DU Ad Platform_SDK for Unity 接入手册

Version: DUAd_SDK_Unity 1.1.0

前提: DU Unity SDK 目前支持 Untiy 5.0b19 及以上版本

Android 目前支持插屏广告，横幅广告，应用墙广告和视频广告。

iOS 目前支持插屏广告。

---

1. 概述

本文档描述如何在应用中接入来自百度开发者平台的 DU Ad Platform_SDK 产品。

百度开发者平台可以为应用提供广告服务。DU Ad Platform_SDK 是百度开发者平台中用来提供广告的一款产品。

1.1 读者对象

本文档面向的读者是 Unity 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、iOS8（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 接入过程中需要的两个身份：APP_ID，广告位 ID。

3.1 APP_ID

1. 定义

   APP_ID 是开发者的应用在广告平台的唯一标识。

2. 获取方式

   访问百度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。

2. 获取方式

   访问百度开发者平台进行申请。

3. 代码

   pid

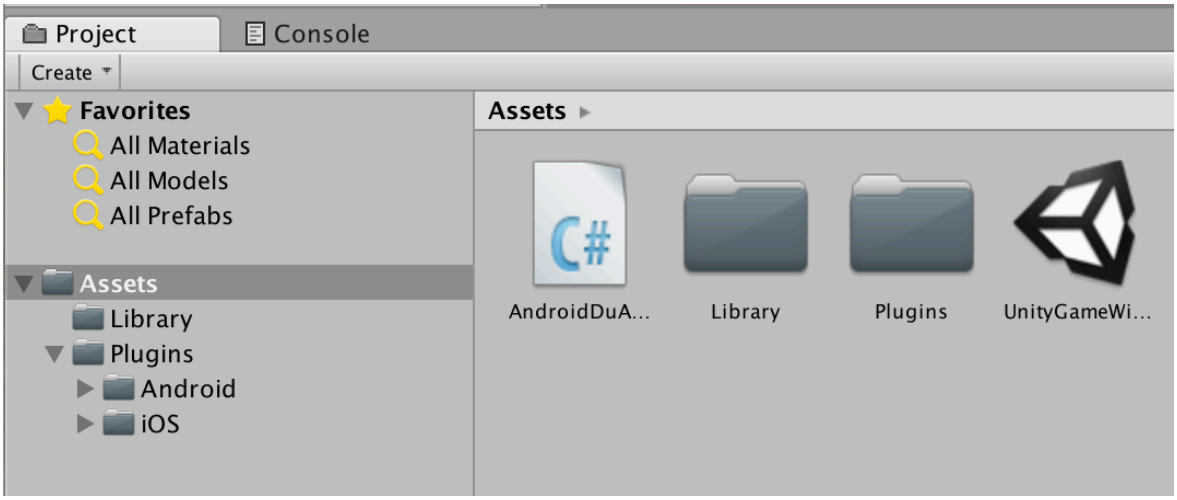
4. 加载与配置

本章描述在应用中如何加载 DU Ad Platform_SDK 的压缩包。

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. 将 DAPUnity.unitypackage 加入 Unity3D 项目中



4.2 配置 AndroidManifest.xml 和代码混淆（仅限Android）

以下步骤仅针对 Android 应用，iOS 应用请跳过该部分。

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: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="package-name.DuAdCacheProvider"
          android:exported="false">
    </provider>
</application>
```

注： “package-name” 为开发者 APP 的包名全称。

3. 注册 APP 安装广播监听

   请正确添加该监听，否则会影响您的变现效率。

```
<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. 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:

   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>;}
-keep class com.duapps.ad.banner.BannerListener { *; }
```

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

5. 初始化

在完成 DU Ad Platform_SDK 接入操作之前，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 String 字符串，将 Placement_ID 与广告位 ID 建立对应关系。具体格式如下（JSON格式）：

