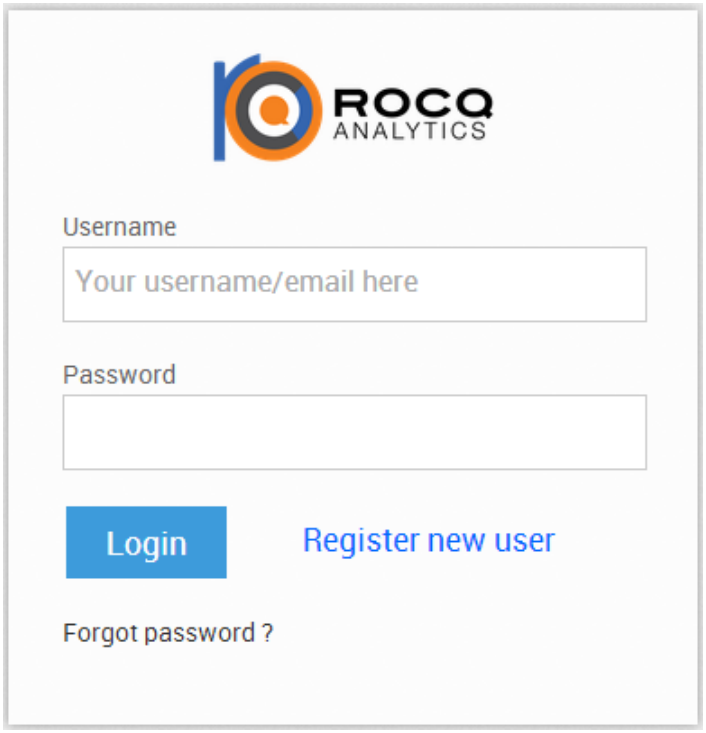
rocQ ANALYTICS

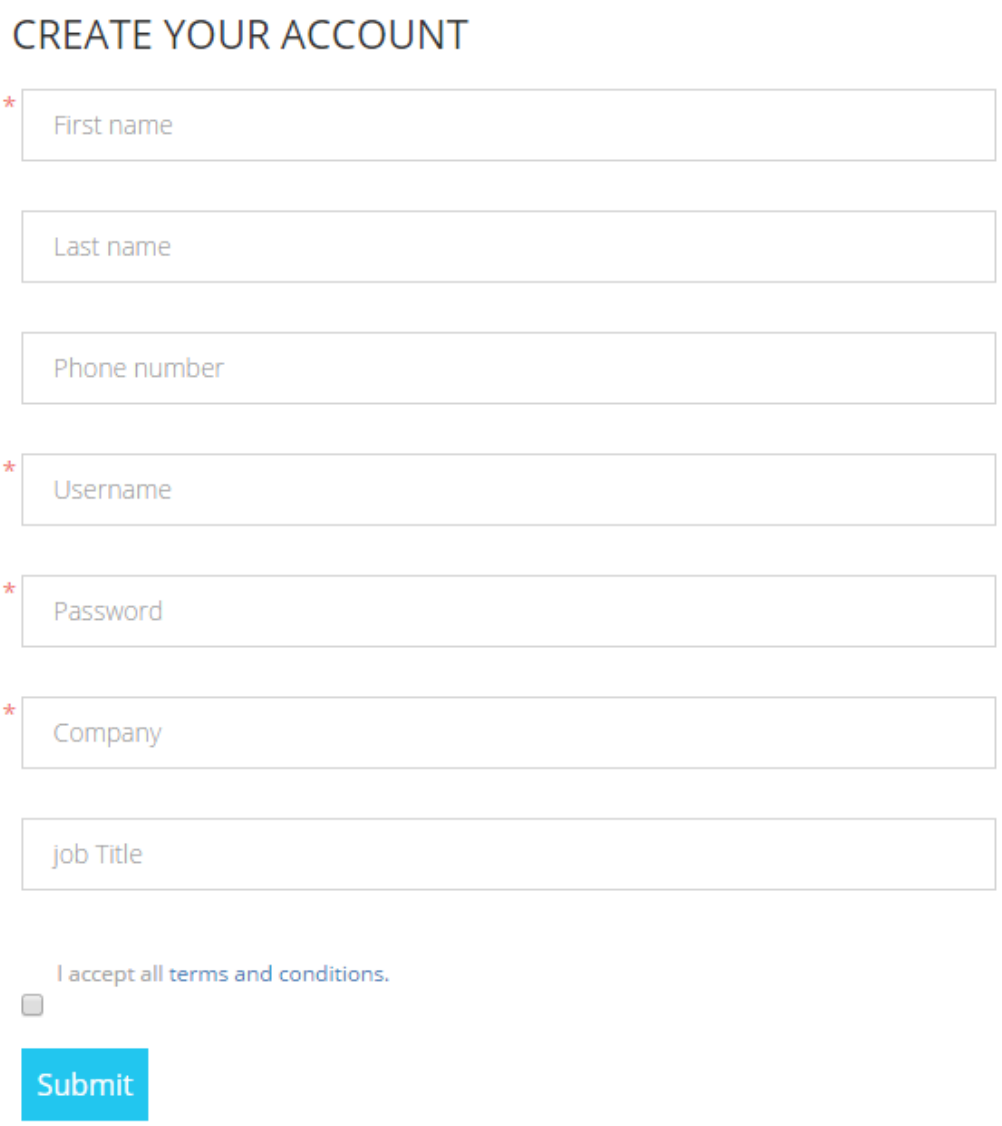
ANDROID API INREGRATION

Step 1: Setting rocQ SDK in an Android Project

