



From time to time Google changes the way it does things, and old tutorials may not apply to some new procedures.

This is another tutorial which, in about 6 months, will probably be irrelevant.

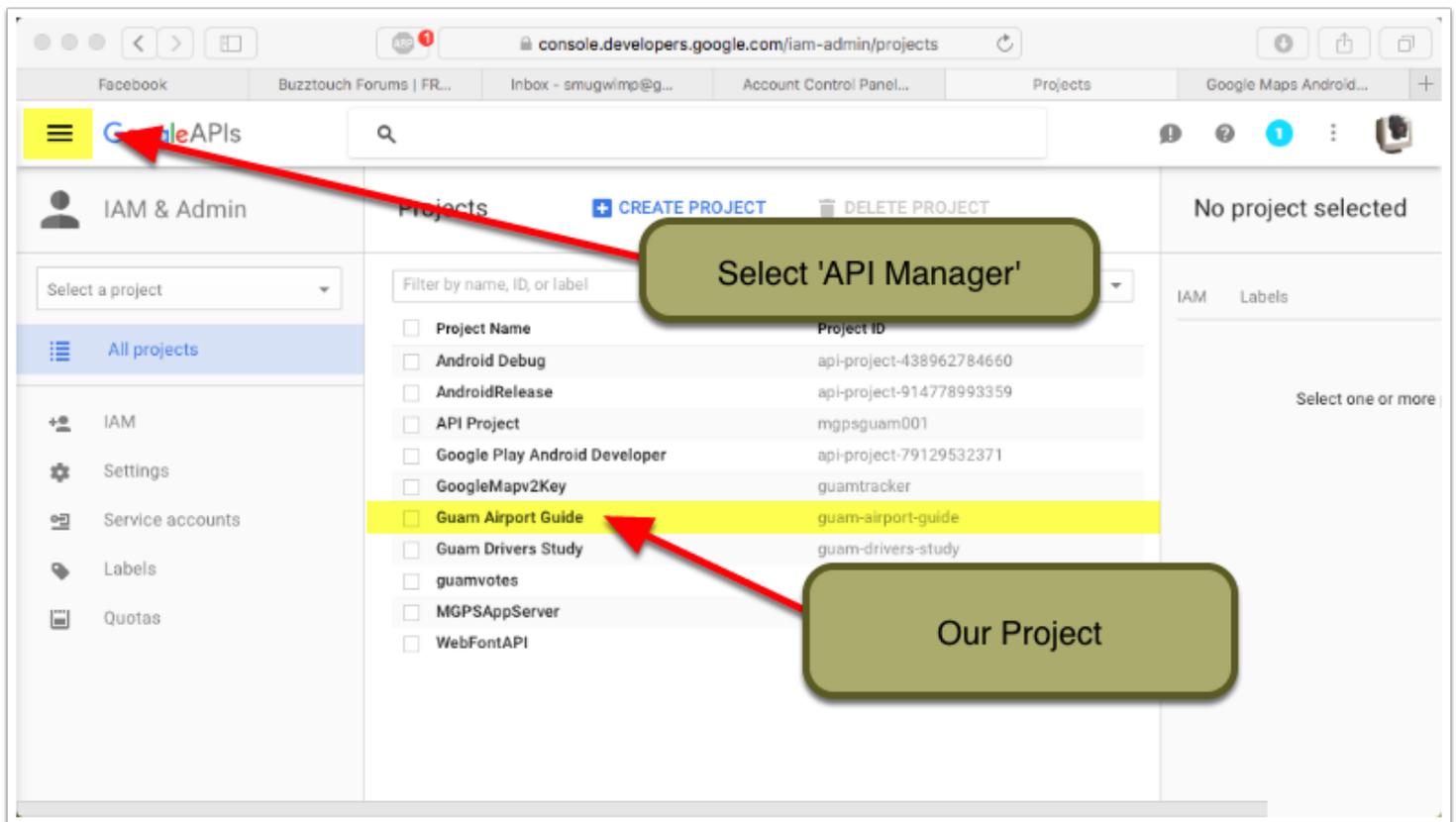
But until then...