```
string DAPJson = "{
  \"native\":
    [
      {
        \"pid\":\"Your_id_for_interstial_ad\"
      },
      {
        \"pid\":\"Your_id_for_banner_ad\"
      }
    ],
  \"offerwall\":
    [
      {
        \"pid\":\"Your_id_for_offerwall\"
      }
    ],
  \"video\":
    [
      {
        \"pid\":\"Your_id_for_video_ad\"
      }
    ]
}";
```

2. Android 应用请在`Start()`方法中使用`DuAdNetworkForUnity.Init()`和`DuVideoAdSDKForUnity.Init()`方法完成初始化。

   接口说明：

   > public static void Init(String pidsJson)

参数	说明
String pidsJson	Placement_ID与广告位ID的对应关系

   代码示例：

```
void Start() {
    //初始化SDK
    DuAdNetworkForUnity.Init(DAPJson);
    DuVideoAdSDKForUnity.Init(DAPJson);
}
```

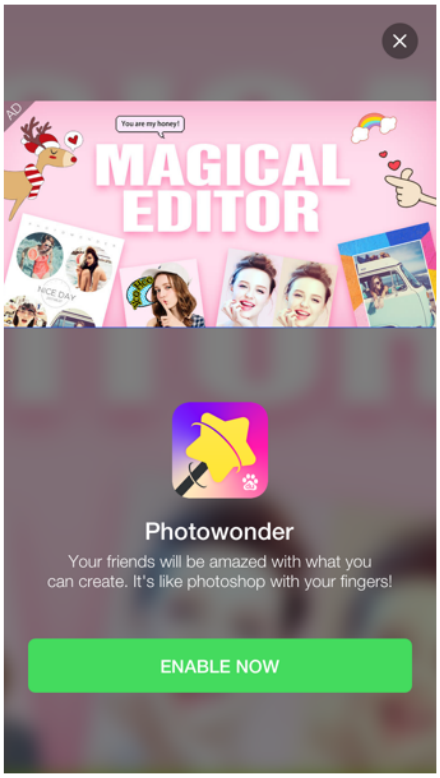
3. iOS 应用请在`Start()`方法中使用`DuAdNetworkForUnity.LicenseConfig()`，`DuAdNetworkForUnity.Init()`方法完成初始化。

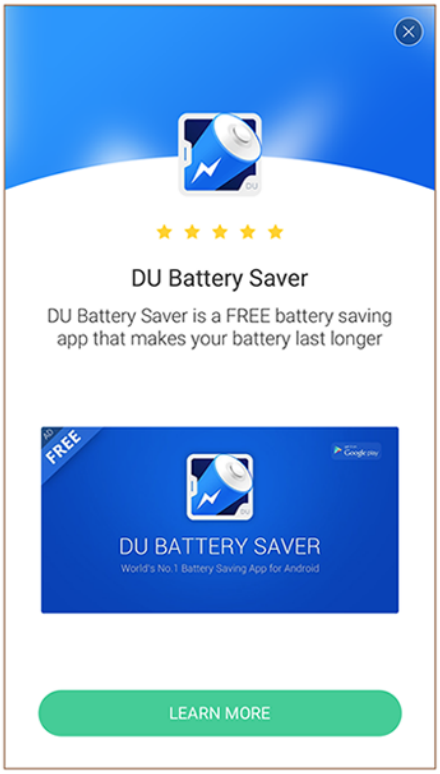
   代码示例：

```
void Start() {
    //初始化SDK
    //在 app_license 填入已申请的 APP_ID
    DuAdNetworkForUnity.LicenseConfig(Your_app_license);
    DuAdNetworkForUnity.Init(DAPJson);
}
```

6. 插屏广告使用

插屏广告样式：iOS 样式（左） Android 样式（右）





6.1 构造插屏广告对象

接口说明：

> public InterstitialAd(int pid)

参数	说明
int pid	广告位ID，该 pid 注册在 Json 的 native 数组中

6.2 注册插屏广告回调

接口说明：