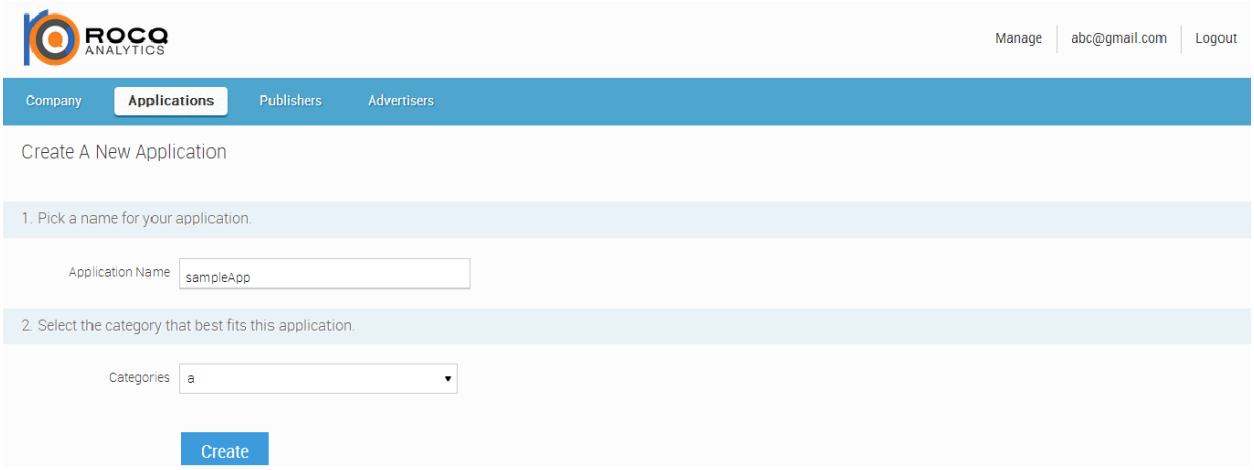
- Go to http://rocq.io/login
- Click on **Register Now** to get yourself registered.



