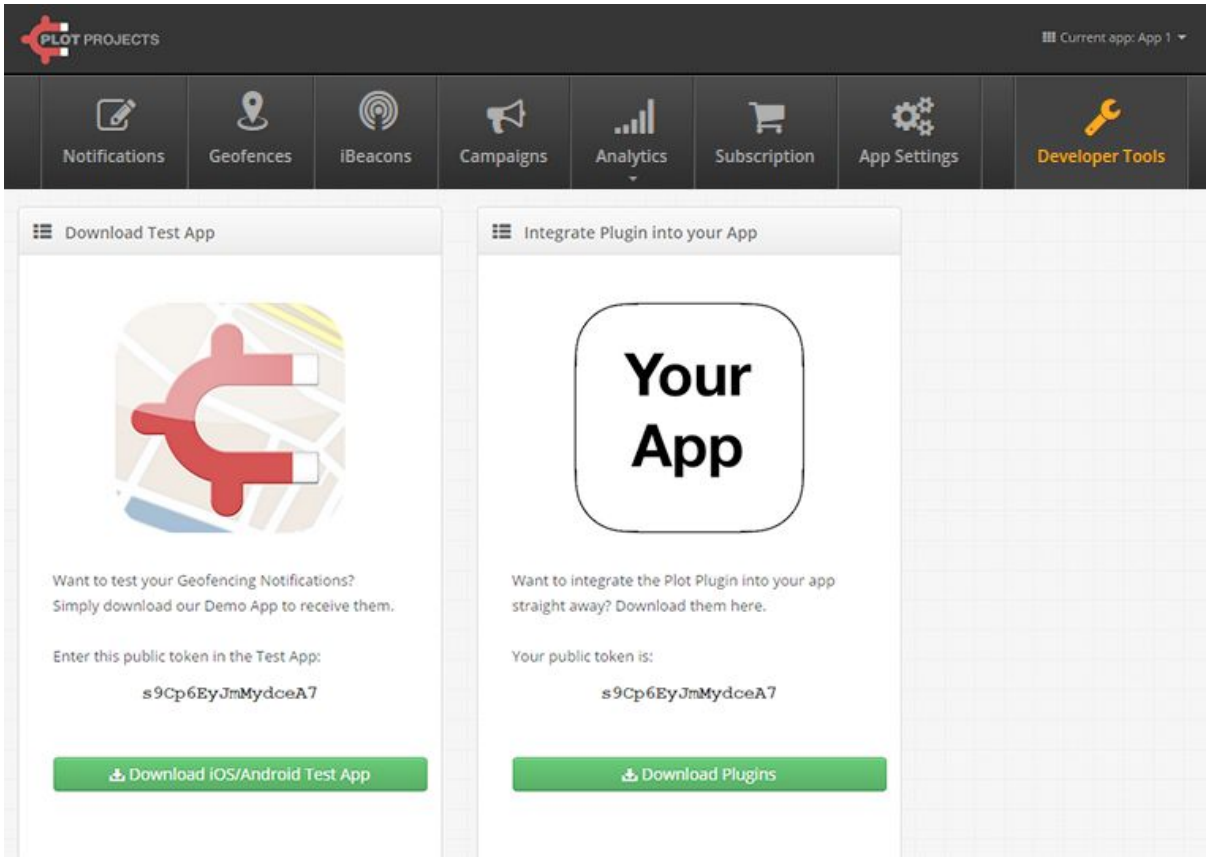
Android Plugin 2.0 (alpha) Easy Integration Guide

Thank you for trying out our new plugin. This plugin version is more accurate while being much more battery efficient, but definitely try it out for yourself. If you have any feedback regarding the plugin integration or the plugin itself, please let us know. You can contact us through the contact form or you can mail us at info@plotprojects.com.

