

Weather Wizard SDK for Android

Access Guide

DAPSDK_Weather-release-1.1

Contents

1. Obtain Identity	2
2. Load SDK and Configuration	3
2.1 Load DU Ad Platform SDK.....	3
2.2 Configure AndroidManifest.xml	3
2.3 Obfuscate Code	6
3. Initialization	7
4. Weather Card	8
5. Weather floating window.....	10
6. Weather Notification bar.....	11

Precondition:

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

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 Wizard】 .

Create placement

* Placement name :

* Steps to trigger Ads : 

* Ad Format : Native Interstitial Offerwall Banner
 Video Trigger Caller
 Weather Wizard

2. Load SDK and Configuration

2.1 Load DU Ad Platform SDK

Please refer to the chapter 4.1 in HW or CW version of DUADplatform SDK Access Guide to load DU SDK.

For accessing Weather Wizard, the below operations are needed.

- 1) Copy the `DAPSDK_Weather-release-xxx.aar` to your Android Project, under the `libs` directory in root directory
- 2) Then configure `build.gradle` :

```
repositories {
    flatDir {
        dirs 'libs'
    }
}
dependencies {
    compile fileTree(include: ['*.jar'], dir: 'libs')
    compile(name: 'DuappsAd-xW-xxx-release', ext: 'aar') // Weather Wizard
    compile(name: 'DAPSDK_Weather-release-xxx', ext:'aar')
}
SDK must rely on DU AD SDK
```

2.2 Configure AndroidManifest.xml

Please refer to the chapter 4.1 in HW or CW version of DUADplatform SDK Access Guide to configure `AndroidManifest.xml`

- A. Add a user-permission element to the manifest. For accessing Weather Wizard, except the basic permissions:

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

Additional needed permissions is as below :

```
<!--Location-->
```

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

```
<!--Floating window-->
```

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

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

- B. Add a meta-data element to the application element, and fill your DAP App ID as the value of “app_license”.

```
<application
    android:name="Your_PackageName.MobulaApplication"
    android:label="@string/app_name"
    ... >
    <meta-data
        android:name="app_license"
        android:value="@string/Your_DAP_APP_ID" />
</application>
```

- C. Declare the `com.duapps.ad.stats.DuAdCacheProvider` in the manifest. Replace the below packagename with your app’s full package name. Please make sure the package name at here is exactly the same as the package name you filled on DAP when registering you app. Otherwise, it will fail to get ad from DAP.

```
<provider
    android:name="com.duapps.ad.stats.DuAdCacheProvider"
    android:authorities="Your_packagename.DuAdCacheProvider"
    android:exported="false">
</provider>
```

D. Add the below part in your AndroidManifest.xml.

```
<!--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"/>
        //version1.1 added
        <action android:name="com.daps.weather.broadcast"/>
    </intent-filter>
</receiver>

<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>

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

2.3 Obfuscate Code

Please refer to the chapter 4.3 in HW or CW version of DUADplatform SDK Access Guide to Obfuscate Code.

A: Exclude classes of DU Ad Platform 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
-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$WeatherN
otificationListener { *; }
-keep attributes InnerClasses

```

3. Initialization

Please refer to the chapter 5 in HW or CW version of DUADplatform SDK Access Guide to finish SDK initialization.

For accessing Weather Wizard ads, please add a “weather” tag in Json, and input your corresponding placement ID.

```

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

```

- **Method:**

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

Please make sure you also call `DuAdNetwork.init()` for DUAd SDK as well. (see chapter 5 in HW or CW version of DUADplatform SDK)

- **Interface Instruction:**

```
public static void init(Context context , String pidsJson);
```

Parameters	Description
Context context	Activity Context
String pidsJson	DAP Placement ID

