DU Ad Platform_SDK for Android接入手册

Version: DuWeatherSDK_1.1.6

前提:

DuWeather SDK 需要依赖 DU Ad Platform_SDK HW1.0.9.8 或 CW1.0.9.7 (含) 以上版本。

在接入 DuWeather 之前需要完成 HW 或 CW 初始化，加载，代码混淆三个部分。

---

1. 获取身份

请参照 HW 或 CW 版 DUADplatform SDK 文档 第 3 章;

申请 DuWeather 广告位时，需申请的广告位类型为「天气通」。

2. 加载与配置

请严格按照本章进行配置，否则有可能会出现运行异常。

请参照 HW 或 CW 版 DUADplatform SDK 文档第 4 章，完成后继续阅读本文档。

2.1 导入 SDK 文件

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-xW-xxx-release', ext: 'aar')
  compile(name: 'DAPSDK_Weather-release-xxx' ext: 'aar')
}
*注：flatDir 指定的位置 即为 aar 存放的位置
```

2.2 配置 AndroidManifest.xml

添加 DuWeather的额外权限：

```
//以下权限用于获取准确地理位置下发天气数据。没有该权限时 DuWeather 会使用模糊地理位置，可能导致信息下发不准确。
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />

//以下权限用于桌面悬浮窗功能，若不使用该功能可不用申请
<uses-permission android:name="android.permission.GET_TASKS"/>
<uses-permission android:name="android.permission.SYSTEM_ALERT_WINDOW" />
```

添加 DuWeather 的 Activity, receiver, service:

```
<!--dapweather begin-->
<activity
          android:name="com.daps.weather.DapWeatherActivity"
          android:screenOrientation="portrait"
/>

<receiver
android:name="com.daps.weather.reciver.DapWeatherBroadcastReceiver">
  <intent-filter android:priority="90000">
    <action android:name="android.intent.action.USER_PRESENT"/>
    //1.1版本新增
    <action android:name="com.daps.weather.broadcast"/>
  </intent-filter>
</receiver>

<service android:name="com.daps.weather.service.DapWeatherMsgService" />
```

```
<service android:name="com.daps.weather.location.DapWeatherLocationsService"
/>
<!--dapweather end-->
```

2.3 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:

将以下类添加到 proguard 配置：

```
-keep public class com.daps.weather.notification.DapWeatherNotification {
 *;
}
-keep public class com.daps.weather.weathercard.DapWeatherView {
  public <methods>;
}
-keep public class com.daps.weather.DapWeatherActivity {
  public <methods>;
}
-keep public class com.daps.weather.service.DapWeatherMsgService {
  *;
}
-keep public class com.daps.weather.location.DapWeatherLocationsService {
  *;
}
-keep public class com.daps.weather.DapWeather {
  public <methods>;
}
-keep public class com.daps.weather.weathercard.DapWeatherEnterImageView {
  public <methods>;
}
-keep public class com.daps.weather.floatdisplay.FloatDisplayController {
  *;
}
-keep public class com.daps.weather.base.SharedPrefsUtils {
  public static boolean isSuspensionOn(android.content.Context);
  public static void setSuspensionOn(android.content.Context,boolean);
}
-keep class com.daps.weather.bean.**{*;}
-keep class
com.daps.weather.notification.DapWeatherNotification$WeatherNotificationList
ener { *; }
-keep attributes InnerClasses
```

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

3. 初始化

1. 创建 Json 文件，将 Placement_ID 与广告位类型建立对应关系。具体格式如下：

```
{
  "weather": [
    {
      "pid": "YOUR_DAP_PLACEMENT_ID"
    }
  ]
}
```

2. 在 application 的`OnCreate()`方法中使用`DapWeather.init()`

   接口说明：

   > public static void init(Context context，int pid)

参数	说明
Context context	ACTIVITY CONTEXT
int pid	天气通广告位 id

   注：请在完成`DuAdNetwork.init`后进行`DapWeather.init`

   代码示例：