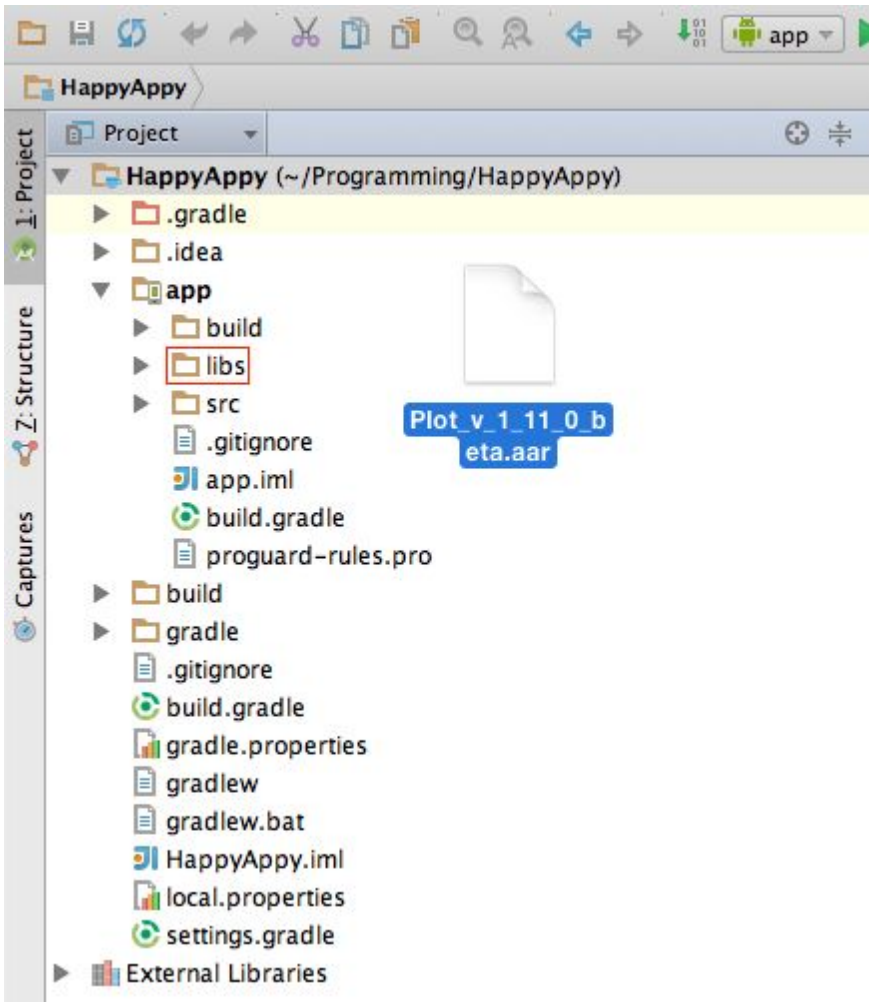
Step 1: Get the library

Once you log into our dashboard, download the library under the “Developer Tools” tab.



Step 2: Place the aar file

Place the file Plot_v_<version>.aar in the libs folder of your app project. If the project doesn’t have a folder named libs yet, then create the folder.



Add it to your project dependencies. Replace the version in the filename with the version of your plugin.

```
repositories {
   jcenter()
   flatDir {
     dirs "libs"
   }
}
dependencies {
   compile name:'plot_v_2_0_0_alpha, ext: 'aar'
}
```

Step 3: Add the dependencies

The plugin achieves its best results when it has access to Google Play Services. Therefore we strongly recommend adding these libraries to your project. Also the plugin depends on the a compat support library.

```
dependencies {
    compile 'com.google.android.gms:play-services-gcm:10.0.1'
    compile 'com.google.android.gms:play-services-location:10.0.1'
```

