

GCM (Google Cloud Message) Integration - Android

www.AnswerCart.com

1. General Information

The purpose of this document is to define the process of the GCM integration by which an Android app can be notified whenever there is any event on the community.

1.1.Intended Audience

This document is intended for community administer and other IT professionals responsible for integrating the internal process with AnswerCart Community product.

1.2.Purpose of GCM Integration

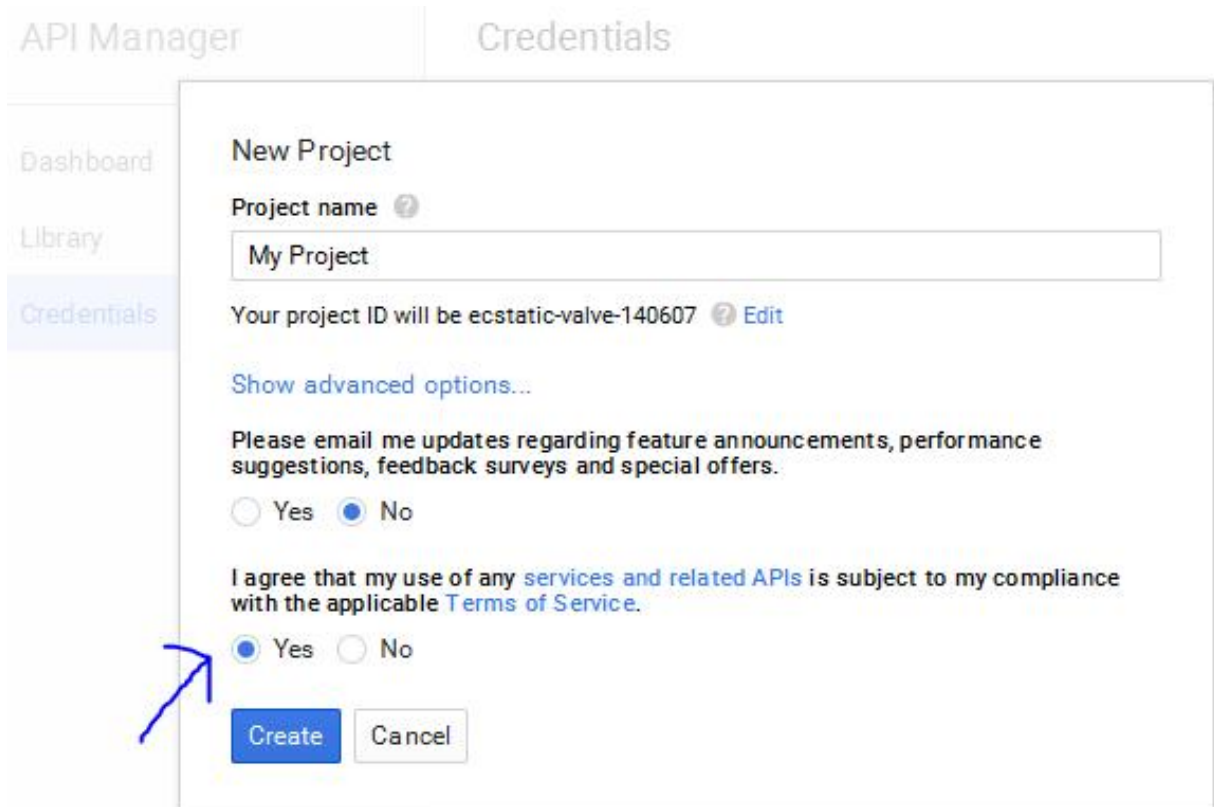
The purpose of GCM Integration is to add Android Push Notifications using Google Cloud Messaging (GCM) feature in an android app.

2. Get Google Project ID and Android Key

As a first step we need to create a Google project and get its key followed by enabling the Cloud Messaging for this project.

2.1. Create a Google Project

Login to your Google account and visit <https://console.developers.google.com/>



The screenshot shows the 'New Project' dialog in the Google Cloud Platform console. The dialog is titled 'New Project' and contains the following elements:

