Getting Started

Before you can begin working with the API, you will need to download the API and ensure that you have a Google Maps Android API v2 key. Both the API and the key are freely available.

Overview

Creating a new Android application that uses the Google Maps Android API v2 requires several steps. Many of the steps outlined in this section will only have to be performed once, but some of the information will be a handy reference for future applications. The overall process of adding a map to an Android application is as follows:

1. Download and configure the Google Play services SDK. The Google Maps Android API is distributed as part of this SDK.
2. Obtain an API key. To do this, you will need to register a project in the Google APIs Console, and get a signing certificate for your app.
3. Specify settings in the Application Manifest.
4. Add a map to a new or existing Android project.
5. Publish your application!

You may wish to begin by looking at some sample code, which is included with the Google Play services SDK.

Getting the Google Maps Android API v2

The API is distributed as part of the Google Play services SDK, which you can download with the Android SDK Manager. To use the Google Maps Android API v2 in your app, you will first need to install the Google Play services SDK. To learn how to install the package, see the Google Play services documentation.

As a prerequisite, you need to install the Android SDK. To learn how to do this, see Installing the SDK.

The Google Maps API Key

> **Note:** The Google Maps Android API v2 uses a new system of managing keys. Existing keys from a Google Maps Android v1 application, commonly known as MapView, will not work with the v2 API.

To access the Google Maps servers with the Maps API, you have to add a Maps API key to your application. The key is free, you can use it with any of your applications that call the Maps API, and it supports an unlimited number of users. You obtain a Maps API key from the Google APIs Console by providing your application's signing certificate and its package name. Once you have the key, you add it to your application by adding an element to your application's manifest file `AndroidManifest.xml`.

Understanding the process of registering your application and obtaining a key requires some knowledge of Android's publishing process and requirements. In summary, all Android applications must be signed with a digital certificate for which you hold the private key. Because digital certificates are unique, they provide a simple way of uniquely identifying your app. This makes them useful for tracking your application in systems such as Google Play Store, and for tracking your application's use of resources such as the Google Maps servers.

> **Note:** Refer to the Android Guide Signing Your Applications for more information regarding digital certificates.

Maps API keys are linked to specific certificate/package pairs, rather than to users or applications. You only need one key for each certificate, no matter how many users you have for an application. Applications that use the same certificate can use the same API key. However, the recommended practice is to sign each of your applications with a different certificate and get a different key for each one.

Obtaining a key for your application requires several steps. These steps are outlined here, and described in detail in the following sections.

1. Retrieve information about your application's certificate.
2. Register a project in the Google APIs Console and add the Maps API as a service for the project.
3. Once you have a project set up, you can request one or more keys.
4. Finally, you can add your key to your application and begin development.

Displaying certificate information

The Maps API key is based on a short form of your application's digital certificate, known as its **SHA-1 fingerprint**. The fingerprint is a unique text string generated from the commonly-used SHA-1 hashing algorithm. Because the fingerprint is itself unique, Google Maps uses it as a way to identify your application.

To display the SHA-1 fingerprint for your certificate, first ensure that you have the certificate itself. You may have two certificates:

- **Debug certificate**: The Android SDK tools generate this certificate automatically when you do a "debug" build from the command line, or when you build and run a project from Eclipse without exporting it as a released application. The certificate is only for use with an application that you're testing; you can't publish an app that's signed with a debug certificate. The debug certificate is described in more detail in the section Signing in Debug Mode in the Android Developer Documentation. You can generate an API key from this certificate, but only use the key for testing, never for production.
- **Release certificate**: The Android SDK tools generate this certificate when you do a "release" build with either `ant` program or Eclipse. You can also generate this certificate using the `keytool` program. This certificate can be used with an app you release to the world. Once you have the correct certificate for your needs, you can display its SHA-1 fingerprint using the `keytool` program.
- For more information about Keytool, see the documentation at http://docs.oracle.com/javase/6/docs/technotes/tools/windows/keytool.html.

▸ Displaying the debug certificate fingerprint

▸ Displaying the release certificate fingerprint

Creating an API Project

Once you have your signing certificate fingerprint, create or modify a project for your application in the Google APIs Console and register for the Maps API.

To get a project and register for the API:

1. In a browser, navigate to the Google APIs Console.
   - If you haven't used the Google APIs Console before, you're prompted to create a project that you use to track your usage of the Google Maps Android API. Click **Create Project**; the Console creates a new project called **API Project**. On the next page, this name appears in the upper left hand corner. To rename or otherwise manage the project, click on its name.
   - If you're already using the Google APIs Console, you will immediately see a list of your existing projects and the available services. It's still a good idea to use a new project for Google Maps Android API, so select the project name in the upper left hand corner and then click **Create**.
2. You should see a list of APIs and services in the main window. If you don't, select **Services** from the left navigation bar.
3. In the list of services displayed in the center of the page, scroll down until you see **Google Maps Android API v2**. To the right of the entry, click the switch indicator so that it is **on**.
4. This displays the Google Maps Android API Terms of Service. If you agree to the terms of service, click the checkbox below the terms of service, then click **Accept**. This returns you to the list of APIs and services.

You're now ready to get a Maps API key.