Login to your Google Developer Console

You need to know how to get at least this far.

Login to your Google Developer Console

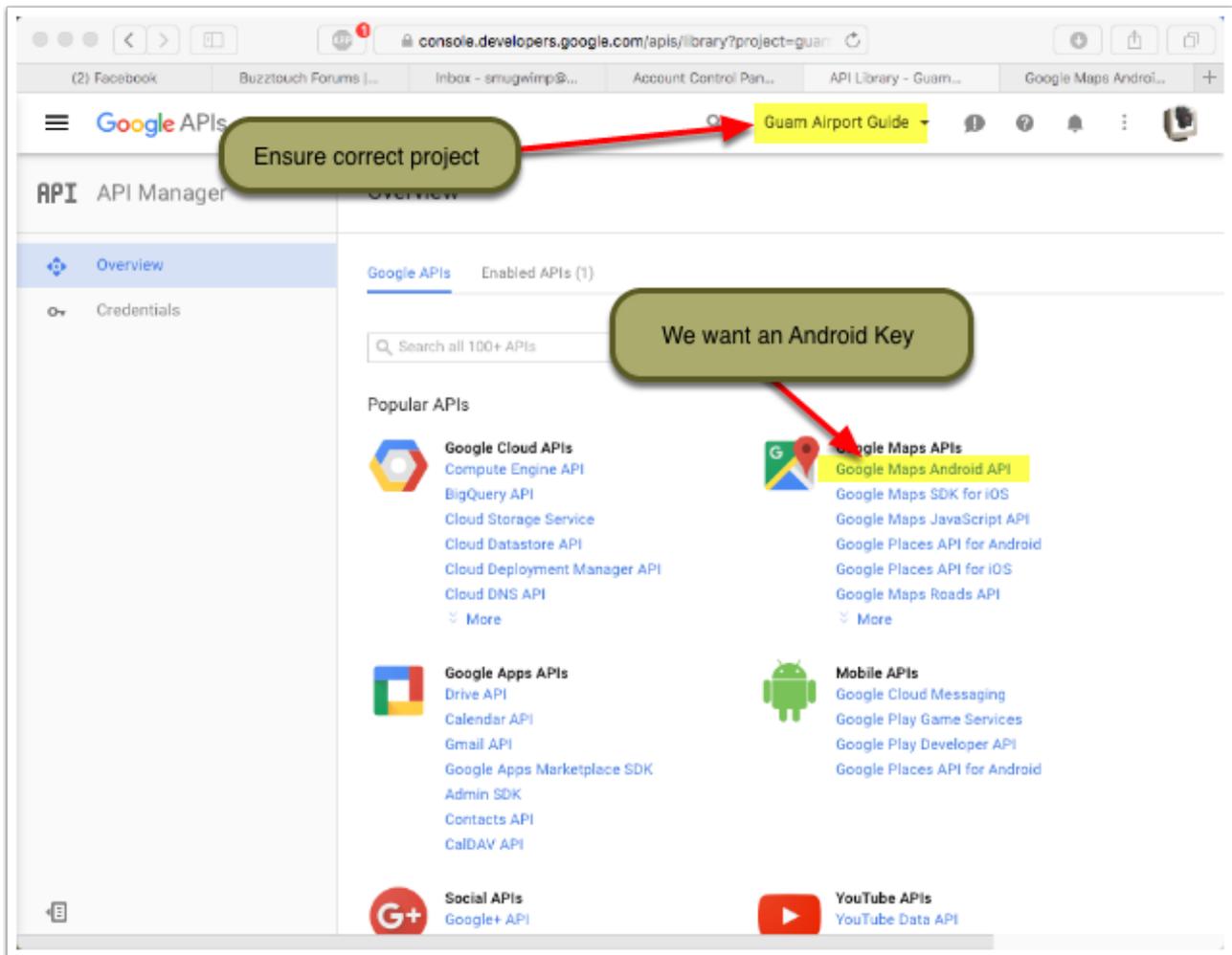
Create a project, or select a project to obtain an API





Select 'API Manager' for your project

Click on 'Google Maps Android API'





