。可以循环拿取，一直到广告缓存为0。

在使用该接口展示广告时，请进行缓存非空判断，避免缓存池为空导致空指针。

建议在使用 `get()` 后再次调用 `fill()` 接口进行广告缓存。

```
public boolean isHasCached()
```

获取当前是否有广告缓存，有缓存则返回 True。

Java代码示例：

```
DuNativeAd nativeAd = new DuNativeAd(this, PID, CACHESZIE);

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) {
        Log.d(TAG, "onClick : click ad");
    }

    @Override
    public void onAdLoaded (final DuNativeAd ad) {
        Log.d(TAG, "onAdLoaded : " + ad.getTitle());
    }
}
```

```
}  
};
```

Kotlin代码示例：

```
lateinit var nativeAd: DuNativeAd  
nativeAd = DuNativeAd(this, PID, DEFAULT_CACHE_SIZE);  
nativeAd.setMobulaAdListener(mListener)  
nativeAd.load()  
  
var mListener = object : DuAdListener {  
    override fun onClick(p0: DuNativeAd?) {  
        Log.d(TAG, "onClick")  
    }  
  
    override fun onError(p0: DuNativeAd?, p1: AdError?) {  
        Log.d(TAG, "onError")  
    }  
  
    override fun onAdLoaded(ad: DuNativeAd?) {  
        Log.d(TAG, "onAdLoaded")  
    }  
}
```

7.4 销毁原生广告对象

在退出原生广告展示界面时，建议销毁原生广告对象。

接口说明：

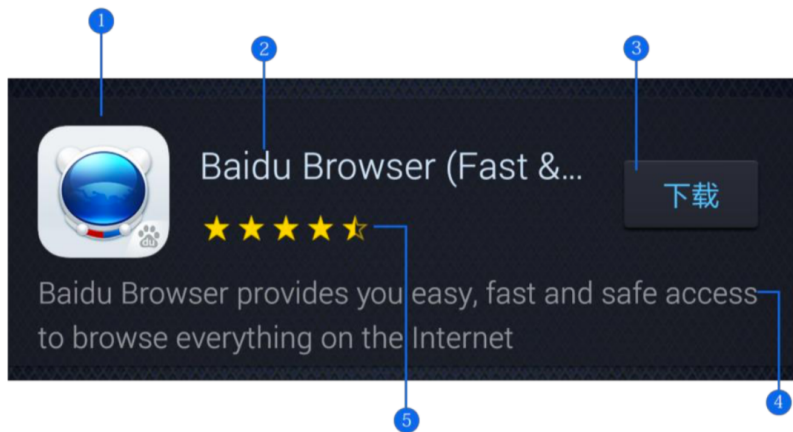
```
public void destroy()
```

8. 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。

8.1 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。



1. 图标 2. 标题 3. CTA 按钮 4. 宣传文案 5. 评价

8.2 数据获取接口

- 图标获取接口

```
public String getIconUrl()
```

返回广告图标的 Url 地址。

- 标题获取接口

```
public String getTitle()
```

返回标题文案。广告中必须包含一个标题。请保留至少 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。

- CTA 按钮获取接口

```
public String getCallToAction()
```

返回 CTA 按钮文案。广告中必须包含一个触发按钮。

请不要缩短或改变按钮文案。按钮文案的最大字符长度个数：25。

- 宣传文案获取接口

```
public String getShortDesc()
```

返回广告的宣传文案。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。

- 评级获取接口

```
public float getRatings()
```

返回该广告应用在 Google Play 上的评级。

- 宣传图获取接口

```
public String getImageUrl()
```

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

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。

9. 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，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。

10. 原生广告 List 使用

建议需要同时展示多条原生广告时使用该方法。

10.1 构造原生广告 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	缓存广告个数

10.2 构造子原生广告类

```
public NativeAd()
```

10.3 注册原生广告 List 监听接口

接口说明：

```
public void setListener(AdListArrivalListener adListener)
```

参数	说明
AdListArrivalListener adListener	回调函数返回获取广告错误，获取广告的数据。

```
public interface AdListArrivalListener {  
    public void onError(AdError error);  
    public void onAdLoaded(List<NativeAd> mNativeAd);  
}
```