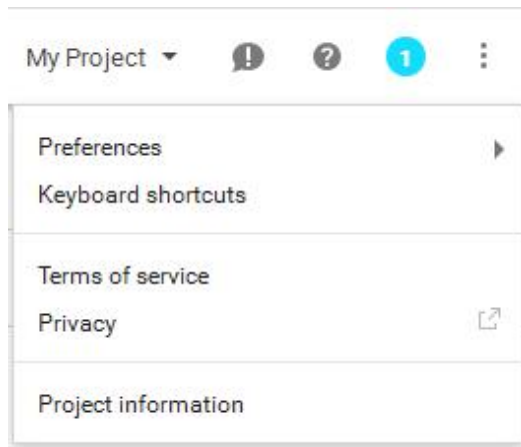
- Project name:** A text input field containing 'My Project'.
- Your project ID will be:** 'ecstatic-valve-140607' with a question mark icon and an 'Edit' link.
- Show advanced options...** A link to expand the dialog.
- Please email me updates regarding feature announcements, performance suggestions, feedback surveys and special offers.** A section with two radio buttons: 'Yes' (unselected) and 'No' (selected).
- I agree that my use of any services and related APIs is subject to my compliance with the applicable Terms of Service.** A section with two radio buttons: 'Yes' (selected) and 'No' (unselected).
- Create** and **Cancel** buttons at the bottom.

A blue arrow points to the 'Yes' radio button for the agreement section.

2.2. Get the Project ID

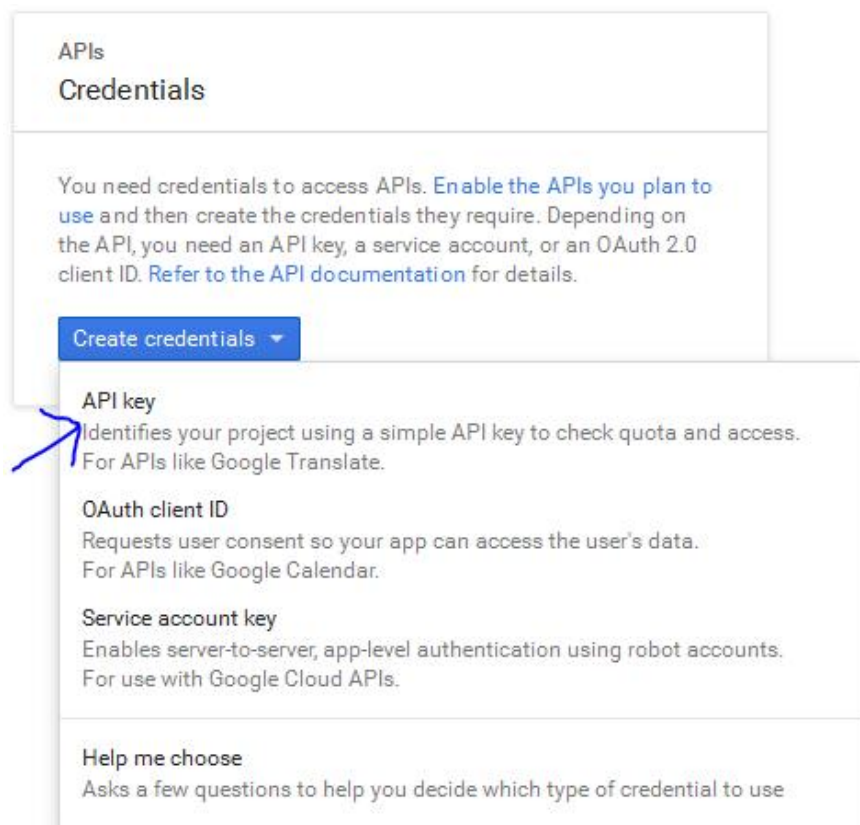
2.2.1. Step 2.1 takes some time and after creation of project we can see your Project name at top bar in Google Console.

Navigate to your Project drop down option > Project Information, and copy the **project id** and keep it for later use.