Enable the API

Press the 'Enable' button

The screenshot shows the Google Developers console interface. The browser address bar displays `console.developers.google.com/apis/api/maps_android_backend/overview/`. The page title is "API Manager" and the current view is "Overview". A red arrow points from a green callout box labeled "Press Button" to the "Enable" button. The "Enable" button is a blue button with a white left-pointing arrow. Below the button, the text reads "Google Maps Android API" and "Add maps based on Google Maps data to your Android application with the Google Maps Android API. The API automatically handles access to Google Maps servers, map display and response to user gestures such as clicks and drags." There is a "Learn more" link. Below that, it says "Using credentials with this API" and "Using an API key". The text explains that an API key is needed to check quotas and access, and provides a "Learn more" link. To the right, a diagram shows "Your application" (represented by a purple code icon) connected to "API key" (represented by a green key icon) which is then connected to "Google service" (represented by a server rack icon).



Enabled, but...

console.developers.google.com/apis/api/maps_android_backend/overview

Google APIs

API Manager

Overview

Disable

Google Maps Android API

⚠ This API is enabled, but you can't use it in your project until you create credentials. Click "Go to Credentials" to do this now (strongly recommended).

[Go to Credentials](#)

Overview Usage Quotas

Add maps based on Google Maps data to your Android application using the Google Maps Android API. The API automatically handles requests to Google Maps servers, map display and response to user interactions such as drags.

[Learn more](#)

Using credentials with this API

Using an API key

To use this API you need an API key. An API key identifies your project to check quotas and access. Go to the Credentials page to get an API key. You'll need a key for each platform, such as Web, Android, and iOS. [Learn more](#)

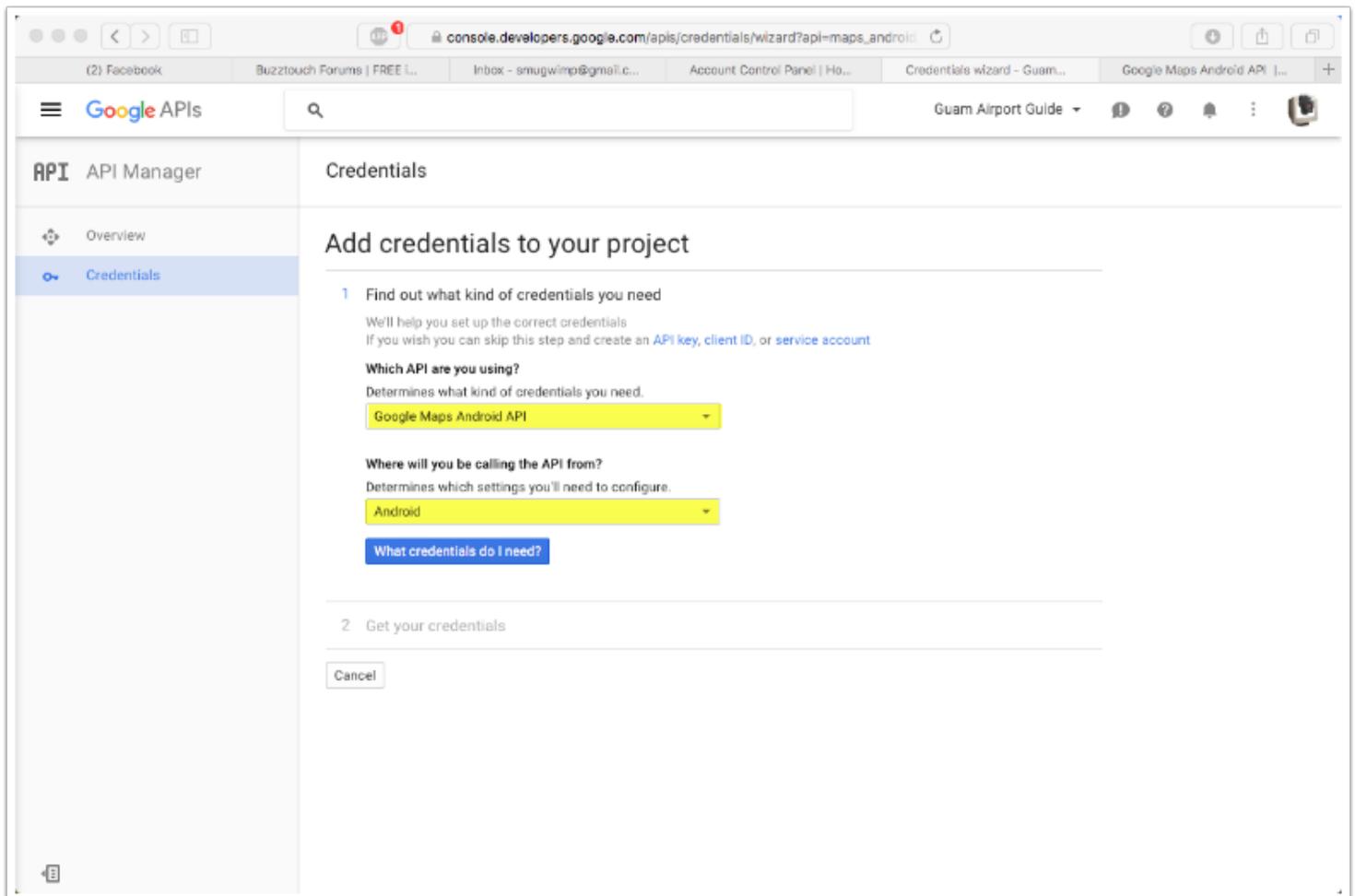
We need to setup our credentials for the App