```
public void onCreate() {
  super.onCreate();
  //初始化 DAP SDK
  DuAdNetwork.init(this, getConfigJSON(getApplicationContext()));
  //初始化天气 SDK
  DapWeather.init((Application) getApplicationContext(),
My_Weather_pid);
}
```

3. 设置定位的更新时间`DapWeather.setLocationUpdateTime()`，默认600秒

   获取定位失败时，天气功能将不能使用。该方法为设置定位失败后再次更新的时间。建议设置间隔在1小时以内。

   接口说明：

   > public static void setLocationUpdateTime(Context context, int second)

参数	说明
Context context	ACTIVITY CONTEXT
int second	定位更新时间，单位为秒

获取当前设置的定位更新时间，单位为秒

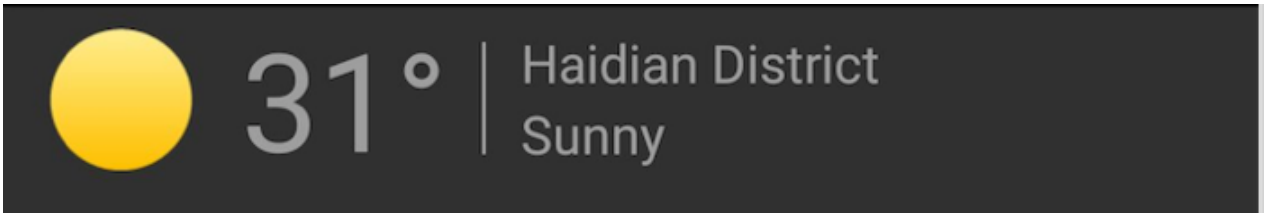
> public static int getLocationUpdateTime(Context context)

参数	说明
Context context	ACTIVITY CONTEXT

4. 功能使用

4.1 卡片控件

应用内卡片控件，主要用于在 app 内部进行天气显示。



> public DapWeatherView(Context context)

天气卡片控件，可以添加到需要显示的布局中。

> public void load()

加载天气数据，有数据自动填充 DapWeatherView 卡片。

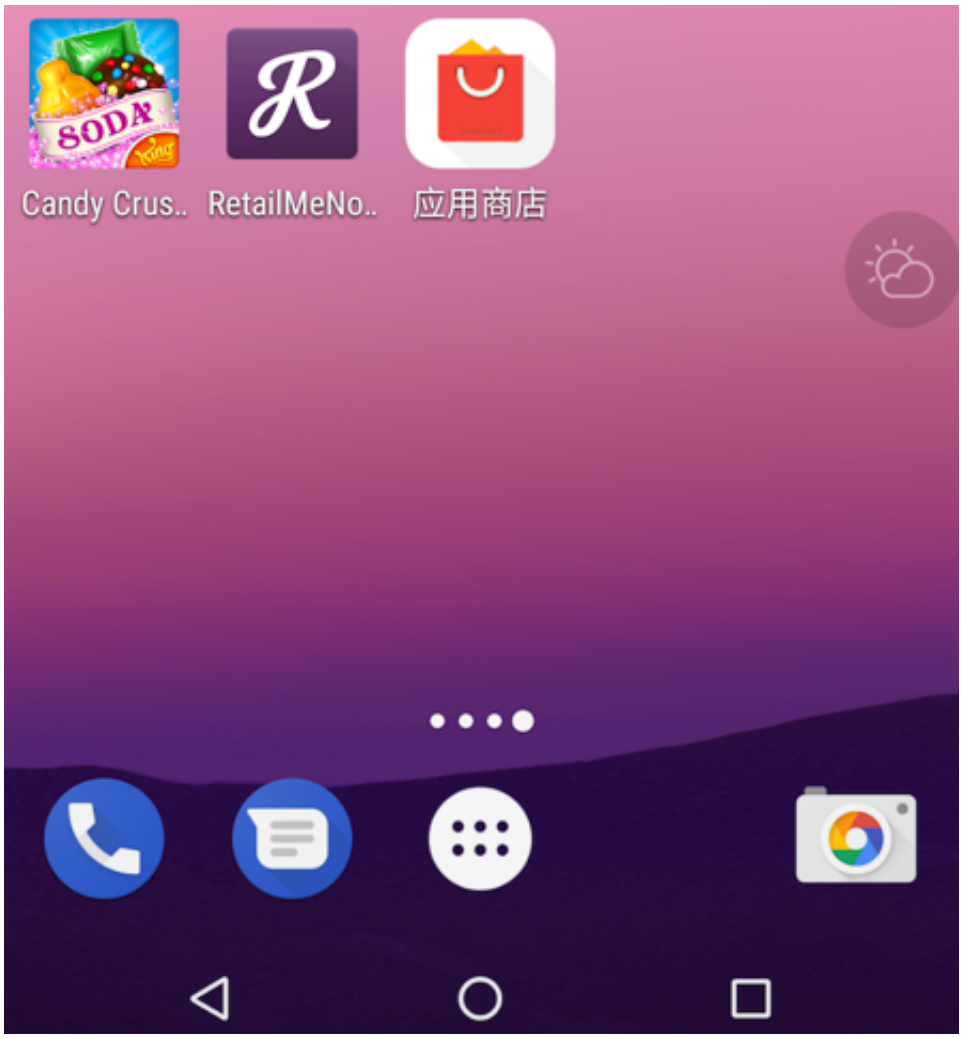
代码示例：