2.3. Create Android API Key

2.3.1. Visit the Credentials and choose API Key > Android Key



2.3.2. Name the Android API key

Create Android API key

Name

Android key

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
```



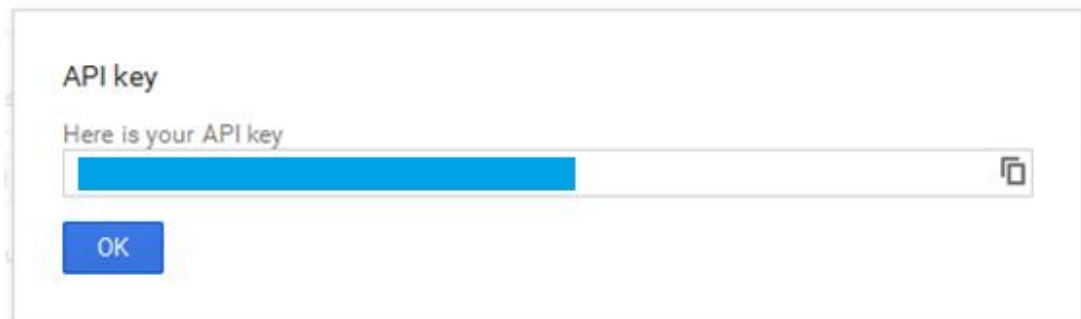
+ Add package name and fingerprint

Note: It may take up to 5 minutes for settings to take effect

Create

Cancel

2.3.3. After Creating Android API Key a new unique Key will be generated. Just copy it for later use.

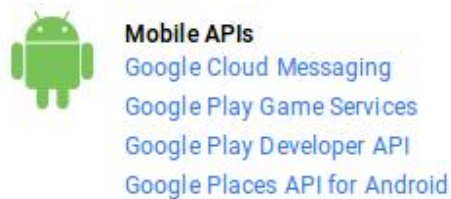


2.4. Enable Google Cloud Message

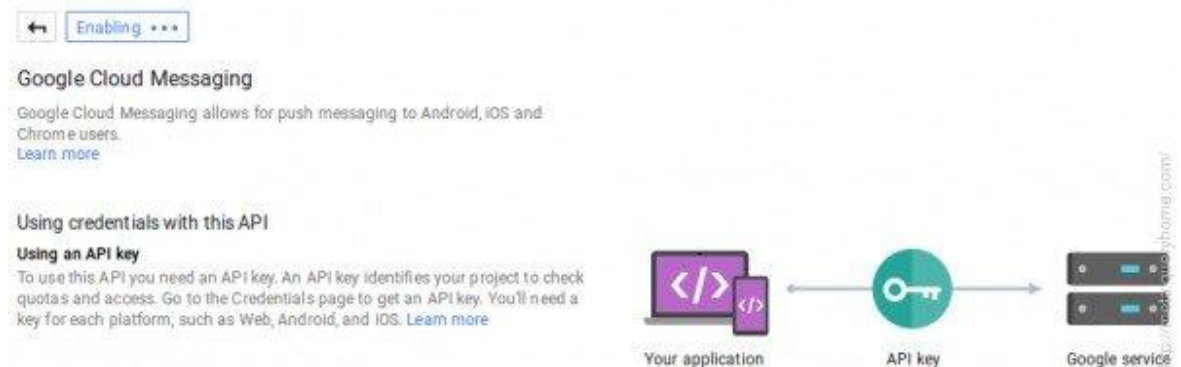
Now you have both Android Key and Google Project ID, the most important part is to enable the GCM for your project.

Click on Overview at Sidebar of the Google console or visit this link <https://console.developers.google.com/apis/library>

Search Google Cloud Message in Search bar. This will be appears as the image bellow



2.4.1. Enabling the GCM - It will take couple of minutes depend upon the Internet speed to enable the functions of GCM.



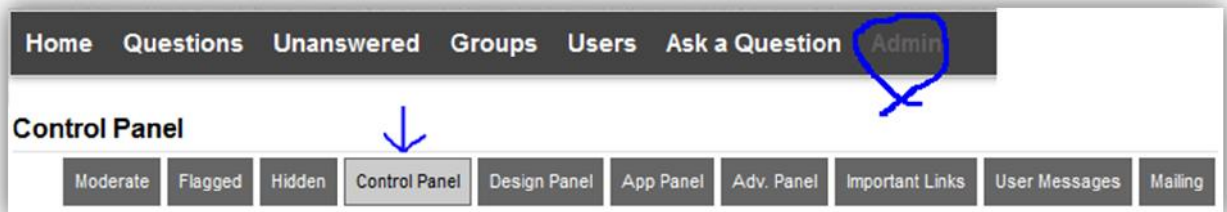
3. GCM Configuration on Community Control Panel

3.1. Add GCM API Key

3.1.1. Login to the community

Login into your community (<http://community.example.com>) as an administrator

3.1.2. Locate the control panel



3.1.3. Add GCM API Key

1. Locate **GCM Configuration** Option
2. Add the required **API Key** in the textbox to receive GCM notification (See 2.3.5)

A screenshot of a form titled 'GCM Configuration'. It contains a label 'GCM API KEY:' followed by a text input field. The input field contains a series of 'x' characters as a placeholder for the API key.

3.1.4. Save the Configuration

Locate the save button at the bottom and press it.

Note: *By default GCM feature is disabled, please send a mail to the support@answercart.com to get it enabled.*

3.2. Get Enterprise Secret Key

Secret key is a method of exchanging information. We may need it in future so copy it.

Secret Key Configuration

A screenshot of a form titled 'Secret Key Configuration'. It contains a label 'Secret Key:' followed by a text input field. The input field contains a series of 'x' characters as a placeholder for the secret key.

4. API Integration in App

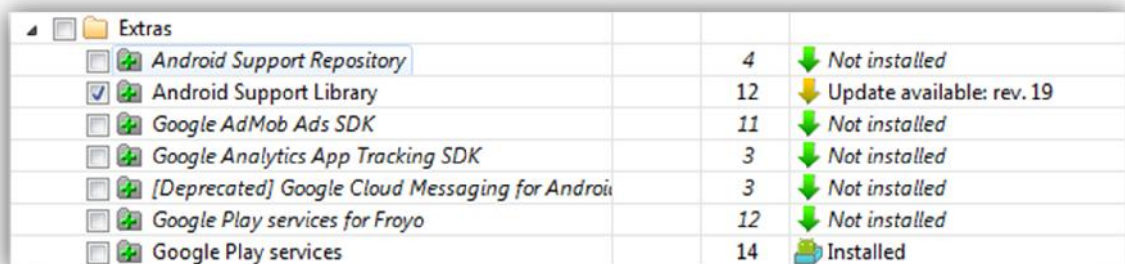
To add GCM notification in existing app, you required two important credential keys - Project ID (see section 2.2.1), Secret Key (see section 3.2).

4.1. Pre-request libraries and setting up the App

Before working with GCM we need to install the helper libraries and make required changes to the android app or emulator we are using.

4.1.1. Install Google Play Services and GCM Libraries

1. Goto android SDK folder > SDK Manager > Extras section
2. Install Google Cloud Messaging for Android Library & android-support-v4 or update your SDK manager to latest version.



Extras			
<input type="checkbox"/>	Android Support Repository	4	Not installed