Select your API Parameters

Select the type of API so that correct credentials can be determined

Press "What Credentials Do I Need" when complete.





Create Credentials - Ensure correct package name

Before we go too much further, I'd like to mention that if you don't like your package/app name, now is the time to change it. If you create credentials, and then try to change your app name, you'll have to come back, and do all of this over again. It's better to go ahead and change the name first.

This is the package name that you are 'signing' with to create your API key. If you don't like your app name, NOW is the time to change it, before you do any credentials.

```
1  apply plugin: 'com.android.application'
2
3  android {
4      compileSdkVersion 23
5      buildToolsVersion "23.0.2"
6      useLibrary 'org.apache.http.legacy'
7
8      defaultConfig {
9          applicationId "com.guamflights"
10         minSdkVersion 14
11         targetSdkVersion 16
12     }
13
14     buildTypes {
15         release {
16             minifyEnabled false
17             proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-rules.pro'
18         }
19     }
20 }
21
22 de
23
24
25
26
27
28
```

following components are ready to update: Android Supp

console.developers.google.com/apis/credentials/wizard?

Google APIs

API Credentials

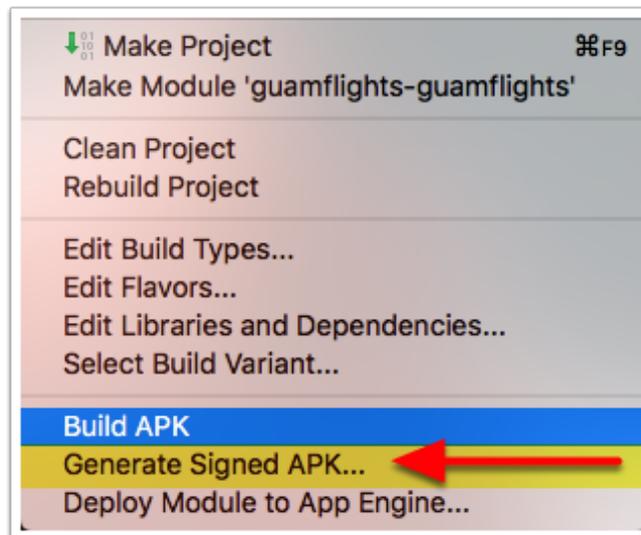
Add credentials to your project

- 1 Find out what kind of credentials you need
Calling Google Maps Android API from Android
- 2 Create an API key
 - Name: GuamAirportGuide
 - Restrict usage to your Android apps (Optional)
Add your package name and SHA-1 signing-certificate fingerprint to restrict usage to your Android apps [Learn more](#)
Get the package name from your AndroidManifest.xml file. Then use the following command to get the fingerprint:
keytool -list -v -keystore mystore.keystore
 - Package name: com.guamflights
 - SHA-1 certificate fingerprint: 12:34:56:78:90:AB:CD:EF:12:34:56:78:90:AB:CD:EF:AA:BB:CC:DD
 - + Add package name and fingerprint
 - Create API key
- 3 Get your credentials