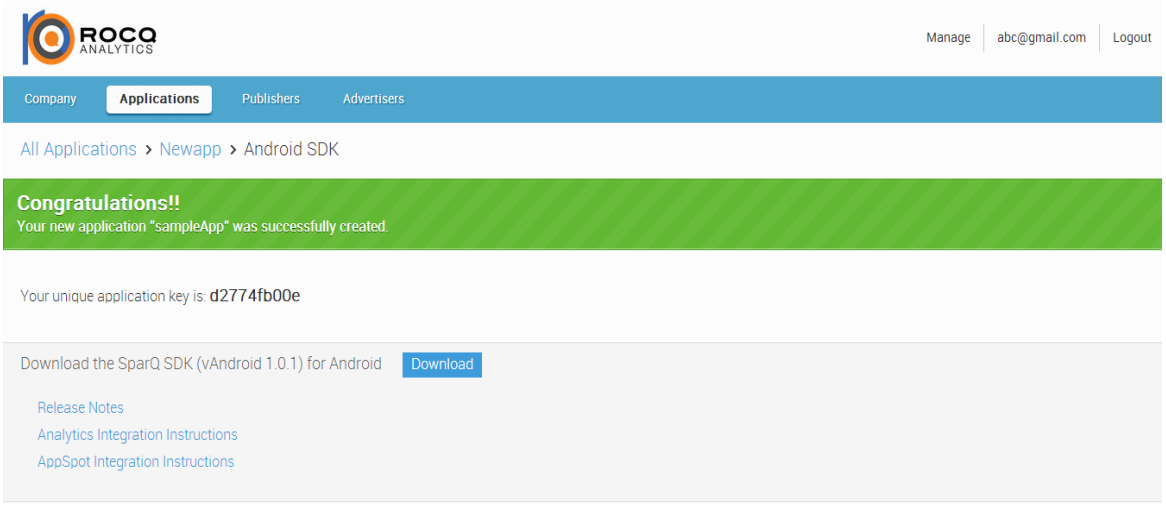
- Fill in all the required credentials and **Submit** to create your account.



- Once you have successfully registered, you will be directed to your first page. Now, click on **Create A New Application**.



- You will be assigned with a Unique Application Key of your app. This key helps rocQ identify your app. Now, download the rocQ SDK from the **Download** button provided at the bottom of the page.