<input checked="" type="checkbox"/>	Android Support Library	12	Update available: rev. 19
<input type="checkbox"/>	Google AdMob Ads SDK	11	Not installed
<input type="checkbox"/>	Google Analytics App Tracking SDK	3	Not installed
<input type="checkbox"/>	[Deprecated] Google Cloud Messaging for Android	3	Not installed
<input type="checkbox"/>	Google Play services for Froyo	12	Not installed
<input type="checkbox"/>	Google Play services	14	Installed

3. After installing the library it will create **gcm.jar** file in your Android SDK Folder > extras > google > gcm > gcm-client > dist. Later you need to add this **.jar** file to your android project.
4. Now open AVD Manager and create a new Google API emulator and start the emulator.

4.2.Android Manifest Permissions

The following permissions are required to make your project support GCM

AndroidManifest.xml

INTERNET – To make your app use internet services

ACCESS_NETWORK_STATE – To access network state (used to detect internet status)

GET_ACCOUNTS – Required as GCM needs Google account

WAKE_LOCK – Needed if your app needs to wake your device when it sleeps

VIBRATE – Needed if your support vibration when receiving notification

Also add some broadcast receivers as mentioned below

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"

    <!-- GCM connects to Internet Services. -->
    <uses-permission android:name="android.permission.INTERNET" />

    <!-- GCM requires a Google account. -->
    <uses-permission android:name="android.permission.GET_ACCOUNTS" />

    <!-- Keeps the processor from sleeping when a message is received. -->
    <uses-permission android:name="android.permission.WAKE_LOCK" />

    <!-- Creates a custom permission so only this app can receive its messages.
-->
    <permission
        android:name="_____.permission.C2D_MESSAGE" //your activity class
path name
        android:protectionLevel="signature" />
    <uses-permission android:name="____.permission.C2D_MESSAGE" /> //your activity
path name

    <!-- This app has permission to register and receive data message. -->
    <uses-permission
        android:name="com.google.android.c2dm.permission.RECEIVE" />

    <!-- Network State Permissions to detect Internet status -->
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />

    <!-- Permission to vibrate -->
    <uses-permission android:name="android.permission.VIBRATE" />

    <receiver android:name="com.google.android.gcm.GCMBroadcastReceiver"
        android:permission="com.google.android.c2dm.permission.SEND">
        <intent-filter>

            <!-- Receives the actual messages. -->
            <action android:name="com.google.android.c2dm.intent.RECEIVE" />
            <!-- Receives the registration id. -->
            <action
                android:name="com.google.android.c2dm.intent.REGISTRATION" />
            </intent-filter>
        </receiver>
```