Create a App 'Key' in your release keystore

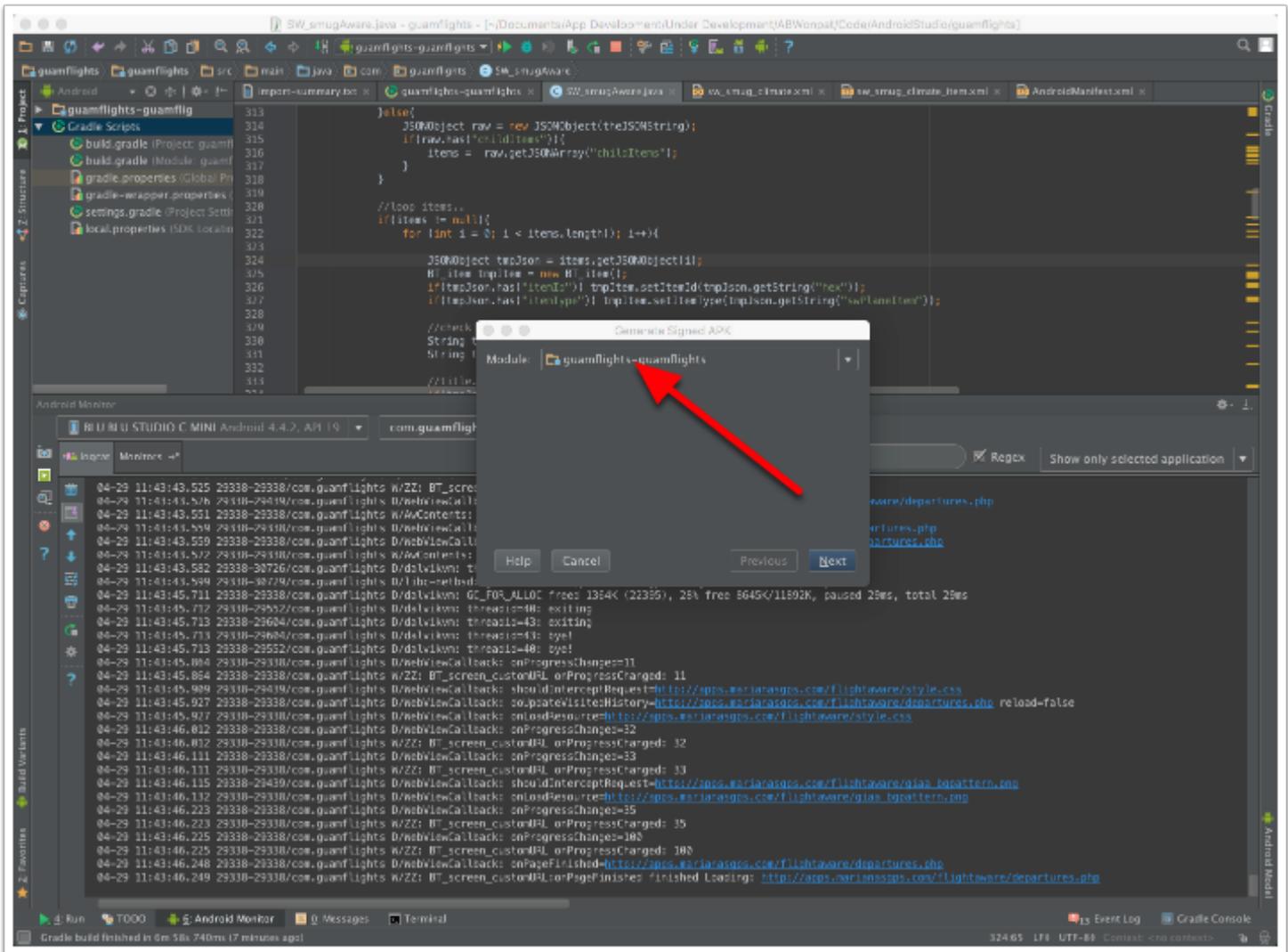
The easiest way to do this, since I can't seem to find a keytool plugin for Android Studio, is to create a signed release apk. It doesn't matter if it's your finished product or not; the key won't change, and that's where we need the information from.





