DU Ad Platform_SDK for Android
接入手册（天气通）

DUAd_SDK_WeatherWizard v1.1

百度在线网络技术（北京）有限公司

目 录

前提： WeatherWizard 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 WeatherWizard 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 WeatherWizard 广告位 ID 时 ， 需申请的广告位类型为「天气通」。

2. 加载与配置

2.1 加载 SDK 文件

1) 将 DAPSDK_Weather-release-xxx.aar 加入工程 libs 文件夹下:
2) 配置工程 build.gradle:

```
repositories {
        flatDir {
            dirs 'libs'
        }
}
dependencies {
        compile fileTree(include: ['*.jar'], dir: 'libs')
        compile(name: 'DuappsAd-xW-xxx-release', ext: 'aar')
        // WeatherSDK 必须依赖 DAP 广告 SDK
        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

A. 添加权限。WeatherSDK 使用的最低权限如下：

```
<!--定位权限-->
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
<!--悬浮框权限-->
<uses-permission android:name="android.permission.GET_TASKS"/>
<uses-permission android:name="android.permission.SYSTEM_ALERT_WINDOW" />
```

B. 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="packagename.DuAdCacheProvider"
        android:exported="false">
    </provider>
```

*注： “packagename” 为开发者 APP 的包名全称。

C. 添加天气 Activity、广播监听和服务。

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

<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.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添加到 proguard 配置，对应用代码进行混淆,否则有可能会出现运行异常:

A 把 DU Ad Platform_SDK 中的类排除在混淆之外；

B 将以下类添加到 proguard 配置：

```
-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
-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 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$WeatherNotification
Listener  {  *;  }
-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

3.1 Json 配置

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

注：用来初始化 DuAdNetwork.init() ， json 数组部分具体参见 HW & CW 版本初始化章节。

3.2 初始化函数

```
DapWeather.init((Application) getApplicationContext(), pid);
```

参数	说明
Context context	Application context
Int pid	广告位 ID

注 :在初始化 WeatherSDK 的同时，也要进行 DuAdNetwork.init()操作,才能正常展示广告（参见 HW & CW 版本初始化章节）；

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。

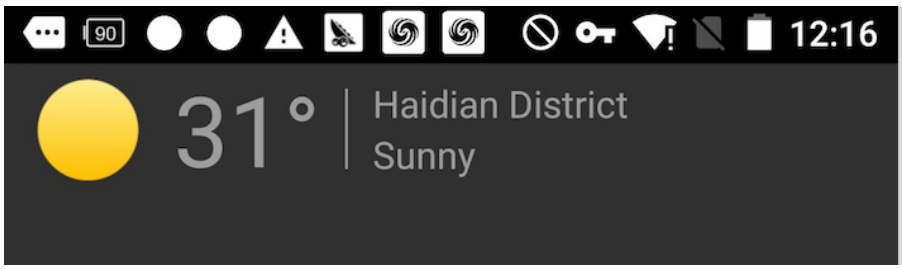

图 4-1 控件示例

- 天气卡片控件，可以添加到需要显示的布局中。
  DapWeatherView
- 加载天气数据接口
  void DapWeatherView load()

代码示例：

```
//卡片入口
RelativeLayout rl = (RelativeLayout) findViewById(R.id.demo_weather_view_rl);
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();
        //天气卡片加载天气数据，有数据自动填充 DapWeatherView 卡片。
    }
});
```

4.2 桌面悬浮窗

该控件为在手机桌面右侧显示的一个半透明天气悬浮窗。点击后直接进入天气结果页。

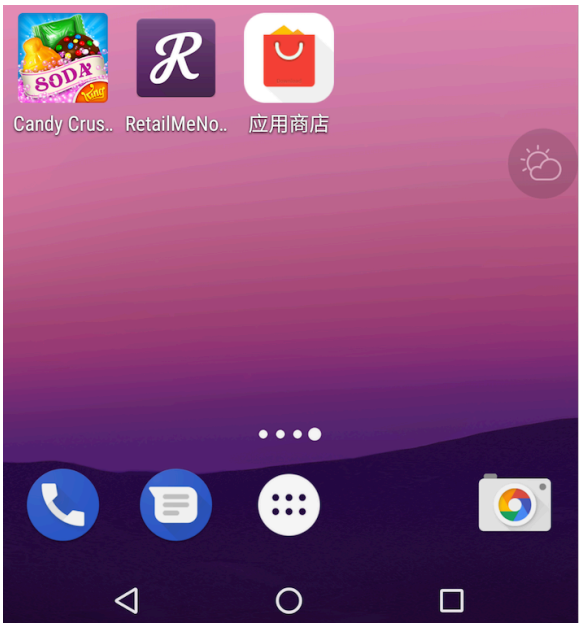

图 4-2 控件示例

注：在 android sdk 23 以上，需要用户手动开启悬浮窗权限，才能正确运行本功能。

- 悬浮窗开关

  FloatDisplayController.setFloatSerachWindowIsShow(Context context, Boolean isShown);

参数	说明
Context context	Application context
Boolean isShown	True 表示打开悬浮窗

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) {
        //悬浮窗入口，传入 true 开启，false 关闭。
        FloatDisplayController.setFloatSerachWindowIsShow(getApplicationContext(), true);
    }
});
```