- **Code Sample:**

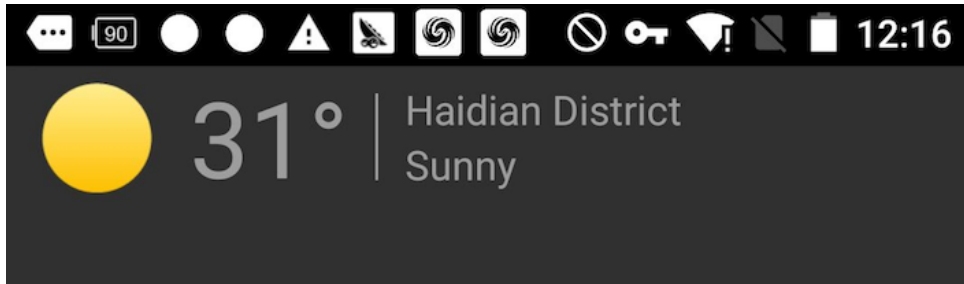
```

/**
 * Initialization Weather Wizard SDK
 * @param context Application Context
 * @param pidsJson configure json
 */
DapWeather.init((Application) getApplicationContext(),pidsJson);

```

4. Weather Card

Figure 1: Weather Card sample



Add the **DapWeatherView** to your layout and use **load()** to load weather data.

● **Code Sample:**

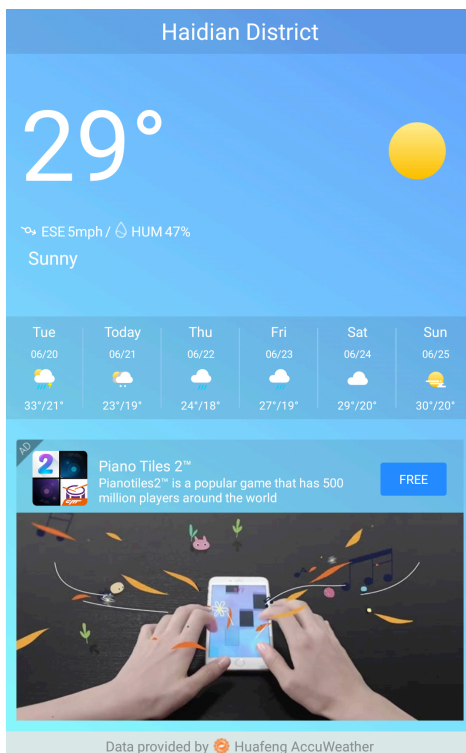
```
//Add weather card View
RelativeLayout rl = (RelativeLayout)
findViewById(R.id.demo_weather_view_rl);
mDuWeatherView = new DapWeatherView(this);
rl.addView(mDuWeatherView);

//load weather data, SDK will auto-fill
DapWeatherView card if data is loaded.
mDuWeatherView.load();
```

After user click the weather card, it will jump to weather landing page automatically.

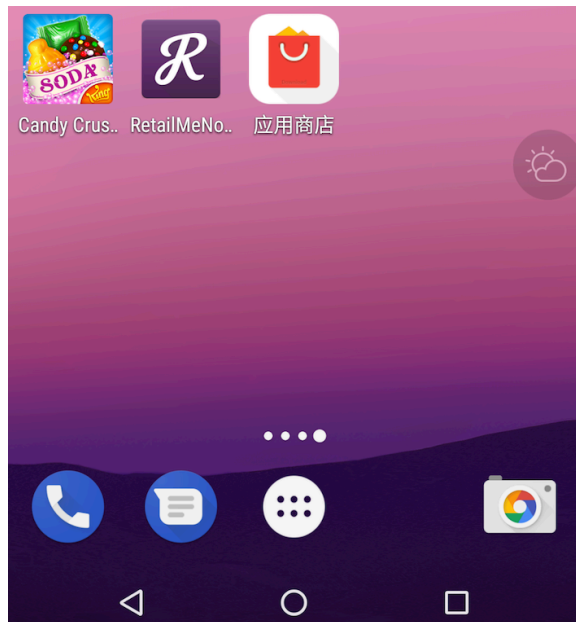
See Figure 2.

