DU Quick Charge SDK for Android Access Guide

DU QuickCharge v1.2

Baidu Online Network Technology (Beijing) Co., Ltd

No.	DUAd10120150810
Date	2016-05-06
Ver.	1.2
Email	support_duad@baidu.com

Contents

1 Introduction

This document describes how to access **DU Quick Charge SDK** for Android apps of Baidu Developers .
Baidu (http://ad.duapps.com) offers advertising services for Android apps. For example, **DU Quick Charge SDK** is a product that provides native ads and quick charge function.

1.1 Target Audience

This document is for Android app developers.

1.2 Prerequisites

DU Quick Charge 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, jar package, etc.**	**DuAD_QuickChargexxx.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 Quick Charge SDK** to Android app is described as below:

- The access workflow of DU Quick Charge LockScreen：
  1. Apply for Location_ID, App_ID and Placement_ID. *See Section 3.*
  2. Load DU Quick Charge_SDK work package; configure Androidmanifest.xml. *See Section 4.*
  3. DU Quick Charge_SDK initialization. *See Section 5.*
  4. Access DU Quick Charge LockScreen *See Section 6.*
  5. Done.

- The access workflow of DU Native Ad:
  1. Apply for Location_ID, App_ID and Placement_ID. *See Section 3.*
  2. Load DU Quick Charge_SDK work package; configure Androidmanifest.xml. *See Section 4.*
  3. DU Quick Charge_SDK initialization. *See Section 5.*
  4. Access Du Native Ad. See *Section 7. Section 8. Section 9. Section10.*

3 Obtain Identity

This section describes the three IDs needed during **DU Quick Charge_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 Quick Charge SDK** package, how to configure the *AndroidManifest.xml* file, and how to obfuscate code against project needs.

4.1 Load DU Quick Charge SDK Package

A. **Download** the DU Quick Charge SDK package.

- Package name: DuAD_QuickCharge1.2.zip

B. **Unzip** the package

After unzipping the package, two folders are available in the subdirectory:

- **DUAd_SDK**

  This folder stores **DU Quick Charge SDK** project package: QuickCharge_library 1.2

- **DUAd_SDK_DEMO**

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

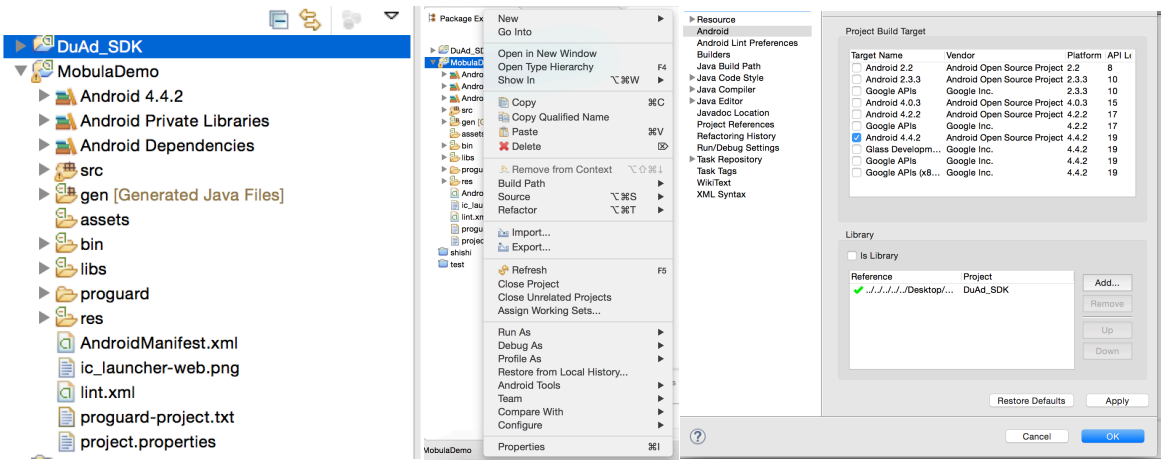
C. **Load** DU Quick Charge SDK

Import project QuickCharge_library 1.2 as android library project (see Figure 2).

Workflow: Import project QuickCharge_library 1.2 >> right click on your project >> click 'Properties' >> click 'Android' >> Under 'Library', add the new imported android library project>> click 'OK' button

*** Note: DU Quick Charge SDK** applies to **Eclipse** of Android Project. Currently for **Android Studio**, the import project needs to be created manually, please take 'Demo for Android Studio' for reference.