Make sure you're in the right place

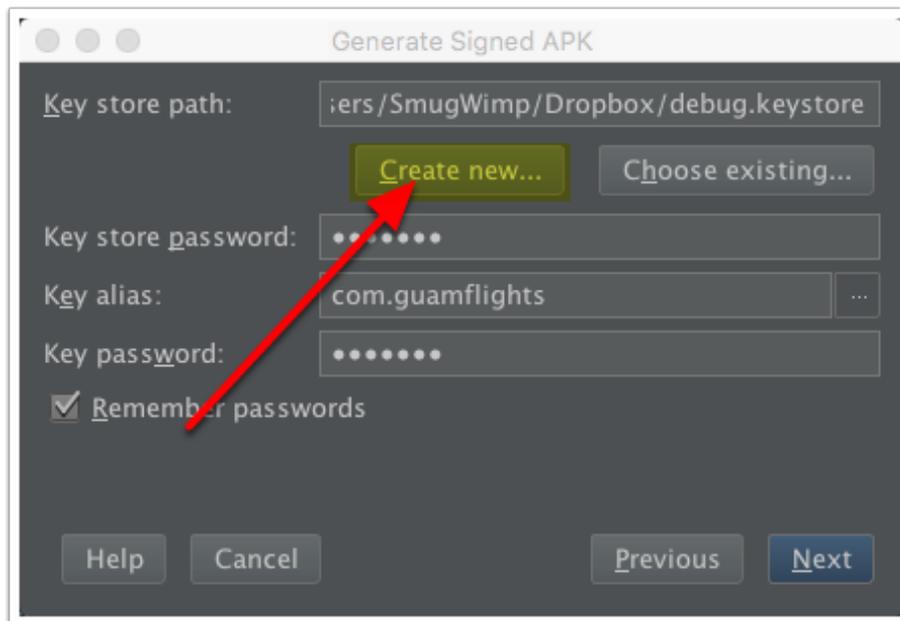
It sounds silly, but make sure you're creating the key for the right module. Depending on how complex your app is, sometimes it can come into question.





Create New Key

If you haven't created a key for the app, this is where you start. Press the 'Create New' button.





Fill in the information

1. Select the keystore. This is my debug keystore, although I said release earlier. You probably want to navigate and select your release keystore. When you select the proper keystore, enter and confirm the 'keystore' password. This is not the app key password. This is the 'master access' password for the keystore. By default, the debug keystore password is 'android'. "You" create your release keystore, so only you know what it is. And it should stay that way.
2. The details about your app key and alias. The alias can be anything, but keeping it pertinent is always wise. You need to create a password for the app key, and confirm it. Don't ever lose these passwords, or these keystores. Never ever ever. Set the validity to something outrageous.
3. Somewhat personal information for the certificate that needs to be created.
4. Once all that is done, press 'OK'.

New Key Store

Key store path: /Users/SmugWimp/Dropbox/debug.keystore 1

Password: Confirm:

Key

Alias: com.yourappname 2

Password: Confirm:

Validity (years): 125

Certificate

First and Last Name: Smug Wimp

Organizational Unit: Mobile Development

Organization: SmugWimp Enterprises LTD 3

City or Locality: Tamuning

State or Province: GU

Country Code (XX): US

Cancel OK 4



A Quick Note about Keys and Signing...

