Du Ad SDK for Cocos
Access Guide

Du Cocos SDK 1.0

Content

1 Obtain Identity

This section describes the two IDs needed during **DU Ad Platform_SDK** integration: **APP ID** and **DAP Placement ID**.

1.1 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

1.2 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

2 Load DAP SDK

1. Copy all the files under directory **DuAd_Cocos_SDK_1.0/Classes** to your project, under directory **Classes**
2. Copy all the files under directory **DuAd_Cocos_SDK_1.0/cpp** to your project, under directory **app/src/org/cocos2dx/cpp**
3. Copy the **aar** and **jar** packages to your project, under directory **app/libs**
4. Change the **build.gradle** of your android project

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

```
dependencies {
        compile fileTree(include: ['*.jar'], dir: 'libs')
        compile(name: 'DuappsAd-CW-xxx-release', ext: 'aar')
        compile(name: 'DuVideoSdk-xxx-release', ext: 'aar')
       compile(name: 'support-v4-23.1.0', ext: 'aar')
        compile 'com.android.support:palette-v7:23.4.0'
}
```

Note:

1. **DuappsAd-CW-vXX-release.aar** is the SDK for native and interstitial ads and **DuVideoSdk-vXX-release.aar** is the SDK for video ads. You can replace it with the latest version from our website.
2. **'com.android.support:palette-v7:23.4.0'** is dedicated to video SDK. You can change the version depending on your project. Version above 21 is suggested. If compilation error happens, change the code to:

```
compile 'com.android.support:palette-v7:23.4.0' {
         transitive = false
         }
```

3 Configure AndroidManifest.xml

Please modify the AndroidManifest.xml as follows:

1. Add permissions. The minimum permissions required by **DU Ad Platform_SDK** are:

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

2. Add the **APP_ID** you applied under value of **app_license**.

```
     <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 name of the developer’s app.

3. Register correctly the **PACKAGE_ADDED** Receiver. Otherwise, it might affect your monetization efficiency.

```
      <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 Obfuscate Code

Please follow the below rules to obfuscate code. Otherwise, there might be exceptions at run time.

1. Exclude classes of **DU Ad Platform SDK** when obfuscating;

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

2. 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>;}
```

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

5 Initialization- Configure Json

Before integration, Android application must initialize the **DU Ad Platform_SDK**. **The placement ID can't retrieve any load when it's not initialized.** You will need to configure the **dxtoolbox.json** under ***Assets/Plugins/Android/assets/dxtoolbox*** directory as follows:

- For **interstitial ads**, please update PID (DAP PlacementID) under **"Native"**
- For **video ads**, please update the PID under **"video"**

```
{
    "native": [
        {
            "pid": "YOUR_DAP_PLACEMENT_ID(for interstitial ads)"
        },
        {
            "pid": "YOUR_DAP_PLACEMENT_ID"
        }
    ],
    "video": [
      {
      "pid": "YOUR_DAP_PLACEMENT_ID(for video)"
       }
    ]
}
```

Please use the initialization method in **OnCreate** method of **AppActivity**.

//Initialize SDK

```
DuAdNetwork.init(Cocos2dxActivity.getContext(), getConfigJSON(this));
DuAdNetwork.setLaunchChannel("cocos2dx");
```

//Initialize video SDK (Skip this step if you don't video ads)