Figure 2 Loading **DU Quick Charge SDK** in 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 Quick Charge 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" />
        <meta-data
            android:name="du_lockscreen_action"
            android:value="com.du.action.public" />
   <provider
        android:name="com.duapps.ad.stats.DuAdCacheProvider"
        android:authorities="packagename.DuAdCacheProvider"
        android:exported="false">
</provider>
    <activity
         android:name="com.dianxinos.lockscreen.LockScreenContainer"
            android:excludeFromRecents="true"
            android:hardwareAccelerated="true"
            android:launchMode="singleInstance"
            android:noHistory="true"
            android:screenOrientation="portrait"
            android:taskAffinity=""
            android:theme="@android:style/Theme.Translucent.NoTitleBar.Fullscreen" />
```

* **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>
  <receiver
      android:name="com.dianxinos.lockscreen.ad.LockPresentReceiver">
    <intent-filter>
      <action android:name="android.intent.action.USER_PRESENT" />
    </intent-filter>
  </receiver>
```

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.

***Note: Please be sure you have registered quick charge receiver in AndroidManifest.xml as below:**

```
  <receiver
      android:name="com.dianxinos.lockscreen.ad.LockPresentReceiver">
    <intent-filter>
      <action android:name="android.intent.action.USER_PRESENT" />
    </intent-filter>
  </receiver>
```

- **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 Quick Charge 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.content.ContentProvider
-keep class com.dianxinos.lockscreen.**{*;}
```

* **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 Quick Charge SDK**.

- **Method:**

  Go **Application Class** of **OnCreate** method, using **ChargingManager.getInstance(Context context).init(String json);**

  * **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 for the **Placement_ID** and **Location_ID** as shown below:

```
{
    "native": [
        {
            "pid": "10036",
            "fbids": [
                "xxxxxx_xxxxxx"
                 ]
        }
    ],
    "list": [
        {
            "pid": "10036",
            "fbids": "xxxxxx_xxxxxxxx"
        }
    ],
    "lockscreen": [
        {
            "pid": "10036",
            "fbids": [
                "xxxxxx_xxxxxxx"
            ]
        }
    ]
}
```

- **Interface Instruction:**

  public static ChargingManager *getInstance*(Context context);

Parameters	Description
Context context	Application context or Activity Context

public static void *init* (String json);

Parameters	Description
String json	The relationship between Placement_ID and Location_ID.

- **Code Sample:**

```
String keyJson="{\"native\":[{\"pid\":\"10032\",\"fbids\":
[\"XXXXXXXXXXX\"]}],\"list\":[{\"pid\":\"10001\",\"fbids\":\"XXXXXXXXXXX\"},{\"pid\":\"10002\",\"fbids\":\"XXXXXXXXXXX\"}],\"lockscreen\":[{\"pid\":\"10036\",\"fbids\": [\"xxxxxxx_xxxxxxx\"]}]}\";
ChargingManager.getInstance(this).init(keyJson);
```

6 Quick Charge Function Access Guide

6.1 Quick Charge (ChargingManager) introduction.

- Set switch of quick charge

Interface Instruction:

public void setOpen(boolean isOpen);

Parameter	Description
boolean isOpen	Setting switch of quick charge. Quick charge is open when 'isOpen' is true. Quick charge is closed when 'isOpen' is false.

- Return status of quick charge switch

Interface Instruction:

public boolean isOpen();

Return Value	Description
boolean isOpen	Return status of quick charge switch. Quick charge is open when 'isOpen' is true. Quick charge is closed when 'isOpen' is false.

- Open log of quick charge

Interface Instruction:

```
public static void setDebug(boolean isDebug);
```

Parameter	Description
boolean isDebug	Get quick charge log. The Log function is open when 'isDebug' is true. The Log function is closed when 'isDebug' is false.

- Set quick charge setting page.

The interface is to provide quick charge setting access for developers. (It's not mandatory. It can be customized.)

Interface Instruction:

```
public void setSettingPager(LockScreenSettingPager lockScreenSettingPager)
```

Parameter	Description
LockScreenSettingPager lockScreenSettingPager	See 6.2. Abstract class (LockScreenSettingPager)

6.2 Usage of LockScreenSettingPager

- **Constructor**

Interface Instruction:

```
public LockScreenSettingPager(Context context);
```

Parameter	Description
Context context	Context

- Get the View of LockScreenSettingPager

Interface Instruction:

```
public abstract View getView();
```

Return Value	Description
View	View of LockScreenSettingPager

- Destroy the View of LockScreenSettingPager

Interface Instruction:

```
protected void dismiss();
```

6.3 Error Code of Lock Screen Ads

Error Code	Description
Code 1	Within the Ads guard period for new users
Code 2	Wthin the guard period of Ads being closed manually
Code 3	Network anomaly
Code 4	Data analysis anomaly
Code 5	Reach the upper limit of the number of daily Ads impression
Code 6	The screen lock function is closed

7 Native Ad Data Obtainment

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

7.1 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, th e 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 dynami cally**

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.

7.2 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();

7.3 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.

7.3.1 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);`
	`}`

7.3.2 Retrieve Ad Data Interfaces

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

a) **Retrieve Ad successful. DU Quick Charge SDK** can callback *onAdLoaded* method. Through the object of **DuNativeAd**, developers can use *get* method to acquire specific Ad data contents. *See 8.2.*

b) **Retrieve Ad error. DU Quick Charge 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 Quick Charge 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) {
        }
};
```