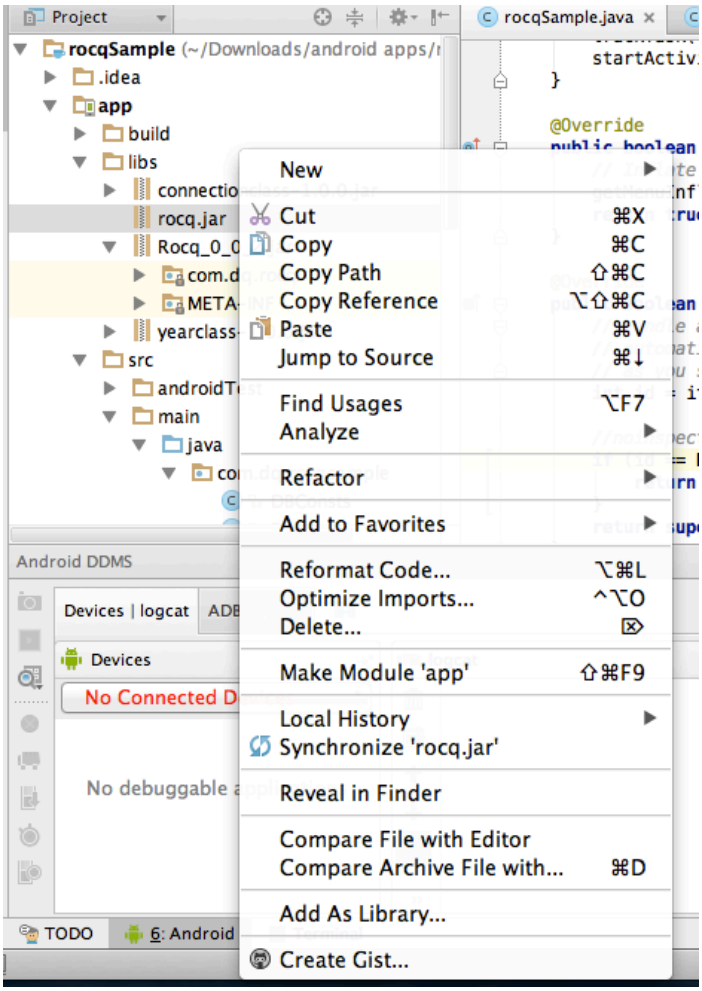
Step 2: Adding rocQ SDK

i) ECLIPSE:

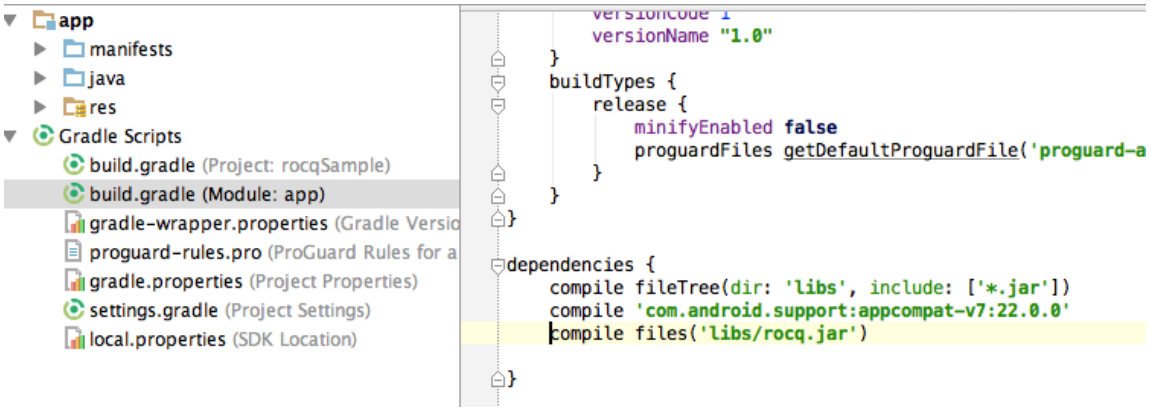
To add the rocq.jar in your project, copy the file from the location you saved it and paste it in **Libs** folder.

ii) ANDROID STUDIO:

- There are two ways to install rocQ SDK in your project –
  1. Copy the rocq.jar from the location you saved it and paste it in **Libs** folder in your project. In android studio, right click on rocq.jar and **Add it as a library**.



2. Add rocq.jar directly to Gradle scripts to compile library.



Step 3: Updating AndroidManifest.xml

Update your **AndroidMainfest.xml** file by adding the following permissions:

1. These are the mandatory permissions to be put into your **AndroidManifest.xml** file.

<!--
This permission is required to allow the application to send events and properties to RocqAnalytics.
-->

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

2. To divide your users in 2G, 3G and Wifi segments, insert these optional permissions into your **AndroidManifest.xml** file.

```
<!--
 This permission is optional but recommended so we can be smart about when to send data.
 -->

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

3. To track the location from where the app is being accessed, installed or uninstalled, add the following permissions in your **AndroidManifest.xml**

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

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

Step 4: Creating your rocq.xml file in project

Add a file called **rocq.xml** to your app's **res/values/** folder:

```
<!--Your Rocq Project secret-->

<string name="rq_appsecret">YOUR_SECRET_KEY</string>
```

(Your Secret Key is the unique application key assigned to your Application.)

Step 5:- Insert the following code in all the activities in onStart() and onStop() methods

Insert the following code in the **onStart()** and **onStop()** methods-

- In onStart() method

```
RocqAnalytics.startScreen(this);
```

- In onStop() method

```
RocqAnalytics.stopScreen(this);
```

Example:

```
@Override
protected void onStart() {
// TODO Auto-generated method stub
super.onStart();
RocqAnalytics.startScreen(this);
}
@Override
protected void onStop() {
// TODO Auto-generated method stub
super.onStop();
```

```
RocqAnalytics.stopScreen(this);
}
```

Step 6:- Tracking Events

To track the events, pass the event name, properties in key-value pairs, and the position of event on the screen. Before calling the event tracking method, initialize the rocQ SDK by calling its **initialize()** method. Events can be tracked in two ways; either using position or view.