Open up terminal, and navigate to the same directory as your keystore. From there, type the following command, substituting your correct values:

```
keytool -list -v -keystore your.release.keystore -alias com.yourAppName
```

Press 'enter' or 'return' or whatever. It should spout out some information about that particular key. Included in that information, is the 'SHA-1 fingerprint' which is what we want.

```
Dropbox — -bash — 120x30
Use "keytool -command_name -help" for usage of command_name
[SmugBookPro:Dropbox SmugWimp$ keytool -list -v -keystore ./real.release.keystore -alias guamairportguide
]
[Enter keystore password:
Alias name: guamairportguide
Creation date: Apr 29, 2016
Entry type: PrivateKeyEntry
Certificate chain length: 1
Certificate[1]:
-----
Valid from: Fri Apr 29 20:02:08 ChST 2016 until: Sat Apr 17 20:02:08 ChST 2066
Certificate fingerprints:
  MD5: 1C:75:E0:D6:E5:F4:56:6A:0C:2B:C2:63:C3:92:13:E3
  SHA1: D7:54:D4:4C:44:46:DD:57:69:A0:E3:E0:9E:7A:36:E8:01:0A:66:BD
  SHA256: E5:D4:84:F0:B9:DF:3F:BE:EF:75:C6:6B:32:67:06:0B:32:CC:E6:D4:2D:40:05:67:1A:BF:07:D3:22:10:4C:3A
Signature algorithm name: SHA256withRSA
Version: 3

Extensions:
-----
]
]
SmugBookPro:Dropbox SmugWimp$
```



Add your SHA-1 Fingerprint, and press 'create api key'.

Your key will appear in the next section. Copy and paste this into the appropriate places in your app.

The screenshot shows the Google APIs console interface. At the top, there is a hamburger menu icon and the text "Google APIs". Below this is a table with two columns: "API" and "Credentials". The "API" column contains a diamond icon and a key icon. The "Credentials" column contains the text "Add credentials to your project". Below this, there are three steps listed with green checkmarks:

- 1 Find out what kind of credentials you need
Calling Google Maps Android API from Android
- 2 Create an API key
Created API key 'GuamAirportGuide'
- 3 Get your credentials
Here is your API key

The API key is displayed in a text box: AIzaS...8mb20wI. At the bottom of the dialog, there are two buttons: "Done" and "Cancel".



Your API is finished

Now that you're done creating your API key, it's time to put it into your BT App.

The screenshot shows the Google APIs console interface. At the top, there is a hamburger menu icon and the text "Google APIs". Below this, there is a navigation bar with "API" and "Credentials". Under "Credentials", there are three tabs: "Credentials" (selected), "OAuth consent screen", and "Domain verification". Below the tabs, there is a "Create credentials" button with a dropdown arrow and a "Delete" button. Below these buttons, there is a text instruction: "Create credentials to access your enabled APIs. Refer to the API documentation for details." Below this, there is a section titled "API keys" with a table of API keys.

<input type="checkbox"/>	Name	Creation date	Type	Key
<input type="checkbox"/>	GuamAirportGuide	Apr 29, 2016	Android	AlzaSyAWPmHYeo



1) AndroidManifest

Add the key into your Android Manifest file

```
<!--  
  Google Maps v2 API Key  
  Replace "GOOGLE_MAPS_FOR_ANDROID_V2_API_KEY_GOES_HERE" on the next line with the Google  
  See: https://developers.google.com/maps/documentation/android/start#installing_the_g  
-->  
<meta-data android:name="com.google.android.maps.v2.API_KEY" android:value="AIzaSyAWPmHY0  
  
<!-- Google Cloud Messaging -->  
<receiver android:name="com.gyamflights.BT_gcmReceiver"  
  android:permission="com.google.android.c2dm.permission.SEND" >
```



Make sure you're referencing the Google Play Services

Make sure that you're referencing the latest Google Play Services in both your Build.Gradle and your AndroidManifest

The diagram illustrates the required code for enabling Google Play Services. It features two code snippets with callouts pointing to them:

- build.gradle:** A code block showing dependencies. A red box highlights the following lines:

```
dependencies {  
    compile files('libs/PdfViewer.jar')  
    compile files('libs/gcm.jar')  
    // compile 'com.android.support:support-v4:23.0.0'  
    compile 'com.google.android.gms:play-services:+'  
}
```

A red arrow points from the 'build.gradle' callout to the highlighted code.
- AndroidManifest:** A code block showing the following XML snippet:

```
<!-- Google Play Services  
Your project must reference the Google Play Service library project.  
See http://developer.android.com/google/play-services/setup.html  
-->  
<meta-data android:name="com.google.android.gms.version" android:value="@integer/google_play_services_version" />
```

A red arrow points from the 'AndroidManifest' callout to the XML code.