使用获取数据方法后，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。

- 获取广告成功
DU Ad Platform_SDK 会回调 `onAdLoaded()` 方法，通过 `List<NativeAd>` 的对象开发者可以得到每个广告对象，并分别获得对应的广告元素
- 获取广告失败
DU Ad Platform_SDK 会回调 `onError()` 方法，通过 `AdError` 对象开发者可以得到具体错误信息。

10.4 注册原生广告子类监听接口

接口说明：

```
public void setMobulaAdListener(DuAdDataCallBack mCallBack)
```

参数	说明
DuAdDataCallBack mCallBack	此接口可以获得单个广告点击事件， <code>onAdLoaded()</code> ， <code>onAdError()</code> 已经由 <code>AdListArrivalListener</code> 回调，在该接口无返回。

```
public interface DuAdDataCallBack {
    public void onError(AdError error);
    public void onAdLoaded(NativeAd mNativeAd);
    public void onAdClick();
}
```

- 获取广告点击事件

DU Ad Platform_SDK 会回调 `onClick()` 方法，通知开发者该 NativeAd 的对象的广告被点击。

Java代码示例：

```
private NativeAd mNativeAD;
private LinkedList<NativeAd> lists = new LinkedList<NativeAd>();
private DuNativeAdsManager adsManager == new
DuNativeAdsManager(getApplicationContext(), PID, CACHESZIE);

@Override
protected void onResume() {
    super.onResume();
    if (adsManager != null) {
        adsManager.setListener(listener);
        adsManager.load();

        mNativeAD = lists.get(mPositon);
        mNativeAD.setMobulaAdListener(callback);
        mNativeAD.registerViewForInteraction(btnView);
    }
}

AdListArrivalListener listener = new AdListArrivalListener() {
    NativeAd nativeAD;
    //返回广告 list
    @Override
    public void onAdLoaded(List arg0) {
        for (int i = 0; i < arg0.size(); i++) {
            //获取单个广告对象
            nativeAD = (NativeAd) arg0.get(i);
            if (!(nativeAD.equals(null))) {
                lists.add(nativeAD);
            }
        }
    }
}
```

```

}

//返回广告错误码
@Override
public void onAdError(AdError arg0) {
    Log.d(TAG, "onError : " + arg0.getErrorCode());
}
};

DuAdDataCallBack callback = new DuAdDataCallBack() {
    @Override
    public void onAdLoaded(NativeAd data) {
    }

    @Override
    public void onAdError(AdError error) {
    }

    @Override
    public void onAdClick() {
        Log.d(TAG, "onClick : click list ad");
    }
};

```

Kotlin代码示例:

```

lateinit var mDuNativeAdsManager: DuNativeAdsManager
val mAdList = arrayListOf<NativeAd>()
lateinit var mNativeAD: NativeAd

override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    mDuNativeAdsManager = DuNativeAdsManager(this, PID, CACHESIZE).apply {
        setListener((object : AdListArrivalListener {
            override fun onAdLoaded(p0: MutableList<NativeAd>?) {
                Log.d(TAG, "onAdLoaded")
                mAdList?.clear()
                p0?.forEach {
                    if (it != null) mAdList.add(it)
                }
                if (mAdList.size == CACHESIZE) {
                    mHandler.apply {
                        removeCallbacksAndMessages(null)
                        post(mRunnable)
                    }
                }
            }
        })
    }

    override fun onAdError(p0: AdError?) {
        Log.d(TAG, "onError : " + p0?.getErrorCode());
    }
}

```

```

    }

    )))
}

Log.d(TAG, "load list ad....")
mDuNativeAdsManager.load()
}

private val mRunnable = object : Runnable {
    override fun run() {
        if (mPositon < mAdList.size) {
            mNativeAD = mAdList.get(mPositon);
            mNativeAD.setMobulaAdListener(callback);
            val url = mNativeAD.adCoverImageUrl
            if (url.isEmpty()) showSmallAdView(mNativeAD) else
showBigAdView(mNativeAD)

            mPositon++
        } else {
            mPositon = 0
        }
        mHandler.postDelayed(this, 8000)
    }
}

var callback = (object : DuAdDataCallBack {
    override fun onAdClick() {
        Log.d(TAG, "onClick : click list ad");
    }

    override fun onAdLoaded(p0: NativeAd?) {
        Log.d(TAG, "onAdLoaded: adload list ad")
    }

    override fun onAdError(p0: AdError?) {
        Log.d(TAG, "onAdError: onAdError list ad:" + p0.toString())
    }
})