```
RocqAnalytics.initialize(this);
RocqAnalytics.trackEvent("<EventName>",newActionProperties("<key>",
"<value>","<key>","<value>","<key>","<value>"), Position);
```

Parameters:

- **String <Event Name> :** It is event name in human-readable format such as “Buy Button” or “news list item”.
- **ActionProperties <key>,<value> (optional)**: This key-value pair would

consist of properties of an event, for example <item>,<bag> or <type>,<read>. There can be N numbers of parameters.

- **Position (optional):** You can pass the position of an item where it is placed on the screen. It will be helpful in finding its relevant position. (Position.TOP, Position.LEFT, Position.RIGHT, Position.BOTTOM, Position.CENTER).
- **View (optional):** You can pass the view of activity to get the exact position of the event on the screen.

Example(Sending optional position):

```
RocqAnalytics.initialize(this);

RocqAnalytics.trackEvent("Buy click", new ActionProperties("position","store screen","where","top"), Position.LEFT);
```

Step 7:- Tracking Screens

You can track the screen by their names and parameters, whenever required.

```
RocqAnalytics.initialize(this);

RocqAnalytics.trackScreen("<Screen Name>", new ActionProperties("<key>","<value>","<key>","<value>","<key>","<value>"));
```

To track the screen, pass Screen name, and properties in key-value pairs. Before calling the screen tracking method, initialize the rocQ SDK by calling its **initialize()** method.

Parameters:

- **String <Screen Name>:** It is screen name in human-readable format such as “Buy Button” or “news list item”.
- **ActionProperties <key>,<value> (optional)**: The key-value pair would consist of properties of screens, for example <items>,<10> or <movie>,<krish>. There can be N numbers of parameters.

Example:

```
RocqAnalytics.initialize(this);

RocqAnalytics.trackScreen("News Screen", new
ActionProperties("content","text","title","news","author","htmedia"));
```

Step 8:- Tracking User Identity/Information

You can track the users by their username, email, age and other parameters. To track the user, pass user name, and properties in key-value pairs. Before calling the identity tracking method, initialize the rocQ SDK by calling it’s **initialize()** method.

```
RocqAnalytics.initialize(this);

RocqAnalytics.identity("<user-name>", new
ActionProperties("<key>","<value>","<key>","<value>"));
```

Parameters:

- **String <User-Name>:** It is user name in human-readable format such as

"abhi23" or "jai@gmail.com".

- **ActionProperties <key>,<value> (optional)** : It carries the properties of the user in key-value pairs example <age>,<26> or <plan>,<trial>. There can be N numbers of parameters.

Example:

```
RocqAnalytics.initialize(this);

RocqAnalytics.identity("shashank734", new
ActionProperties("Email","shashank.agarwal@hindustantimes.com","Age","24","c
ity","gurgaon"));
```

Congratulations!! You are now done with standard implementation of rocQ Analytics SDK. You can also check advance configuration of analytics.

Enabling Push Notification and Uninstallation

This document will guide you to send and receive push notifications in your Android app. Enabling push notifications with rocQ Analytics is quite easy. Let's have a look.

In case you are not registering push notification through rocQ, send the Registration Id to our rocQ server, which involves otherwise move to

Step 1:- Enabling Google Cloud Messaging in Google API Console

Setting Push Registration ID

```
RocqAnalytics.initialize(this);