```
DuVideoAdSDK.init(Cocos2dxActivity.getContext(), getConfigJSON(this));
```

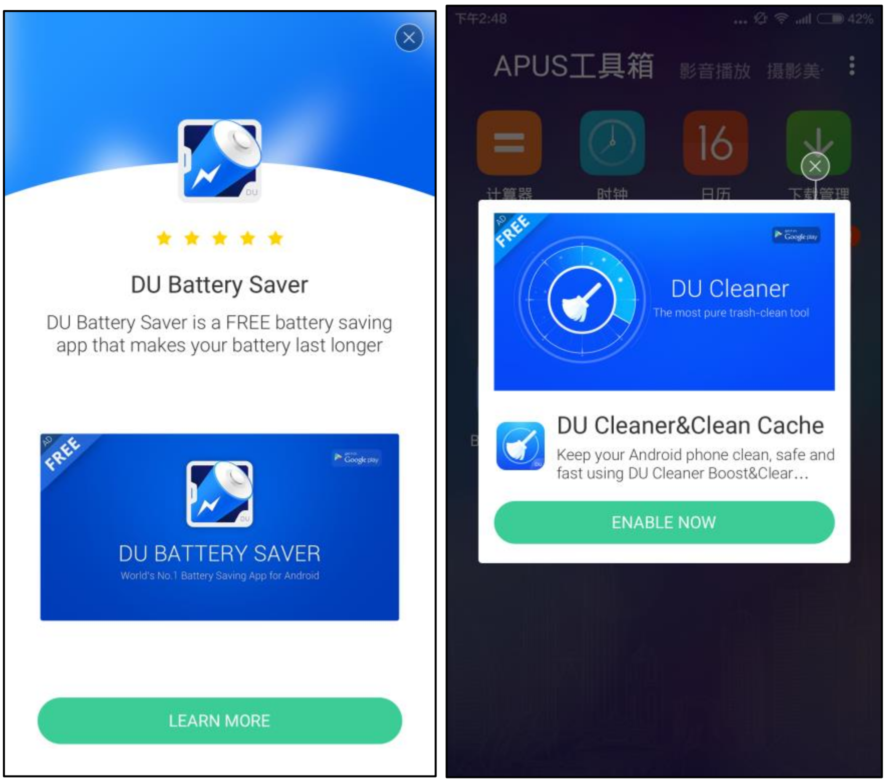
Sample codes:

```
private static String TOOLBOX_AD_CONFIG = "dxtoolbox/dxtoolbox.json";