Figure 2: Weather landing page



5. Weather floating window

Figure 3: weather floating window sample



`FloatDisplayController.setFloatSerachWindowIsShow (Context context, Boolean shown)`

Parameters	Description
Context context	Activity Context
Boolean shown	"true" for enable, "false" for disable

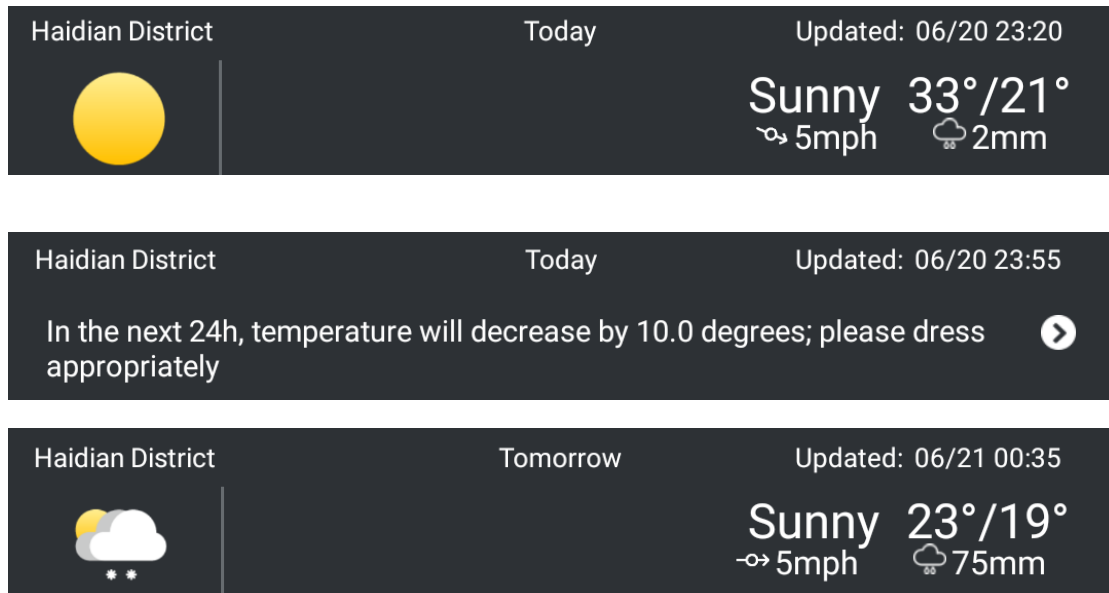
- **Code Sample:**

```
//Floating window entrance, input "true" to enable,  
input "false" to disable.  
FloatDisplayController.setFloatSerachWindowIsShow(getApplic  
ationContext(), true);  
  
FloatDisplayController.setFloatSerachWindowIsShow(getApplic  
ationContext(), false);
```

After user click the weather floating window, it will jump to weather landing page automatically. See Figure 2.

6. Weather Notification bar

Figure 4: weather notification bar sample



The weather notification bar is controlled by DU weather wizard 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.

Please set the notification in your Application class

- Initialize notification
`DapWeatherNotification.getInstance(getApplicationContext())`
`);`
- Resident notification bar
`DapWeatherNotification setOngoing(Boolean bool)`
"true" for Resident notification bar. "False" for temporary notification bar
- Set notification for temperature rise or drop
`DapWeatherNotification setPushElevatingTemperature(Boolean`

```
bool);
```

Enable or disable notification for temperature rise or drop , default value is true

- Set notification for today and tomorrow's weather

```
DapWeatherNotification setPushTodayTomorrowWeather(Boolean  
bool);
```

Enable or disable notification for today and tomorrow's weather or not ,
default value is true

- Set notification listener

```
DapWeatherNotification  
setNotificationClickListener(WeatherNotificationListener);
```

- ✧ Notified when the notification is shown.

```
public void onShow(int weatherType)
```

- ✧ Notified when the notification is clicked.

```
public void onClick(int weatherType)
```

Parameters	Description
WeatherType = 1	Today's weather notification
WeatherType = 2	Temperature rise or drop notification
WeatherType = 3	Tomorrow's weather notification

Code Sample:

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

```
Log.e(TAG,"notification is shown" + weatherType);  
    }  
});
```

After user click the weather notification bar, it will jump to weather landing page automatically. See Figure 2.