RocqAnalytics.setPushtRegistrationId(regId, context);
```

*where, **regId** is the ID which you got while registering to GCM and, **context** can be **this**.*

Example,

RocqAnalytics.setPushtRegistrationId(String xxxxxxxxxxxxxxxxxx,Context this);

Go to **GCMIntentService** class that contains method **onHandleInent** or **onMessage** and insert that code within the following condition:

```
if(!new

RocqGcmIntentService().handleRocqMessage(intent,getApplicationContext()))
{
//Insert your code here
}
```

Or, if you're using **GCMListenerService** class that contains method **onMessageReceived,** insert that code within the following condition:

```
if(!new
RocqGcmIntentService().handleRocqMessage(bundle,getApplicationContext())
)
{
//Insert your code here
}
```

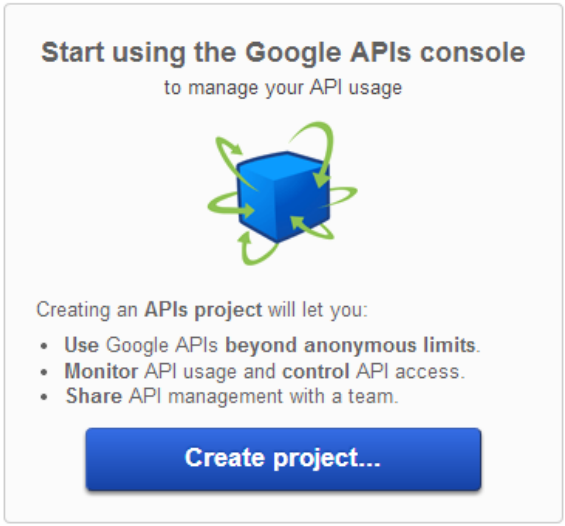
Step 1:- Enabling Google Cloud Messaging in Google API Console

In order to enable push notification, enable Google Cloud Messaging (GCM) for Android from Google's API Console Page. In case you already have a GCM sender ID, update your rocq.xml file by adding it in the file using the below given code.