```

10.5 获取广告数据接口

接口说明：

```
public void fill()
```

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该方法。

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。

```
public void load()
```

异步获取广告对象数据，没有缓存时会进行广告请求

10.6 原生广告数据获取

- 图标获取接口

```
public String getAdIconUrl()
```

返回广告图标的 Url 地址。

- 标题获取接口

```
public String getAdTitle()
```

返回标题文案。广告中必须包含一个标题。请保留至少 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。

- CTA 按钮获取接口

```
public String getAdCallToAction()
```

返回 CTA 按钮文案。广告中必须包含一个触发按钮。

请不要缩短或改变按钮文案。按钮文案的最大字符长度个数：25。

- 宣传文案获取接口

```
public String getAdBody()
```

返回广告的宣传文案。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。

- 评级获取接口

```
public float getAdStarRating()
```

返回该广告应用在 Google Play 上的评级。

- 宣传图获取接口

```
public String getAdCoverImageUrl()
```

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

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。

10.7 销毁原生广告 List 对象

在退出原生广告展示界面时，建议销毁原生广告 List 对象。

接口说明：

```
public void destroy()
```

Java代码示例：

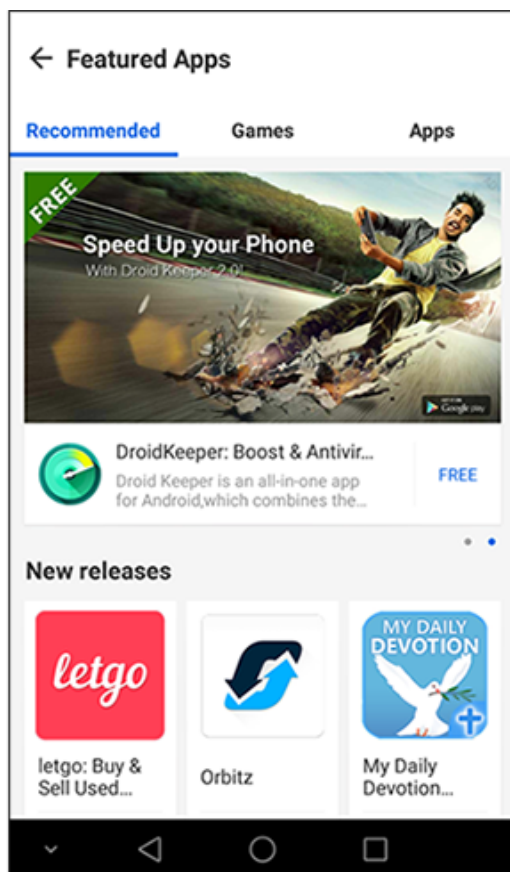
```
@Override
protected void onDestroy() {
    super.onDestroy();
    adsManager.setListener(null);
    adsManager.destroy();
}
```

Kotlin代码示例：

```
override fun onDestroy() {
    super.onDestroy()
    mDuNativeAdsManager.apply {
        setListener(null)
        destroy()
    }
}
```

11. 广告墙使用

广告墙是一个被封装的列表广告。



11.1 注册广告墙 Activity

在安卓工程目录下，打开 AndroidManifest.xml，添加以下 Activity：

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

11.2 设置相关参数

广告墙支持设置的参数如下：

OfferWallAct.KEY_PID

DAP 平台申请的广告位 id，该 pid 注册在 Json 的 Offerwall 数组中。在 DAP 后台创建该广告位时，广告形式必须选择「广告墙」，否则将无法获取广告。

