TapIt SDK for Android
Developer Guide

SDK Version 1.0

Contents

Overview

TapIt Android SDK is an advertisement SDK that is useful for showing regular banners (with ORMMA support), full screen ads, interstitials, offer walls and install tracking for applications.
See below how to enable each of these features.

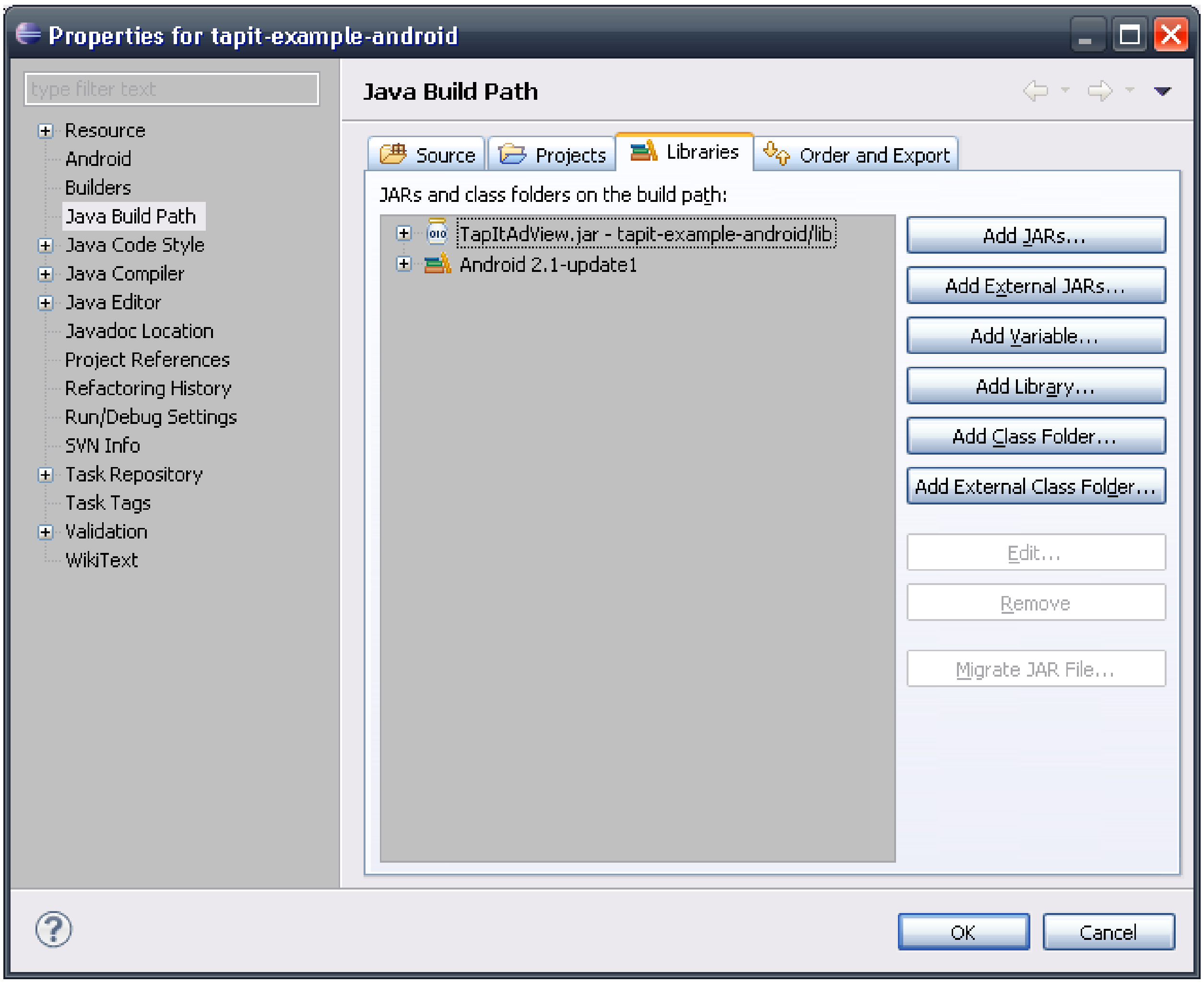
Requirements

The TapIt SDK for Android requires Android 1.6 or later.

Incorporating the SDK

Copy the TapIt SDK jar into your project folder.
Then, select Project ->Properties and add the library to the dialog under Java Build Path ->Libraries. Click"Add JARs..." (or "Add External JARs...") and select TapItAdView.jar:

Also you can add javadoc to your jar



JARs and class folders on the build path:

TapItAdView.jar - tapit-example-android/lib
Android 2.1-update1

Regular advertisement

Editing AndroidManifest.xml

These permissions are required for making requests:

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

For detecting user location these permissions are required (but if you don't want to detect user location, please don't include these)

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

And if you want to export logs to file at external storage by using AdLog.setFileLog(String filename) please add this permission:

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

For processing some ORMMA feature like playing embedded video please add this before the </application>:

```
<activity android:name="com.tapit.adview.ormma.util.OrmmaActionHandler"/>
```

For opening pages in internal browser please add this before the </application>:

```
<activity android:name="com.tapit.adview.AdActivity"
              android:configChanges="keyboard|keyboardHidden|orientation"/>
```

Internal browser will be used as default browser. But if you don't add that activity into AndroidManifest.xml system browser will be used. Also you can set whether to open URLs in internal browser via properties of AdView.