Obtaining an API Key

If your application is registered with the Google Maps Android API v2 service, then you can request an API key. It's possible to register more than one key per project.

To get the key:

1. In the left navigation bar, click **API Access**.
2. In the resulting page, click **Create New Android Key...**.
3. In the resulting dialog, enter the SHA-1 fingerprint, then a semicolon, then your application's package name. For example:

```
BB:0D:AC:74:D3:21:E1:43:67:71:9B:62:91:AF:A1:66:6E:44:5D:75;com.example.an
```

4. The Google APIs Console responds by displaying **Key for Android apps (with certificates)** followed by a forty-character API key, for example:

```
AIzaSyBdVl-cTICSwYKrZ95SuvNw7dbMuDt1KG0
```

5. Copy this key value. You will use it in the next step.

Adding the API Key to your application

The final step is to add the API key to your application. It goes in your application's manifest, contained in the file `AndroidManifest.xml`. From there, the Maps API reads the key value and passes it to the Google Maps server, which then confirms that you have access to Google Maps data.

To add the key to your application:

1. In `AndroidManifest.xml`, add the following element as a child of the `<application>` element, by inserting it just before the closing tag `</application>`:

```
<meta-data
    android:name="com.google.android.maps.v2.API_KEY"
    android:value="API_KEY"/>
```

substituting your API key for *API_KEY*. This element sets the key `com.google.android.maps.v2.API_KEY` to the value *API_KEY* and makes the API key visible to any `MapFragment` in your application.

2. Add the following elements to your manifest. Replace `com.example.mapdemo` with the package name of your application.

```
<permission
        android:name="com.example.mapdemo.permission.MAPS_RECEIVE"
        android:protectionLevel="signature"/>
<uses-permission android:name="com.example.mapdemo.permission.MAPS_RECEIVE
```

3. Save `AndroidManifest.xml` and re-build your application.

Specify settings in the Application Manifest

An Android application that uses the Google Maps Android API needs to specify the following settings in its manifest file, `AndroidManifest.xml`:

- Permissions that give the application access to Android system features and to the Google Maps servers.
- Notification that the application requires OpenGL ES version 2. External services can detect this notification and act accordingly. For example, Google Play Store won't display the application on devices that don't have OpenGL ES version 2.
- The Maps API key for the application. The key confirms that you've registered with the Google Maps service via the Google APIs Console.

This section describes each of these settings and how to add them to `AndroidManifest.xml`.

Specifying permissions

Set permissions by adding <uses-permission> elements as children of the <manifest> element. The syntax is:

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

For example, to request the Internet permission, add:

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

Besides permissions required by other parts of your application, you must add the following permissions in order to use the Google Maps Android API:

- `android.permission.INTERNET` Used by the API to download map tiles from Google Maps servers.
- `android.permission.ACCESS_NETWORK_STATE` Allows the API to check the connection status in order to determine whether data can be downloaded.
- `com.google.android.providers.gsf.permission.READ_GSERVICES` Allows the API to access Google web-based services.
- `android.permission.WRITE_EXTERNAL_STORAGE` Allows the API to cache map tile data in the device's external storage area.

The following permissions are recommended, but can be ignored if your application does not access the user's current location, either programmatically, or by enabling the My Location layer.

- `android.permission.ACCESS_COARSE_LOCATION` Allows the API to use WiFi or mobile cell data (or both) to determine the device's location.
- `android.permission.ACCESS_FINE_LOCATION` Allows the API to use the Global Positioning System (GPS) to determine the device's location to within a very small area.

```
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/
<uses-permission android:name="com.google.android.providers.gsf.permission
<!-- The following two permissions are not required to use
     Google Maps Android API v2, but are recommended. -->
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
```

Requiring OpenGL ES version 2

Because version 2 of the Google Maps Android API requires OpenGL ES version 2, you must add a <uses-feature> element as a child of the <manifest> element in `AndroidManifest.xml`:

```
<uses-feature
        android:glEsVersion="0x00020000"
        android:required="true"/>
```

This notifies external services of the requirement. In particular, it has the effect of preventing Google Play Store from displaying your app on devices that don't support OpenGL ES version 2.

Add a Map

After you've added references to the Google Play services SDK, added your key and customized your Android Manifest, you can try adding a map to your application.

The easiest way to test that your application is configured correctly is to add a simple map. You will have to make changes in two files: `main.xml` and `MainActivity.java`. Please note that the code below is only useful for testing your settings in an application targeting Android API 12 or later, This code should not be used in a production application. Examples of how to add more robust code appear throughout this guide and in the sample code.

1. In `main.xml`, add the following fragment.

```
<?xml version="1.0" encoding="utf-8"?>
<fragment xmlns:android="http://schemas.android.com/apk/res/android"
          android:id="@+id/map"
          android:layout_width="match_parent"
          android:layout_height="match_parent"
          android:name="com.google.android.gms.maps.MapFragment"/>
```

2. In `MainActivity.java`, add the following code.

```
package com.example.mapdemo;

import android.app.Activity;
import android.os.Bundle;

public class MainActivity extends Activity {

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

3. Build and run your application. You should see a map. If you don't see a map, confirm that you've completed all of the steps appearing earlier in this document.

Last updated June 6, 2013.