```
<!--Your Rocq GCM PUSH Sender Id -->
<string name="rq_gcmSenderId">XXXXXXXXXXXX</string>
```

However, if you don't have a GCM sender ID, generate it by following these steps:

- Sign In with your Google account credentials. However, if you don't have a project yet, create one.

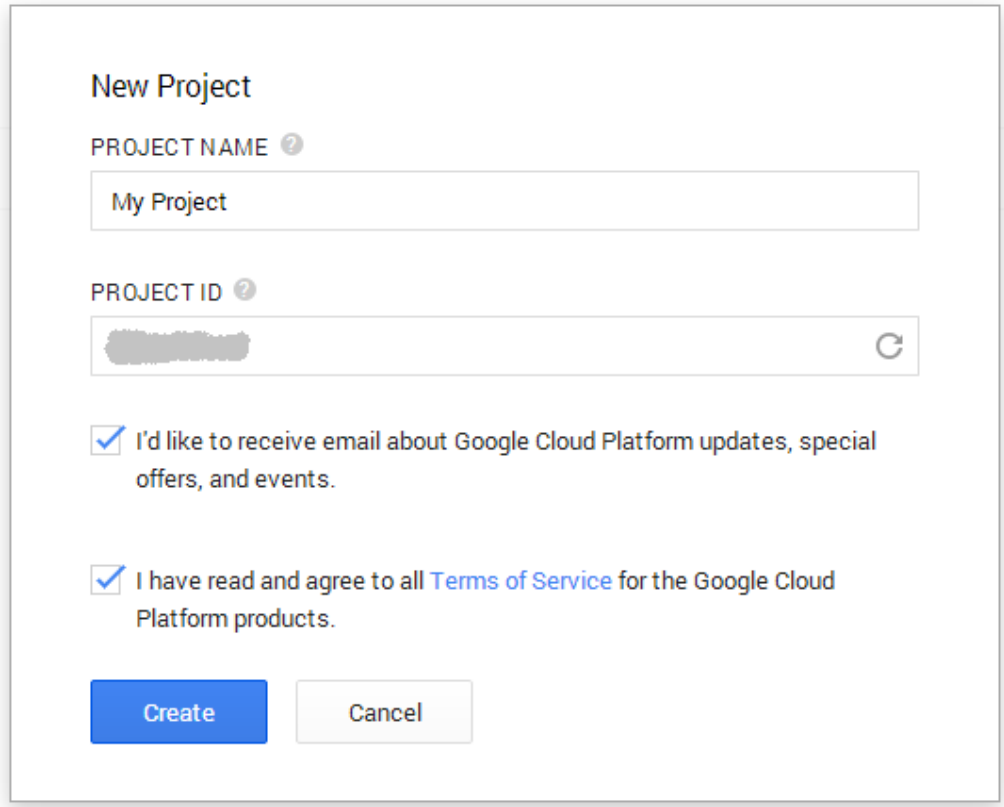


- Click on **Create Project**. If you already have a project, you will be taken directly to the console's dashboard.
- Click on **Project** in left topmost menu and click on **Create Project**.
- Note the browser URL. It should resemble **https://code.google.com/apis/console/#project:XXXXXXXXXXXX**. The **XXXXXXXXXXXX** will be your twelve digit Google Sender ID, which you will need to use in your code to register your application for push notifications.