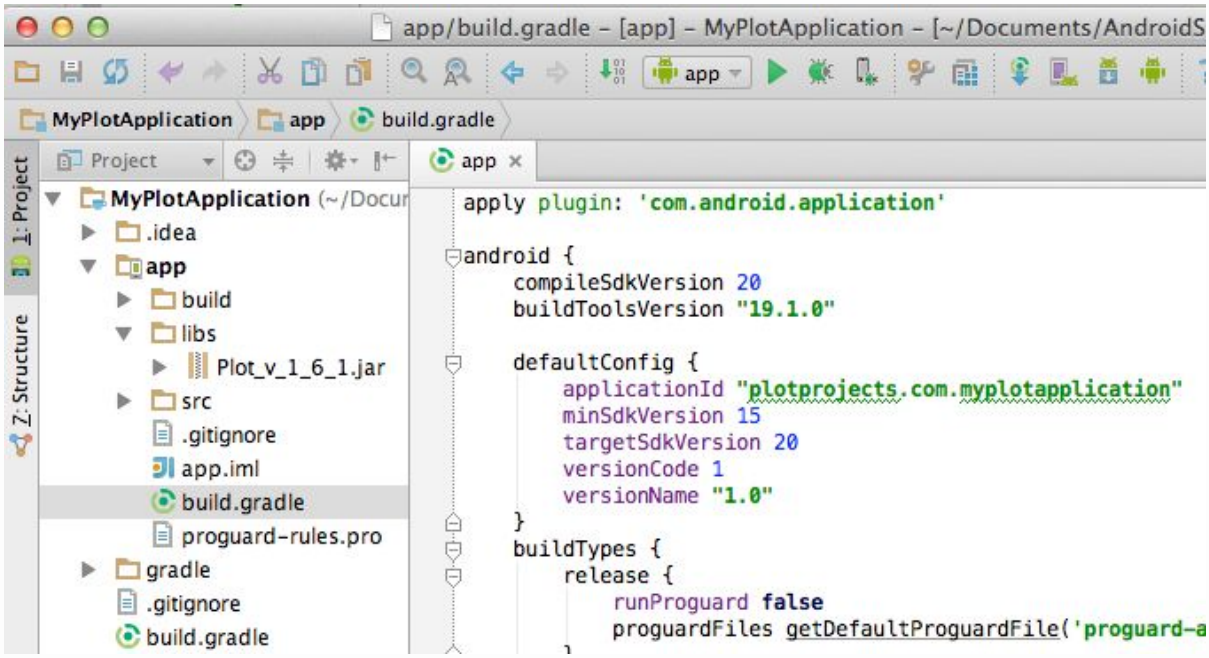
```
    compile 'com.google.android.gms:play-services-nearby:10.0.1'
    compile 'com.android.support:support-v4:25.1.0'

}
```

Instead of the *com.android.support:support-v4* it is also possible to use the newer *com.android.support:support-compat* library. This may be needed when it collides with one of the transitive dependencies of your own dependencies.

Step 4: Set the MinSDK version

Set the minSdkVersion version to at least 9. When using Gradle you can do that by changing build.gradle in the folder of your app.

```
defaultConfig {
    applicationId "com.myplotapplication"
    minSdkVersion 9
    targetSdkVersion 22
    versionCode 1
    versionName "1.0"
}
```



Step 5: create the plotconfig.json

Plot uses a configuration file which you have to define as plotconfig.json in your assets folder. When the folder doesn't exist yet, you can create it under src/main.

It is possible to have a separate plotconfig.json file for each build type/flavour. When the plotconfig file is placed at *src/debug/assets*, then it is only used for debug builds. Similarly *src/release/assets* is only used for release builds.

An example of such a config file can be found on our dashboard, as well as the public token you will have to use.

```
{
    "publicToken": "REPLACE_ME",
    "enableOnFirstRun": true,
    "debug": true
}
```

Step 6: Call Plot.init()

Call *Plot.init(activity)* to initialize the library from your main Activity.

When calling init from an activity, you can use the *this* keyword as the first parameter.

```
public class MyActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_my);

        Plot.init(this);
    }
}
```

An example configuration file and your public token can be found on the Download page on our dashboard. You are now ready to receive your first notification. Need more help during testing, look at this guide.

For more details, look at the extensive documentation. There is no longer a separate guide for beacon support, since that is now available by default.