Sample usage

Dynamic creation

You can dynamically create an AdView and add it to the view hierarchy:

```
AdView adView = new AdView(this, "YOUR ZONE ID HERE");
adView.setLayoutParams(new
ViewGroup.LayoutParams(ViewGroup.LayoutParams.WRAP_CONTENT,ViewGroup.LayoutPa
rams.FILL_PARENT));
linearLayout.addView(adView);
```

Creation in layout xml

Add this into your xml that describes view hierarchy:

```
<com.tapit.adview.AdView android:id="@+id/adViewer1"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            zone="YOUR ZONE ID HERE" />
```

IMPORTANT: Whether you use dynamic or layout xml creation you should call adView.destroy() from Activity.onDestroy() (or when you decide to destroy adView) for each adView that you use. It is needed for stopping inner updating system and releasing system resources.

```
@Override
protected void onDestroy() {
      AdView adView = (AdView) findViewById(R.id.adViewer1);
      if (adView != null)
            adView.destroy();
      super.onDestroy();
}
```

Update time

Default update time is 120 sec, but you can change this value:

```
adView.setUpdateTime(300); // 300 sec
Also you can set 0 value – stop updating after first download, or null value
– default 120 sec
```

Auto-detect location

```
By default auto-detect location is enabled. But if you want you can
explicitly set these values:
double latitude = location.getLatitude();
double longitude = location.getLongitude();
adView.setLatitude(Double.toString(latitude));
adView.setLongitude(Double.toString(longitude));
```

Open in internal or system browser

```
By default pages are opened in internal browser. You can change this behavior
by setting this:
adView.setOpenInInternalBrowser(false); // default value is true
```

Event handling

Also you can handle events in this way:
adView.setOnAdDownload(new UserAdDownload()); // download process callback adView.setOnAdClickListener(new UserAdClickListener()); // block opening URL

```
...
class UserAdDownload implements OnAdDownload {
        public void begin() {
                // begin download
        }
        public void end(){
        // end download
        }
        public void error(String arg0){
        //arg0 error download
        }
}

class UserAdClickListener implements OnAdClickListener {
        public void click(String arg0){
                // to do smth with link arg0
        }
}
```

Background transparency

You can set transparent background in this way:
`adView.setBackgroundColor(Color.TRANSPARENT);`

Logging and testing

AdLog.setDefaultLogLevel(AdLog.LOG_LEVEL_3); // set logLevel for all banners in app AdLog.setFileLog("/sdcard/log.txt"); // set log file

Where logLever can be one of:

AdLog.LOG_LEVEL_NONE	none
AdLog.LOG_LEVEL_1	only errors
AdLog.LOG_LEVEL_2	+warning
AdLog.LOG_LEVEL_3	+server traffic

NOTE: See javadoc for details about available methods.

Full-screen advertisements

Interstitial advertisement

You can show interstitial ad for showing advertising in full screen mode in the center of screen:

```
AdInterstitialView interstitialView = new AdInterstitialView(this, "YOUR ZONE ID HERE");
interstitialView.setShowCloseButtonTime(2);
interstitialView.setAutoCloseInterstitialTime(5); // 0 value – disable auto close.
interstitialView.setIsShowPhoneStatusBar(false); // option. Default true
interstitialView.show();
```

Note*:* For this view it is not needed to call destroy, because it is called when “close” button is pressed.

Full-screen advertisement

You can show full-screen ad. It presents full screen advertising during some time, cannot be closed before timeout expiration:

```
AdFullscreenView fullscreenView = new AdFullscreenView(this, "YOUR ZONE ID HERE");
fullscreenView.setAutoCloseTime(10);
fullscreenView.setIsShowPhoneStatusBar(false); // option. Default true
fullscreenView.show();
```

Note*:* For this view it is not needed to call destroy, because it is called when “close” button is pressed.

Offer Wall advertisement

You can show OfferWall ad. It presents list of offer-advertising list in full screen mode:

```
AdOfferWallView offerWallView = new AdOfferWallView(this, "YOUR ZONE ID HERE");
offerWallView.setIsShowPhoneStatusBar(false); // option. Default true
offerWallView.show();
```

Note*:* For this view it is not needed to call destroy, because it is called when “close” button is pressed.

Video Unit advertisement

You can show video ad. It presents full screen video advertising during some time, cannot be closed before timeout expiration:

```
AdVideoUnitView adVideoUnitView = new AdVideoUnitView(this, "YOUR ZONE ID HERE");
adVideoUnitView.setIsShowPhoneStatusBar(false); // option. Default true
adVideoUnitView.setAutoCloseTime(45);
adVideoUnitView.show();
```

Note*:* For this view it is not needed to call destroy, because it is called when “close” button is pressed.

Install tracking and Event tracking

Install tracking

For tracking your installation please add this code into Activity.onCreate() method of your main activity:

```
InstallTracker.getInstance().reportInstall(this);
or
InstallTracker.getInstance().reportInstall(this, "YOUR CAMPAIGN ID HERE");
Also you can change log level for logs for this class:
InstallTracker.getInstance().setLogLevel(AdLog.LOG_LEVEL_3);
```

Event tracking

For tracking any event in your app please add call like this into the right place:

```
EventTracker.getInstance().reportEvent(this, "Show InterstitialView");

Also you can change log level for logs for this class:
EventTracker.getInstance().setLogLevel(AdLog.LOG_LEVEL_3);
```