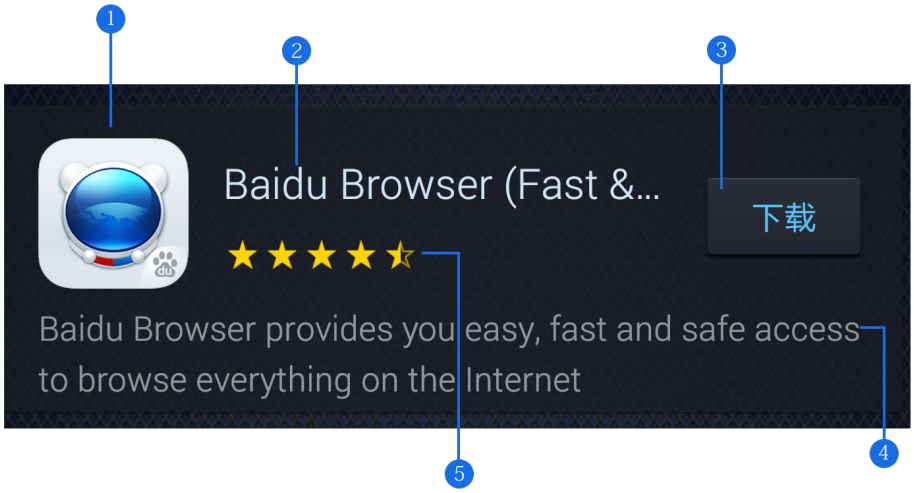
8. Native Ad Data Interface Introduction

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

8.1 Constituent Elements

Ad data's constituent elements include: Logo, title, CTA button, promotion copy, rating, promotion image, etc. *See Figure 2*.

Figure 2 Constituent elements of Ad data



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

8.2 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.

9 Register the Listener of Native Ad View

DU Quick Charge SDK can automatically count the number of times an Ad is displayed and clicked.
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.

10 The use of Native Ad List

*** Note： The whole workflow of getting Ad were done in AsyncTask. Please call this function in main thread.**

10.1 Create Class for Native Ad List

- **Interface Instruction:**

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

Parameter	Description
Context context	ACTIVITY CONTEXT
int pid	Location ID
int cacheSize	Native Ad List cache number

- **Code Sample:**

```
DuNativeAdsManager adsManager =
```

```
    new    DuNativeAdsManager(getApplicationContext(),    PID,
cacheSize);
```

***Note:** The Manger Class of Native Ad list.

10.2 Create listener interface for Native Ad List

- **Code Sample:**

```
AdListArrivalListener listener =
new AdListArrivalListener()
{
   //Callback Ad 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++)
      {
          //Obtain a single NativeAd object
          NativeAd ad = (NativeAd) arg0.get(i);
          rootContainer.addView(createItem(ad));

          //Set Ad listener for a single Native Ad object（Regester click
eventof a single Ad, see 10.3 for callback）
          ad.setMobulaAdListener(callback);
       }
       Log.d(TAG, "------end to fill view-----");
    }
    //Return Ad error code
    @Override
    public void onAdError(AdError arg0) {}
 };
```

***Note: Callback interface of getting Native Ad list**

10.3 Callback interface of Native Ad ‘s click event

- **Code Sample:**

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

***Note: This interface could obtain the click event of a single Native Ad. There is no callback for onAdLoaded() and onAdError().**

10.4 Register listener interface of Native Ad list

- **Interface Instruction:**

public void setListener (AdListArrivalListener listener);

Parameter	Description
AdListArrivalListener listener	Listener for Native Ad list

- **Code Sample:**

```
    DuNativeAdsManager adsManager = new
DuNativeAdsManager(getApplicationContext(), PID, cacheSize);
        // Set listener for Native Ad list, get Ad data from callback (see 10.2 for
    listener).
    adsManager.setListener(listener);
```

10.5 Get Native Ad list

10.5.1 Interface for filling Ad

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

Suggestion: Use the fill() at the page before the Ad showing page. Use the fill() to cache Ad in advance, for faster loading the Ad when using load().

***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();

- **Code Sample:**

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

10.5.2 Interface for loading Ad

Using *load* method, **DU Quick Charge SDK** can notify developers of Ad data result in callback function (see 10.2).

- **Interface Instruction:**

public void load();

- **Code Sample:**

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

***Note: An interface for Getting Ad. Use this interface to get Native Ad list.**

10.6 Destroy the object and listener interface of Native Ad list

When exiting out of the presentation interface of Native Ad list, the object(DuNativeAdsManager) and listener(AdListArrivalListener) of Native Ad list should be destroyed.

- **Code Sample:**

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