```
//卡片入口
RelativeLayout rl = (RelativeLayout)
findViewById(R.id.demo_weather_view_rl);
DapWeatherView mDuWeatherView = new DapWeatherView(this);
rl.addView(mDuWeatherView);
findViewById(R.id.btn_weather_view).setOnClickListener(new
View.OnClickListener() {
  @Override
  public void onClick(View view) {
    mDuWeatherView.load();
  }
});
```

4.2 桌面悬浮窗控件

该控件为在手机桌面右侧显示的一个半透明天气悬浮窗。点击后进入天气结果页。

注：在 android 6.0，需要用户手动开启悬浮窗权限，实现方式请参考Demo。



> public void FloatDisplayController.setFloatSerachWindowIsShow(Context context, Boolean isShown)

参数	说明
Context context	ACTIVITY CONTEXT
Boolean isShown	功能开关。True 表示功能开启

代码示例：

```
//悬浮框入口
findViewById(R.id.btn_weather_suspension).setOnClickListener(new
View.OnClickListener() {
  @Override
  public void onClick(View view) {

 FloatDisplayController.setFloatSerachWindowIsShow(getApplicationContext(),
true);
  }
});
```

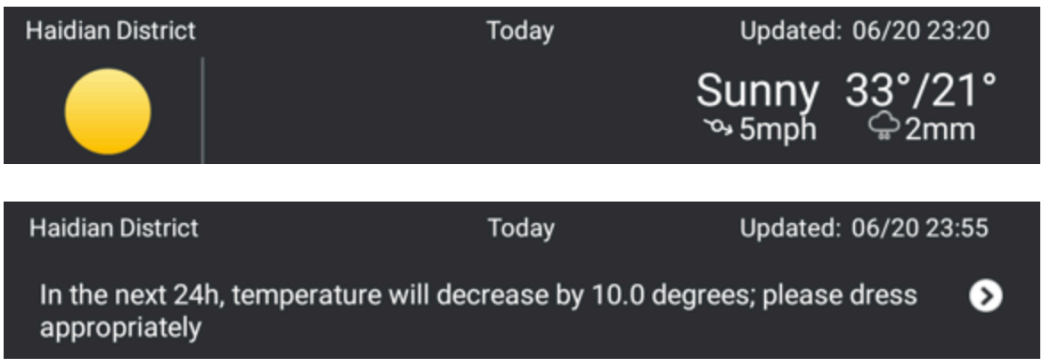
4.3 天气通知栏

该控件为定时推送的天气通知。点击后进入天气结果页。

请在应用 Application 类或子类中进行天气通知栏相关设置。如果在 Activity 中注册通知栏回调，当 Activity 被销毁时，通知栏会没有点击和展示回调。

通知栏样式：

暗色主题：



亮色主题：

注：通知栏是定时推送服务，由百度服务器控制；

7:30-12:00 推送当日天气；

13:00-20:30 推送升降温天气；

20:30-23:00 推送次日天气；

每个时间段只会推送一次天气通知，通知栏功能开启后 SDK 会自动进行通知加载。

相关接口：

> public DapWeatherNotification.getInstance(Context context)

初始化天气通知栏功能

> public void setOngoing(boolean bool)

设置通知栏常驻（不可被用户忽略），true 为常驻通知栏，false 为临时通知栏

该功能默认为 true

> public void setAutoCancel(boolean bool)

设置点击通知栏后是否自动关闭通知栏，true 为自动关闭通知栏，false为不自动关闭通知栏

该功能默认为 false，只对临时通知栏`setOngoing(false)`有效。

> public void setPushTodayTomorrowWeather(boolean bool)

设置今天和明日天气推送功能。true 为功能开启，false 为功能关闭

该功能默认为 true

> public void setPushElevatingTemperature(boolean bool)

设置升降温推送功能。true 为功能开启，false 为功能关闭

该功能默认为 true

> public void setClosedNotification(boolean bool)

设置通知栏展示是否关闭。设置关闭后已展示通知会被清除，同时不会再有新的通知。

true 为通知栏关闭，false 为通知栏开启。该功能默认为 false

> public void setNotificationStyle(int NotificationStyle)

设置通知栏整体主题颜色。设置后通知栏样式会在下次刷新时改变。

参数	说明
DapWeatherNotification.NOTIFICATIONSTYLE_DARK	黑色主题
DapWeatherNotification.NOTIFICATIONSTYLE_WHITE	白色主题

通知栏监听设置：

接口说明：

> public void setNotificationClickListener(WeatherNotificationListener adListener)

参数	说明
WeatherNotificationListener adListener	回调函数返回通知栏展示和点击事件。

```
public interface WeatherNotificationListener  {
  //通知栏展示回调
  public void onShow(int weatherType);

  //通知栏点击回调
  public void onClick(int weatherType);
}
```

`int weatherType` 数值及其含义如下：

int weatherType	含义
1	当日天气通知展示/点击
2	升降温天气通知展示/点击
3	次日天气通知展示/点击

代码示例：

```
mNotification.setNotificationClickListener(new
DapWeatherNotification.WeatherNotificationListener() {
  @Override
  public void onClick(int weatherType) {
    Log.e(TAG,"通知栏被点击了" + weatherType);
  }

  @Override
  public void onShow(int weatherType) {
    Log.e(TAG,"通知栏展示了" + weatherType);
  }
});
```