private String getConfigJSON(Context context) {
    BufferedInputStream bis = null;
    ByteArrayOutputStream bos = new ByteArrayOutputStream();
    try {
        bis = new BufferedInputStream(context.getAssets().open(TOOLBOX_AD_CONFIG));
         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();
}
```

6 Request DAP Interstitial Ad

A sample of full screen interstitial ad and half screen interstitial ad



DU Battery Saver is a FREE battery saving app that makes your battery last longer

Keep your Android phone clean, safe and fast using DU Cleaner Boost&Clear…

6.1 Construction

- **Interface Instruction:**
  public DAPInterstitialAd(int placementID, int type);

Parameters	Description
int type	Interstitial type: *DAPInterstitialAd::TYPE_NORMAL*: for half screen interstitial *DAPInterstitialAd:: TYPE_FULL_SCREEN*: for full screen

	interstitial **Note**: Half size as default when type is missing
int pid	Your DAP placement ID for interstitial, register under **“native”**

6.2 Set callback for interstitial ad

Please register the callback for reception, and then retrieve the ad. The retrieval result, whether successful or not, click event is returned through a callback. This is an unsynchronized thread. So it won’t block the developer’s thread.

- **Interface Instruction:**

  public void onAdReceive(InterstitialAdBridgeCallback adDidLoad);

  Ad retrieval success callback, the developer can display the ad.

  public void onAdPresent(InterstitialAdBridgeCallback adDidShow);

  Callback of the impression, it will notify the developer that the ad is been displayed.

  public void onAdClicked(InterstitialAdBridgeCallback adDidClick);

  Callback of the click, it will notify the developer that the ad is been clicked.

  public void onAdDismissed(InterstitialAdBridgeCallback adDidClose);

  Ad closure callback, it will notify the developer that the ad is been closed.

  public void onAdError(InterstitialAdBridgeErrorCallback adFailWithError);

  Ad retrieval failure callback, the developer can get he error code. Please check chapter 6 for the meanings of error codes

6.3 Pre-load interstitial ad

The developer can choose when to use this function depending on the design of the app.

- Use the **fillAd()** to pre-cache ad in advance for faster loading the ad when using **loadAd().**
  Suggestion: Use the **loadAd()** at the page before the ad showing page.
  Please 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 fillAd();

6.4 Load interstitial ad

- **Interface Instruction:**

public void loadAd();

Note: Please set callback for interstitial ad before calling **loadAd().**

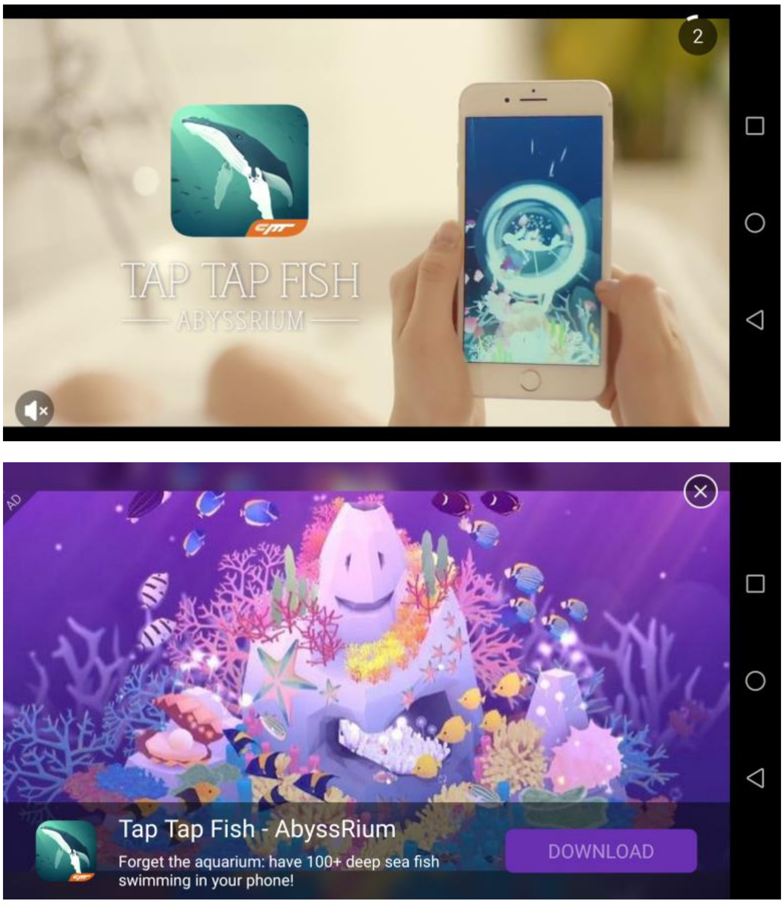
6.5 Show interstitial ad

- **Interface Instruction:**

  public void showAd();

7 Request DAP video ad

A sample of DU Video Ad （playing page and download page）



Tap Tap Fish - AbyssRium

Forget the aquarium: have 100+ deep sea fish swimming in your phone!

7.1 Construction

- **Interface Instruction:**

  public DAPVideoAd(int placementID);

7.2 Set callback for video ad

Please set the callback as follows, see chapter 6 for error codes.

- **Interface Instruction:**

  public void onAdPlayable(VideoAdBridgeCallback adPlayable);

  Callback when video ad is ready, then you can call playAd().

  public void onAdStart(VideoAdBridgeCallback adStart);

  Callback when ad starts to play.

  public void onAdEnd(VideoAdBridgeEndCallback adEnd);

  Callback when the video finishes.

  public void adEnd(bool isSuccessfulView, bool isCallToActionClicked)

  - *bool isSuccessfulView*

    Return whether the user watched the whole video or not.
  - *bool isCallToActionClicked*

    Return whether the user clicked CallToAction button or not.

  public void onAdError(VideoAdBridgeErrorCallback adFailWithError);

  Callback when ad retrieval failed.

7.3 Load Video ad

- **Interface Instruction:**

  public void loadAd();

You only need to call it once. The background thread will retrieve the video ad and notify the developer when it's done. Please setup the listener in advance.

Note: The time to download video ad is long, we suggest starting listener registration and loading right after the creation of the instance.

7.4 Check if there is playable video or not

- **Interface Instruction:**

  public bool isAdPlayable();

  Return **true** when there is playable video ad, **false** when there is not.

7.5 Play video ad

- **Interface Instruction:**

  public void playAd();

  Note: The rotation of screen is automatic.

8 Instruction of DuAD_SDK_DEMO

- Copy file **android/Android.mk** to your project under directory **app/ini**. This is a demo configuration file. Please modify this file to redirect it to your local path.

- Copy the files under **assets** to your project under directory **Resources**. This folder contains the configuration files and the materials of the demo application.
- Copy all files under directory **Classes** to directory **Classes** of your project. They are the source codes of ad impression, including interstitial, video and native ads. The developer can take it as an example in his own integration.
- Make sure the **package name** of the android project is **com.mobula.sample**. And modify **build.gradle** correctly.