```
interstitialAd.InterstitialAdReceive = delegate() {
  //广告获取成功回调
  Debug.Log ("InterstitialAdReceive");
  interstitialAd.ShowAd ();
};

interstitialAd.InterstitialAdPresent = delegate() {
  //广告展示回调
  Debug.Log ("InterstitialAdPresent");
};

interstitialAd.InterstitialAdClicked = delegate() {
  //广告点击回调
  Debug.Log ("InterstitialAdClicked");
};

interstitialAd.InterstitialAdDismissed = delegate() {
  //广告被用户关闭回调
  Debug.Log ("InterstitialAdDismissed");
};

interstitialAd.InterstitialAdError = delegate(int errorCode) {
  //广告错误回调
  Debug.Log ("InterstitialAdError : " + errorCode);
};
```

ErrorCode 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
UNKNOW_ERROR_CODE	3001	未知错误

6.3 使用插屏广告

接口说明：

> public void FillAd()

调用`FillAd()`接口可以提前缓存广告，在`LoadAd`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 LoadAd()

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`LoadAd`后再次调用`FillAd()`接口进行广告缓存。

> public void ShowAd()

广告展示方法，请在回调`InterstitialAdReceive()`中使用本方法

> public void Dispose()

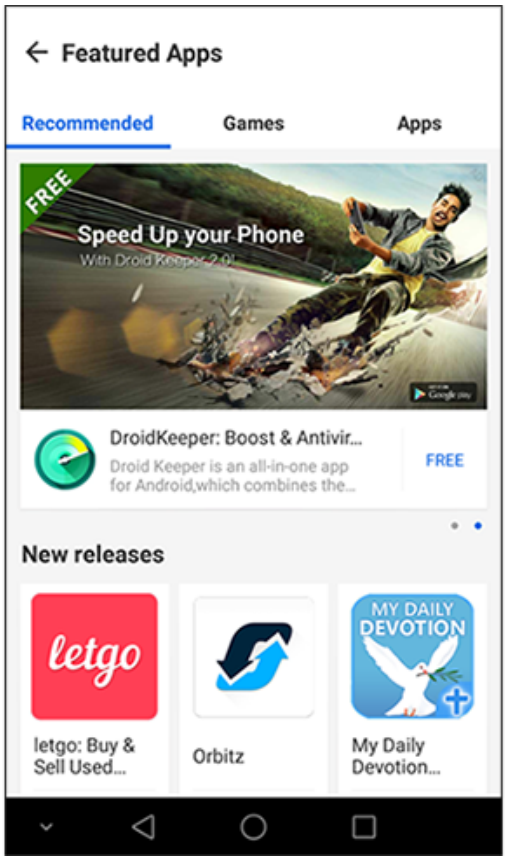
广告对象销毁方法，在退出插屏广告展示界面时建议使用

代码示例：

```
void OnApplicationQuit ()
{
  Debug.Log ("OnApplicationQuit");
  If(interstitialAd != null)
  {
    interstitialAd.Dispose ();
  }
}
```

7. 广告墙使用（仅限 Android）

广告墙是一个被封装的列表广告。



7.1 构造广告墙对象

接口说明：

> public OfferwallAd(int pid)

参数	说明
int pid	广告位ID，该 pid 注册在 Json 的 offerwall 数组中

7.2 广告展示接口

接口说明：

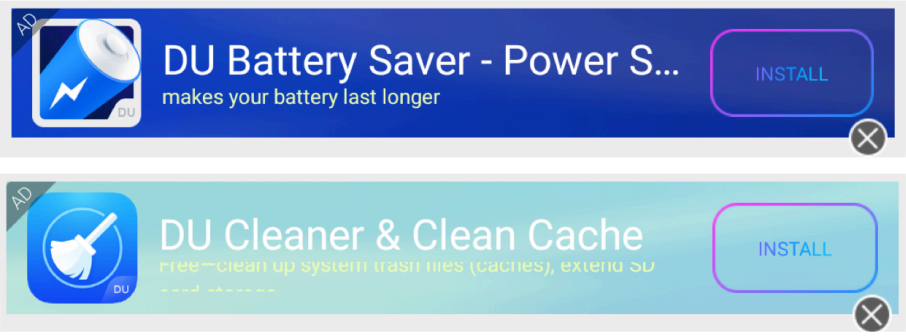
> public void Show()

代码示例：

