DU Ad Platform_SDK for Android Access Guide

Version: DuWeatherSDK_1.1.5

Precondition:

DuWeather SDK must rely on DU Ad Platform_SDK CW1.0.9.7 or HW1.0.9.8 (included) plus SDK version.

Before accesing DuWeather SDK, You need to finish initialization, loading, and code obfuscation with HW or CW version.

DuWeather is not support ads from AdMob or Facebook now.

---

1. Obtain Identity

Please refer to the chapter 3 in HW or CW version of DUADplatform SDK Access Guide to obtain necessary identities.

When applying for the DAP Placement ID , please make sure the app format you choose is **Weather Wizzard**.

2. Load SDK and Configuration

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

Please refer to the chapter 4 in HW or CW version of DUADplatform SDK Access Guide to load DU SDK.

2.1 Add Weather SDK

Copy the DAPSDK_Weather-release-xxx.aar to your Android Project, under the libs directory in root directory. Then configure 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')
}
*Note: The assigned directory of flatDir is where the aar file is placed.
```

2.2 Config AndroidManifest.xml

Add the additional needed permissions for DuWeather

```
//This permission is used to obtain accurate location for weather data. DuWeather will use a fuzzy location without this permission, which may result in inaccurate information.
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />

//The following permissions are used for floating window function. If you do not use this function, you do not need to apply.
<uses-permission android:name="android.permission.GET_TASKS"/>
<uses-permission android:name="android.permission.SYSTEM_ALERT_WINDOW" />
```

Add the additional Activity, receiver and service for DuWeather:

```
<!--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"/>
    <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 Obfuscate Code

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

Add below classes to proguard configuration:

```
-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$WeatherNotificationListene
r { *; }
-keep attributes InnerClasses
```

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

3. Initialization

1. Create a json file with mappings for the DAP Placement ID and other platform id.

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

2. Add a call to `DapWeather.init()` from `onCreate()` in your Application class.

   Interface Instruction:

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

Parameters	Description
Context context	ACTIVITY CONTEXT
int pid	The relationship between DAP Placement ID and other platform.

   Note: Please make sure you call `DuAdNetwork.init()` for DUAd SDK in advance. `DuAdNetwork.init` and `DapWeather.init` can use the same json file.

   Code sample:

```
public void onCreate() {
  super.onCreate();
  //Init DAP SDK
  DuAdNetwork.init(this, getConfigJSON(getApplicationContext()));
  //Init weather SDK
  DapWeather.init((Application) getApplicationContext(), My_Weather_pid);
}
```

3. Set the update time for location information `DapWeather.setLocationUpdateTime()`. The default value is 600 seconds.

   The weather function will not be available when getting location fails. This method sets timer for updating again after fails. It is recommended to set the timer within 1 hour.

   Interface Instruction:

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

Parameters	Description
Context context	ACTIVITY CONTEXT
int second	Location update time, the unit is second

   Get the currently location update time, the unit is second.

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

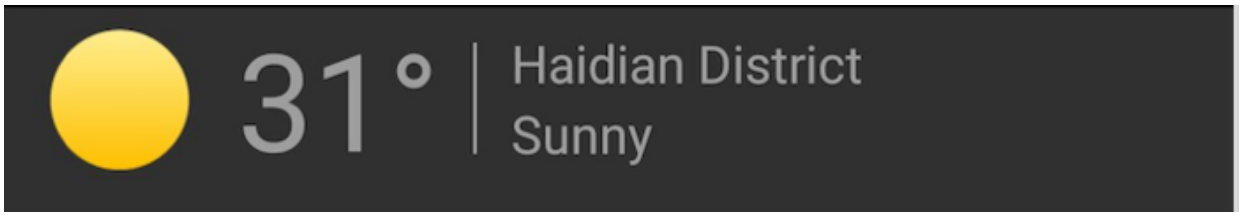
Parameters	Description
Context context	ACTIVITY CONTEXT

4. Weather Function

4.1 Weather Card

In-app weaher card, mainly used for weather display within the app. After user click the weather card, it will jump to weather landing page automatically.