Add your Google API Key in your Android Manifest

Copy the Google Maps API key provided in your Google Developer Console. Paste in the appropriate location in your AndroidManifest file.

```
<!--  
  Google Maps v2 API Key  
  Replace "GOOGLE_MAPS_FOR_ANDROID_V2_API_KEY_GOES_HERE" on the next line with the Google Maps for Android API Key provided  
  See: https://developers.google.com/maps/documentation/android/start#installing\_the\_google\_maps\_android\_v2\_api  
-->  
<meta-data android:name="com.google.android.maps.v2.API_KEY" android:value="AIza...eM8"/>
```



Remove the debug declaration. It gets annoying.

Remove the line:

```
android:debuggable="false"
```

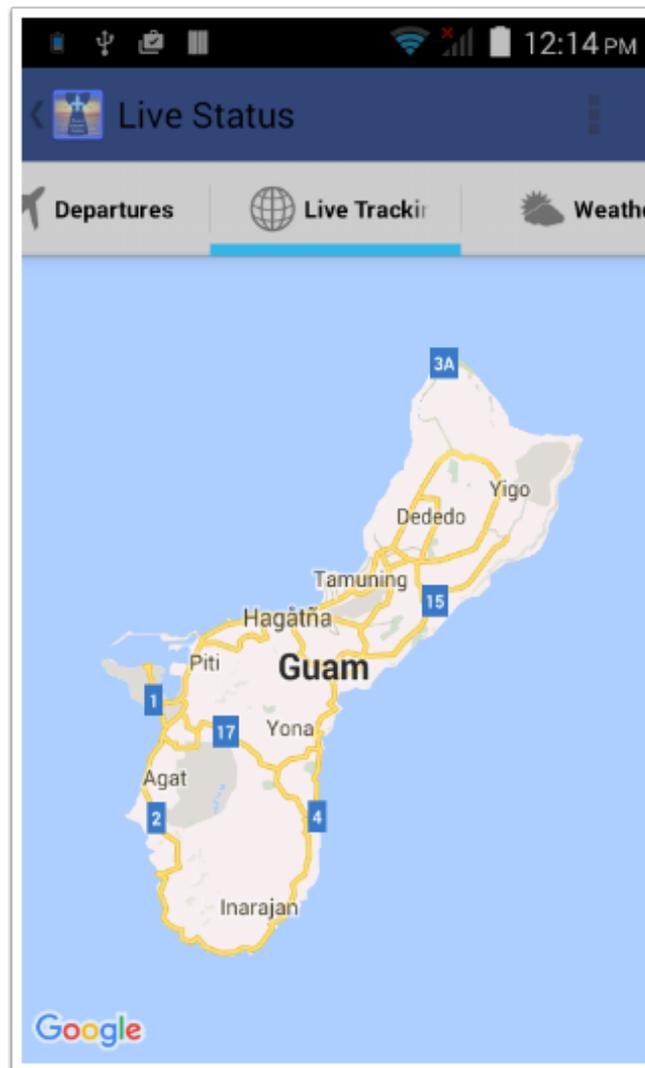
Remember to keep the definition capped ">"

```
<!-- application -->  
<application android:name="guamflights_appDelegate"  
             android:icon="@drawable/icon"  
             android:label="@string/app_name"  
             android:theme="@style/hostThemeWithTitle"  
             android:hardwareAccelerated="false"  
             >
```



And that is the name of that game.

Provided good omens are upon us, Google Maps should appear in the whatever plugin. If it does not, CHECK THE LOGCAT. That is where the errors will tell you what didn't work, and possibly why. This is important information for troubleshooting, so don't forget.





Good Luck. Happy Appy.

Cheers. If you have questions, don't message me. Post it in the forums, so that all may benefit from your question or problem.