该属性必须填写

OfferWallAct.KEY_TITLE_ID

标题文案设置，支持传入资源 ID。默认值为 "Featured Apps" 及对应语言文本

OfferWallAct.KEY_TAB_BACKGROUND_COLOR

设置广告墙 Tab 背景色，形如#FFFFFF。默认值为#EDEDED

OfferWallAct.KEY_TAB_INDICATOR_COLOR

设置 Tab 选择指示器的颜色及被选中的 Tab 的文字颜色；形如#FFFFFF。默认值为#1C86EE

OfferWallAct.KEY_TAB_TEXT_COLOR

设置标题及未被选中的 Tab 的文字颜色；形如#FFFFFF。默认值为#000000

Java代码示例：

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

Bundle b = new Bundle();
b.putInt("pid", YOUR_PID);
b.putInt(OfferWallAct.KEY_TITLE_ID, R.string.app_name); // 可选
b.putString(OfferWallAct.KEY_TAB_BACKGROUND_COLOR, "#EDEDED"); //可选
b.putString(OfferWallAct.KEY_TAB_INDICATOR_COLOR, "#1C86EE"); //可选
b.putString(OfferWallAct.KEY_TAB_TEXT_COLOR, "#000000"); //可选

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

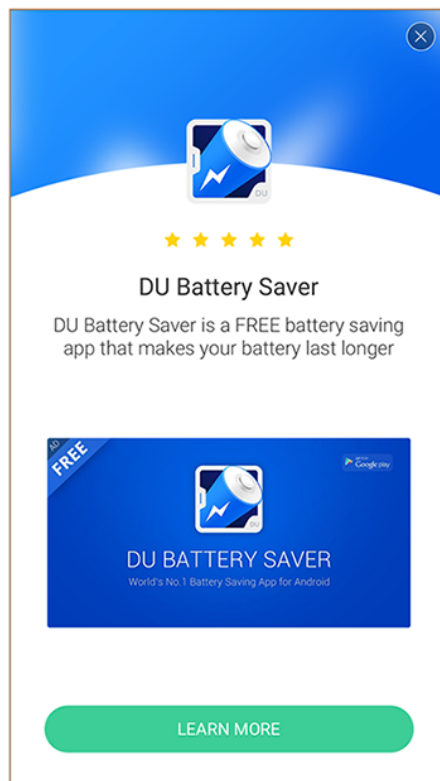
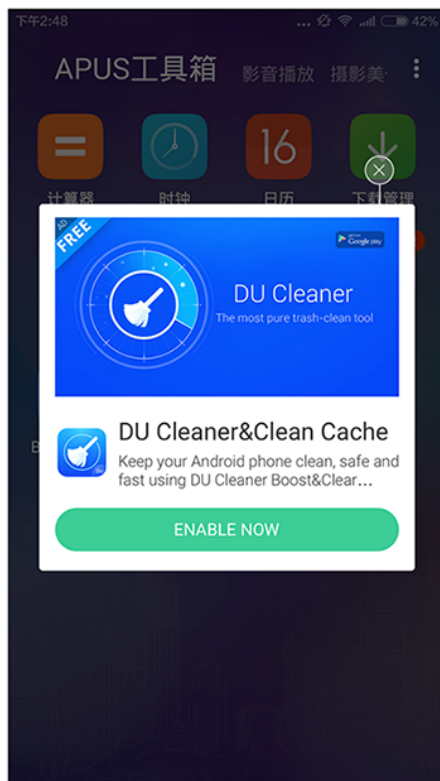
kotlin代码示例：

```
var offerwallIntent = Intent(this, offerWallAct::class.java)
var bundle = Bundle()
bundle.apply {
    putInt("pid", 61709);
    putInt(OfferWallAct.KEY_TITLE_ID, R.string.app_name); // 可选
    putString(OfferWallAct.KEY_TAB_BACKGROUND_COLOR, "#EDEDED"); // 可 选
    putString(OfferWallAct.KEY_TAB_INDICATOR_COLOR, "#1C86EE"); // 可选
    putString(OfferWallAct.KEY_TAB_TEXT_COLOR, "#000000"); // 可选
}

offerwallIntent.putExtras(bundle)
startActivity(offerWallIntent)
```