4.3. Registration of App to the Community server

4.3.1. Why to inform to the server

Once we are done with the prerequisite of the Android, we need to tell the identity of the app via the means of JSON API as explained in the section 4.3.3 to the server so that server can generate the cloud messages towards this app whenever there is any event.

4.3.2. When to inform to the server

When user login is completed and we have a valid email-id of the user is the point to call the JSON API to tell the server about the client detail.

4.3.3. How to inform to the server

1. JSON API

JSON API is as follows, it requires md5 (key) which we got in the section 3.2, regid which we will see in next section and the email-id of the user. All parameters are compulsory.

[http://community.example.com/json-request/gcm-register?
key=<md5\(key\)>&email=<useremail>®id=XXXXXXXXXX](http://community.example.com/json-request/gcm-register?key=<md5(key)>&email=<useremail>®id=XXXXXXXXXX)

JSON API returns error as a false if call is successful or true if not successful along with the error reason. So just check the JSON Response.

2. How to Get the RegID

- We may need to import GCMRegistrar as follows

```
import com.google.android.gcm.GCMRegistrar;
```

- Initially the GCMRegistrar class need to register with your project id

```
GCMRegistrar.register(this, PROJECT_ID);
```

Here PROJECT_ID is a string and its value is the project id which we got in section 2.2.

- Get the regid as follows

```
final String regId=GCMRegistrar.getRegistrationId(this);
```

5. Integration of Messages Received from the server

After the Step 4 server will start sending the message on every event which has the following structure.

5.1. Notification Structure

Server will broadcast the following 4 information on any notification

- msg: message to the user
- title: Notification title
- url: URL for the notification which should be opened on click of notification
- siteurl: The URL of the website which is generating the message i.e. <http://community.example.com>

5.2. Showing new notification in notification bar

5.2.1. Class which handles all GCM related services – **GCMIntentService**

You may need to import GCMBaseIntentService as follows

```
import com.google.android.gcm.GCMBaseIntentService;
```

GCMIntentService will **extends** **GCMBaseIntentService** class

GCMIntentService.java

Important Functions:

1. **public GCMIntentService():** It's a constructor and will call the super class by sending the Project ID which we got in section 2.2.

```
public GCMIntentService() {  
    //Here PROJECT_ID is a string and its value is the project id.  
    super(SENDER_ID);  
}
```

2. **protected void onMessage():** Called when a new message arrived to device

```
protected void onMessage(Context context, Intent intent) {  
    String message = intent.getExtras().getString("msg");  
    String message_title=intent.getExtras().getString("title");  
    String message_url = intent.getExtras().getString("url");  
    generateNotification(context, message, message_title,  
message_url);  
}
```

3. **public void onError():**- On receiving an error

```
public void onError(Context context, String errorId) {  
    Log.i(TAG, "Received error: " + errorId);  
    displayMessage(context, getString(R.string.gcm_error, errorId));  
}
```

4. **private static void generateNotification():** Method to generate a notification

```
private static void generateNotification  
(Context context, String title, String message, String url)  
{  
    int icon = R.drawable.ic_launcher; // App Icon which can be changed  
    long when = System.currentTimeMillis();  
    NotificationManager notificationManager = (NotificationManager)  
        context.getSystemService(Context.NOTIFICATION_SERVICE);  
  
    Notification notification = new Notification(icon, message, when);  
    Intent notificationIntent = new Intent(Intent.ACTION_VIEW);  
    notificationIntent.setData(Uri.parse(url));  
  
    // set intent so it does not start a new activity  
    notificationIntent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP |  
        Intent.FLAG_ACTIVITY_SINGLE_TOP);  
  
    PendingIntent intent = PendingIntent.getActivity(context, 0,  
        notificationIntent, 0);  
  
    notification.setLatestEventInfo(context, title, message, intent);  
  
    // To open the notification in the home activity of the APP.  
    notification.flags |= Notification.FLAG_AUTO_CANCEL;  
  
    // play default notification sound  
    notification.defaults |= Notification.DEFAULT_SOUND;  
  
    // vibrate if vibration is enabled  
    notification.defaults |= Notification.DEFAULT_VIBRATE;  
    notificationManager.notify(0, notification);  
}
```

6. Un-registration of user device

This step is not required and server handles the un-registration whenever next event is generated towards the device and device is found as unregistered.

Now we are done with our android project and we can run your emulator or android app and test.

7. Testing GCM push Message

Following are the simple steps to test the GCM notifications -

- 1.** Install the APP
- 2.** Go to the community section
- 3.** Login and proceed
- 4.** Ask a question
- 5.** Go to the web version of the community and +svote to this question
- 6.** We should receive the notification and on the click of the notification we should see the question page in the web view.
- 7.** Web view related code is not covered in this document so developer need to take care of the URL opening in the web view as per app needs.