```
OfferwallAd  offerwallAd = new OfferwallAd (YOUR_DAP_PLACEMENT_ID);
offerwallAd.Show();
```

8. 横幅广告使用（仅限 Android）

横幅广告样式：



横幅广告尺寸：320 x 50 px

8.1 构造横幅广告对象

接口说明：

> public BannerAd(int pid, int cachesize, int bgStyle, int closeStyle, int positionStyle)

参数	说明
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
int bgSize	设置横幅广告背景颜色 `BannerStyle.STYLE_BLUE`: 蓝色背景 `BannerStyle.STYLE_GREEN`: 绿色背景
int closeStyle	设置横幅广告关闭按钮的位置 `BannerCloseStyle.STYLE_BOTTOM`: 右下角 `BannerCloseStyle.STYLE_TOP`: 右上角
int positionStyle	设置横幅广告的位置 `BannerAd.POSITION_TYPE_TOP`: 页面顶部 `BannerAd.POSITION_TYPE_BOTTOM`: 页面底部

8.2 注册横幅广告回调

接口说明：

```
//banner广告各个回调
bannerAd.BannerAdLoaded = delegate() {
  //广告加载成功
  Debug.Log ("BannerAdLoaded");
};

bannerAd.BannerAdError = delegate(string errorMessage) {
  //广告错误回调
  Debug.Log ("BannerAdError : " + errorMessage);
};
```

8.3 使用横幅广告

> public void LoadAd()

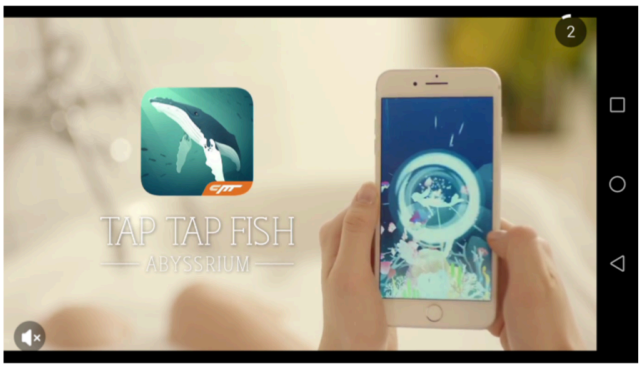
获取广告对象数据，没有缓存时会进行广告请求。

> public void Dispose()

广告对象销毁方法，在退出横幅广告展示界面时建议使用

9. 视频广告使用（仅限 Android）

视频广告示例（播放页与播放完成页）：



9.1 构造视频广告对象

接口说明：

> public VideoAd(int pid)

参数	说明
int pid	广告位ID，该 pid 注册在 Json 的 video 数组中

9.2 注册视频广告回调

接口说明：

```
videoAd.VideoAdEnd = delegate(bool isSuccessfulView, bool
isCallToActionClicked) {
  //视频广告完全结束时回调
  // bool isCallToActionClicked  返回用户是否点击了CallToAction按钮
```

```
  Debug.Log ("isSuccessfulView : " + isSuccessfulView + ",
isCallToActionClicked : " + isCallToActionClicked);
};

videoAd.VideoAdError = delegate(string errorMessage) {
  //广告错误回调
  Debug.Log ("errorMessage : " + errorMessage);
};

videoAd.VideoAdPlayable = delegate() {
  //广告已经准备好，可以调用PlayVideo()方法
  Debug.Log ("VideoAdPlayable");
};

videoAd.VideoAdStart = delegate() {
  //广告开始播放时回调
  Debug.Log ("VideoAdStart");
};
```

9.3 使用视频广告

> public void LoadAd()

此接口只需调用一次，视频广告会在后台线程持续拉取，拉取到广告后会通过回调通知。

> public boolean IsAdPlayable();

返回当前是否有可以播放的广告，有返回true，没有则返回false

> public void PlayAdVideo()

播放广告，视频广告方向将根据设备的屏幕方向自动旋转

> public void Dispose()

广告对象销毁方法，在退出视频广告展示界面时建议使用