12. 插屏广告使用

插屏广告样式：



12.1 构造插屏广告对象

接口说明：

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

参数	说明
Context context	ACTIVITY CONTEXT
int pid	广告位 ID，该 pid 注册在 Json 的 native 数组中
int type	<code>InterstitialAd.Type.SCREEN</code> 全屏广告 <code>InterstitialAd.Type.NORMAL</code> 半屏广告 此参数缺省时，默认为半屏

12.2 注册插屏广告回调

接口说明：

```
public void setInterstitialListener (AbsInterstitialListener adListener)
```

参数	说明
<code>AbsInterstitialListener</code> <code>adListener</code>	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 AbsInterstitialListener {  
    //广告接收失败回调  
    public void onAdFail(int errorCode);  
  
    //广告接收成功回调  
    public void onAdReceive();  
  
    //广告销毁回调  
    public void onAdDismissed();  
  
    //广告展示回调  
    public void onAdPresent();  
  
    //广告点击回调  
    public void onAdClicked();  
}
```

12.3 使用插屏广告

接口说明：

```
public void fill()
```

调用 `fill()` 接口可以提前缓存广告，在 `load()` 广告时可以更快获取。建议在广告展示的前置场景调用该方法。

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。

```
public void load()
```

获取广告对象数据，没有缓存时会进行广告请求。

建议在使用 `load()` 后再次调用 `fill()` 接口进行广告缓存。

```
public void show()
```

广告展示方法，请在回调 `onAdReceive()` 中使用本方法。

```
public void close()
```

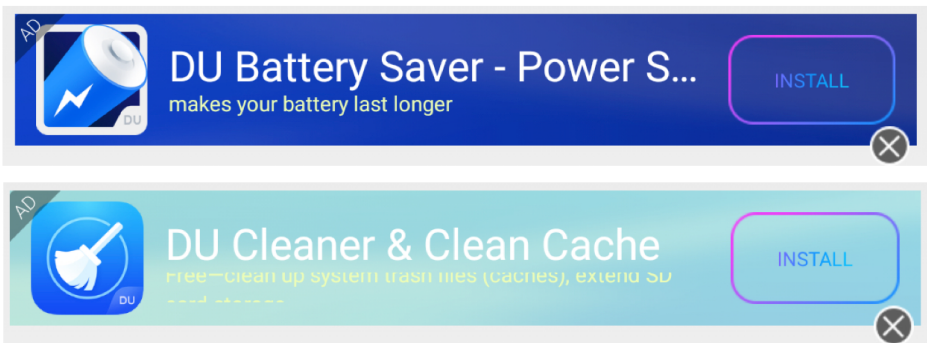
广告关闭方法，**该方法已废弃**。

```
public void destroy()
```

广告对象销毁方法，在退出插屏广告展示界面时建议使用。

13. 横幅广告使用

横幅广告样式：



横幅广告尺寸：320 x 50 px

13.1 构造横幅广告对象

接口说明：

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

参数	说明
Context context	ACTIVITY CONTEXT
int pid	广告位 ID，该 pid 注册在 json 的 native 数组中
int cachesize	广告缓存个数
BannerType	展示类广告使用TYPE_CPM
BannerListener listener	横幅广告监听器（同一个 BannerAdView 只回调首次 load() 的结果）

13.2 注册横幅广告回调

接口说明：

```
public BannerListener ()
```

```

public interface BannerListener {
    //广告接收失败回调
    public void onError(String msg);

    //广告接收成功回调
    public void onAdLoaded();
}

```

Java代码示例:

```

BannerAdView mBannerAdView = new BannerAdView(this, PID, CACHESIZE, new
BannerListener() {
    @Override
    public void onAdLoaded() {
        Log.d(TAG, "onAdLoaded");
    }
    @Override
    public void onError(String msg) {
        Log.d(TAG, "onError:" + msg);
    }
});

```

Kotlin代码示例:

```

mBannerAdView = BannerAdView(this, PID, 5, (object : BannerListener {
    override fun onError(p0: String?) {
        Log.d(TAG, "onError")
    }

    override fun onAdLoaded() {
        Log.d(TAG, "onAdLoaded")
    }
}))

```

13.3 设置相关参数