Weather Card sample:



> public DapWeatherView(Context context, int cardSize)

Weather card View

cardSize value	size
VIEWSIZE_DEFAULT	300 x 45 dp
VIEWSIZE_HALFT	150 x 22.5 dp
VIEWSIZE_ONE_THIRD	100 x 15 dp
VIEWSIZE_TWO_THIRD	200 x 30 dp
VIEWSIZE_THREE_FOURTH	225 x 33.75 dp

> public void load()

Load weather data, SDK will auto-fill DapWeatherView card if data is loaded.

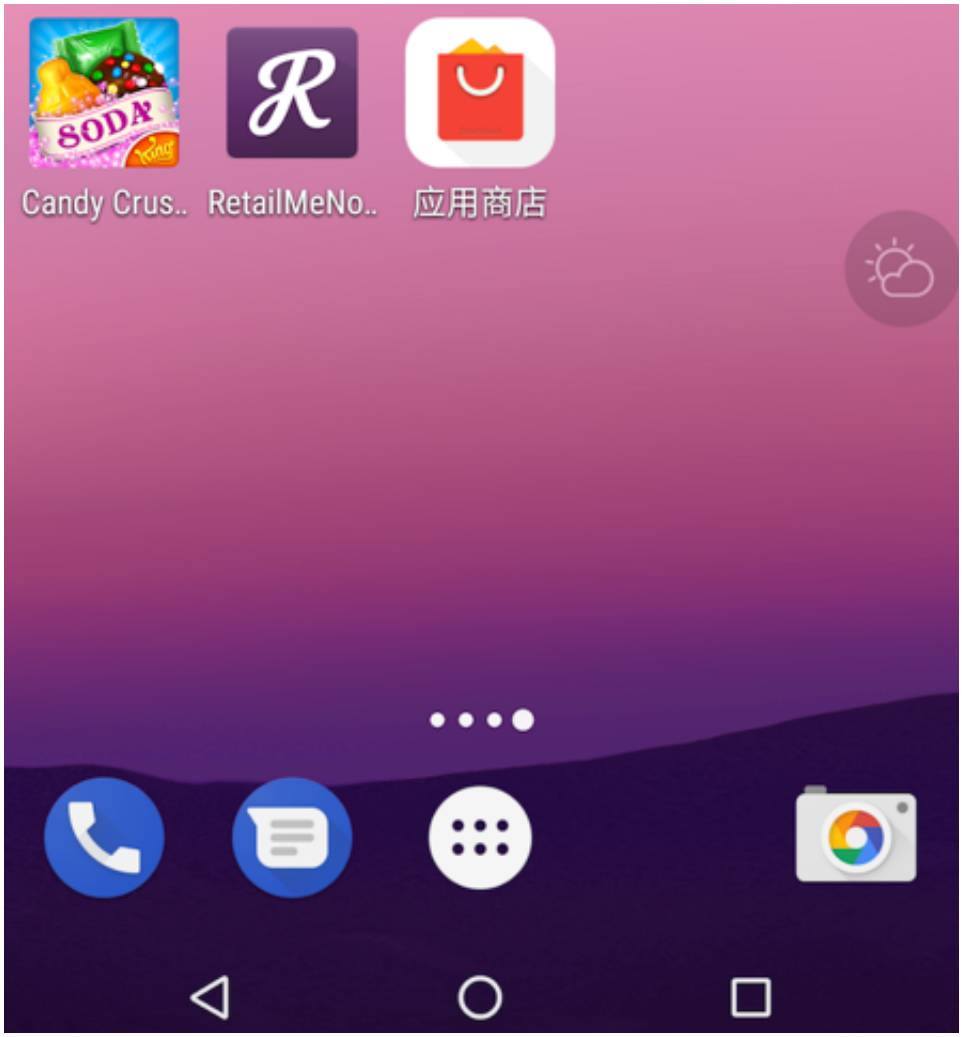
Code Sample:

```
//Add weather card view
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 Weather Floating Window

Translucent weather floating window displayed on the right side of the phone's desktop. After user click the floating window, it will jump to weather landing page automatically.

Note: In android sdk 23 or above, users need to manually open the floating window permissions in order to properly run this function.



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

Parameters	Description
Context context	ACTIVITY CONTEXT
Boolean isShown	Function switch. True for enable the feature

Code sample:

```
//Floating window entrance
findViewById(R.id.btn_weather_suspension).setOnClickListener(new
View.OnClickListener() {
  @Override
  public void onClick(View view) {
    FloatDisplayController.setFloatSearchWindowIsShow(getApplicationContext(),
true);
  }
});
```

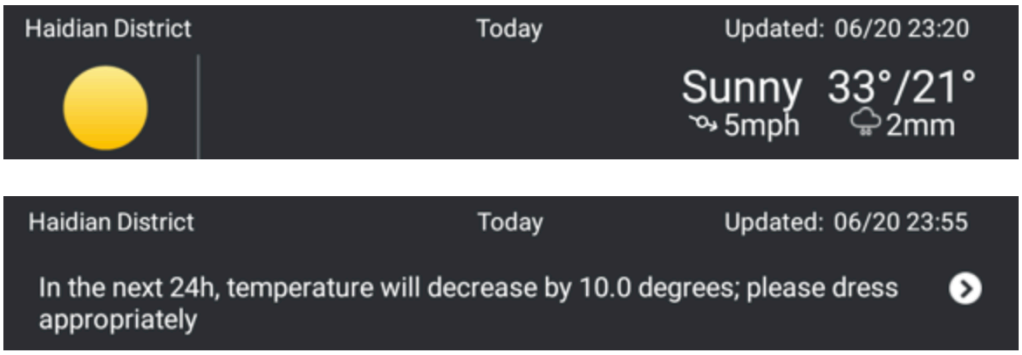
4.3 Weather Notification bar

A regularly pushed weather notification. After user click the notification, it will jump to weather landing page automatically.

Please set the weather notification bar in the **Application class or subclass**. If you register the notification bar callback in an Activity, it can not get click and impression callbacks when the Activity is destroyed.

Notification bar style:

Dark style:



White style:

The weather notification bar is controlled by DU weather SDK, which has a timer to push notification, followed by the bellow rules.

7:30-12:00 push today's weather.

13:00-20:30 push temperature rise or drop.

20.30-23:00 push tomorrow's weather.

The notification will be pushed only for one time within each time period. The SDK will automatically load notification after enabled.

Interface Instruction:

> public DapWeatherNotification.getInstance(Context context)

Initialize notification

> public void setOngoing(boolean bool)

Resident notification bar(Cannot be ignored by user), "True" for resident. "Fault" for temporary.

Default value is "True".

> public void setAutoCancel(boolean bool)

Set automatically close notification bar after click. "True" for automatically close, "False" for won't close .

Default value is "False". Tthis function only valid while `setOnGoing(False)`.

> public void setPushTodayTomorrowWeather(boolean bool)

Enable or disable notification for temperature rise or drop function.

Default value is "True".

> public void setPushElevatingTemperature(boolean bool)

Enable or disable notification for today and tomorrow's weather function or not.

Default value is "True".

> public void setClosedNotification(boolean bool)

Enable or disable all weather notification or not. After setting"True", The notification remaining will be closed, and no more notification even after `setPushElevatingTemperature(true)` and `setPushTodayTomorrowWeather(true)`.

Default value is "False".

> public void setNotificationStyle(int NotificationStyle)

Set notification style. After setting new notification style value, the notification bar will change to new style after next new push.

Parameters	Description
DapWeatherNotification.NOTIFICATIONSTYLE_DARK	Dark Style
DapWeatherNotification.NOTIFICATIONSTYLE_WHITE	White Style

Set notification listener

Interface Instruction:

> public void setNotificationClickListener(WeatherNotificationListener adListener)

Parameters	Description
WeatherNotificationListener adListener	Callback function returns: notification show and click event

```
public interface WeatherNotificationListener  {
  //Notified when the notification is shown.
  public void onShow(int weatherType);

  //Notified when the notification is clicked.
  public void onClick(int weatherType);
}
```

`int weatherType` values and meanings:

int weatherType	Description
1	Today’s weather notification is shown/clicked.
2	Temperature rise or drop notification is shown/clicked.
3	Tomorrow’s weather notification is shown/clicked.

Code sample:

```
mNotification.setNotificationClickListener(new
DapWeatherNotification.WeatherNotificationListener() {
  @Override
  public void onClick(int weatherType) {
    Log.e(TAG,"Notification clicked:" + weatherType);
  }

  @Override
  public void onShow(int weatherType) {
    Log.e(TAG,"Notification shown:" + weatherType);
  }
});
```