4.3 通知栏天气 DapWeatherNotification

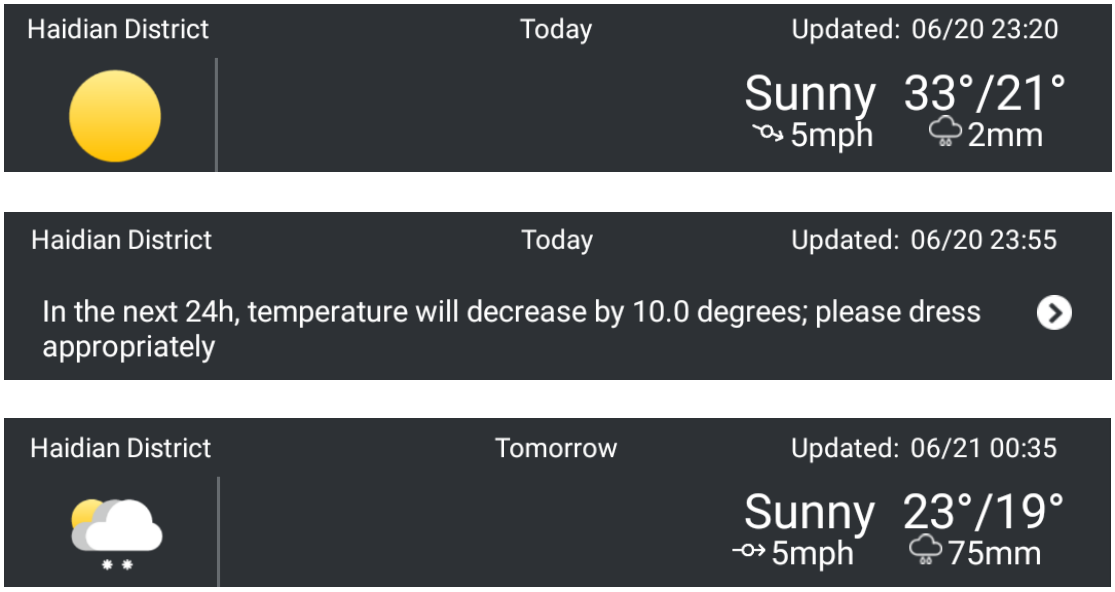

图 4-3 控件示例

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。

推荐在 Application 中设置通知栏。

- 初始化
  DapWeatherNotification.getInstance(getApplicationContext());

- 设置通知栏常驻
  DapWeatherNotification setOngoing(Boolean bool)
  true 为常驻通知栏，false 为临时通知栏

- 设置升降温推送功能
  DapWeatherNotification setPushElevatingTemperature(Boolean bool);
  是否开启升降温推送功能，默认为 true

- 设置今天和明日天气推送功能
  DapWeatherNotification setPushTodayTomorrowWeather(Boolean bool);
  是否开启今天和明日天气推送功能，默认是 true

- 通知栏监听
  DapWeatherNotification setNotificationClickListener(WeatherNotificationListener);

推荐在 Application 中设置通知栏监听。

相关回调：

- 通知栏展示事件

  public void onShow (int weatherType)

weatherType 值	说明
WeatherType = 1	当日天气通知已展示
WeatherType = 2	升降温天气通知已展示
WeatherType = 3	次日天气通知已展示

- 通知栏点击事件

  public void onClick(int weatherType)

weatherType 值	说明
WeatherType = 1	当日天气通知已被点击
WeatherType = 2	升降温天气通知已被点击
WeatherType = 3	次日天气通知已被点击

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);
    }
});
```