```
public void setBgStyle(int BannerStyle)
```

参数	说明
Int BannerStyle	设置背景颜色: BannerStyle.STYLE_BLUE:蓝色背景 BannerStyle.STYLE_GREEN:绿色背景;

```
public void setCloseStyle(int BannerCloseStyle)
```

参数	说明
Int BannerCloseStyle	设置按钮位置： BannerCloseStyle.STYLE_BOTTOM:右下角 BannerCloseStyle.STYLE_TOP:右上角

13.4 添加横幅到布局中

```
YourLayout.addView(mBannerAdView);
```

13.5 使用横幅广告

```
public void load()
```

获取广告对象数据，没有缓存时会进行广告请求。

```
public void destroy()
```

广告对象销毁方法，在退出横幅广告展示界面时建议使用

14. 常见问题

14.1 SDK接入

- Q：最新版SDK、接入文档下载地址
A： http://ad.duapps.com/zh_CN/sdk/
- Q：SDK下载页提供基础包和扩展包，我应该选择哪个？
A：请根据您所需的广告形式自由选择，其中基础包为必选包，扩展包为可选包
 - eg1：我只需要原生广告，请选择基础包HW 1.x.x；
 - eg2：我需要原生、插屏、视频广告，请选择基础包CW 1.x.x+扩展包Video SDK 1.x.x.x；
- Q：json可否重复初始化？
A：可以多次初始化，以最后一次为准，请确保最后一次初始化时，已传入全部正确的pid；若已接入视频SDK，在初始化DuVideoSDK.init()时，也要保证已传入全部正确的pid。
- Q：返回错误之后是否会自动重试？

A: 不会自动重试, 获取广告失败后请根据错误码再次发起请求, 注意不要在回调onError()中重试, 否则可能导致死循环。

- Q: 原生广告点击率低

A: 点击属于用户行为, 建议增大原生广告的可点击区域, 并确认已经注册广告可点击区域视图的监听; 同时, 广告位设计的样式也会影响用户的点击, 如需帮助请携广告截图联系我们。

- Q: 原生广告怎么没有回调?

A: 请确认原生广告对象与监听是否一一对应, 销毁原生广告对象后, 重新构造的广告对象时, 需要对新的对象注册广告数据监听; 同一个广告版位, 不同的广告对象, 需要重新注册对应的监听。

14.2 平台

- Q: 为什么应用在平台的状态是“待激活”?

A: 已经通过平台审核, 可接收测试广告, 有广告展示后, 系统会下发正式广告, 同时状态变为“发布中”。

- Q: 平台数据是实时的吗?

A: 不是, 平台数据有延迟, 当天可查看前一天的数据, 以北京时间0:00-23:59为一天计算。

- Q: 为什么平台上的数据跟我统计的不一样?

A: 获取广告load()时, 如果有缓存, 默认不会再次发起请求, 因此平台上的请求数可能比您统计到的load()次数偏低, 而且展示数可能比填充数还要高; 不同平台的统计规则略有不同, DAP的广告数据请以DAP平台的为准。

- Q: 如何对广告进行过滤?

A: 在平台填写【过滤设置】, 可选择按照应用包名/广告类别/受众年龄进行过滤。

14.3 广告

- Q: 获取广告时, 返回错误码1001

A: 请核对包名、app license和pid三者是否正确匹配, 配置方法详见4.加载与配置

- Q: 获取广告时, 返回错误码3000

A: 请确认测试机是否已连接全局vpn, 确认方法: 使用测试设备查询IP归属地, 非中国大陆IP可拉取广告。

- Q: 广告点击后无法跳转

A: 请确认android设备已安装Google Play。

14.4 其他

- Q: 为什么我的APP有展示, 但没有收入/收入过低?

A: DAP目前是效果类的广告, 展示后需要用户有点击、安装、激活等行为才会带来收入; 对于新接入的应用, 建议提高dau加快优化进度; 放量后系统需要一段时间优化; 如对收入仍有疑问, 请及时联系我们。

- Q: 是否支持聚合?

A: DAP不支持聚合, 如需更多的第三方广告, 推荐使用Admob作为mediation, 并在后台勾选Du Ad Platform。