- You need to update your rocq.xml file by adding GCM sender ID in the file. The code is as follows:-

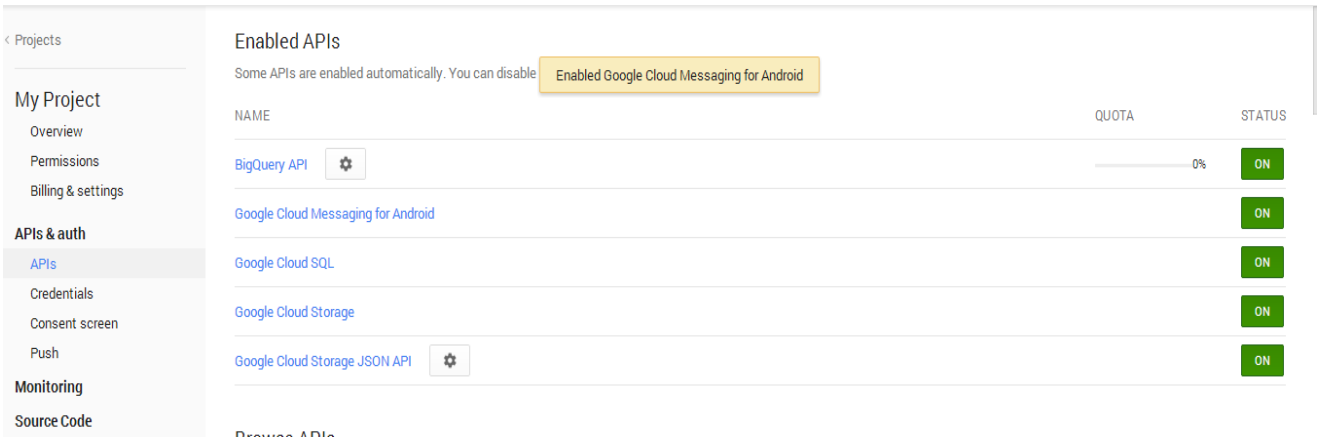
```
<!--Your Rocq GCM PUSH Sender Id -->
<string name="rq_gcmSenderId">XXXXXXXXXXXX</string>
```

XXXXXXXXXXXX will be your own GCM Sender ID.

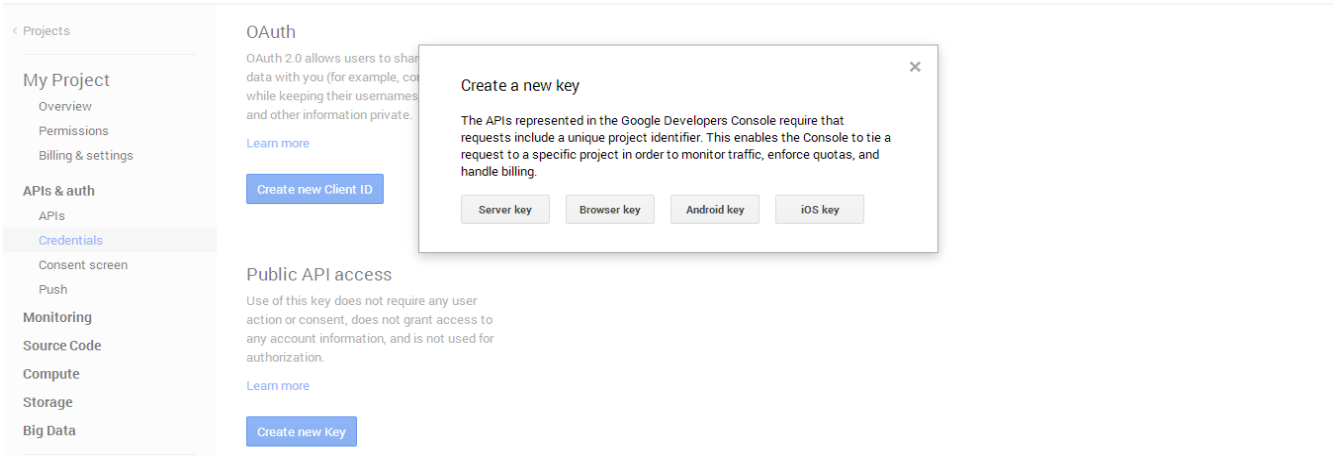
- Write your Project Name, and you will automatically be given a Project ID. Check all the terms and conditions before clicking on **Create**.



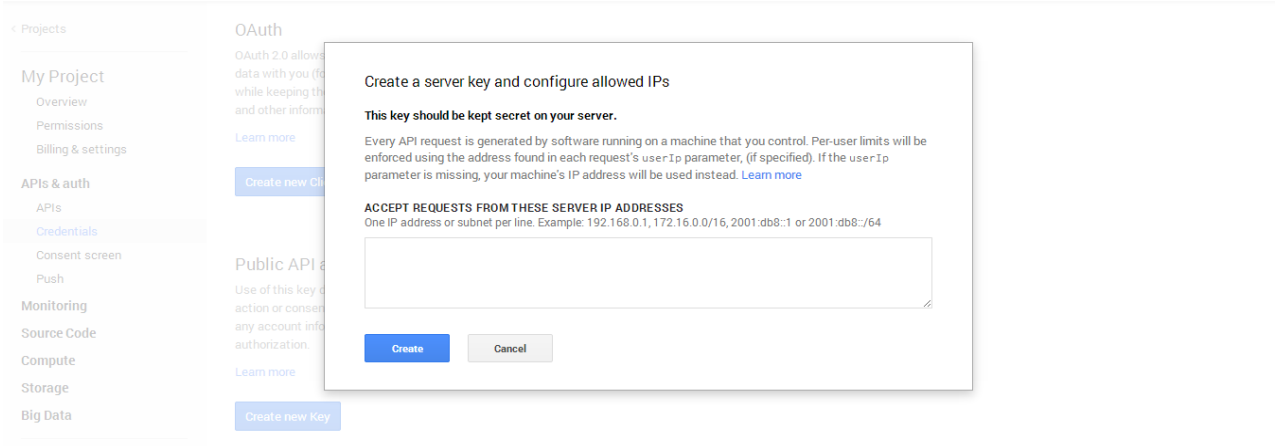
- Click on **API's** in the left menu and locate **Google Cloud Messaging for Android** in the list of available services. Click the **ON** toggle. Accept the terms and conditions.



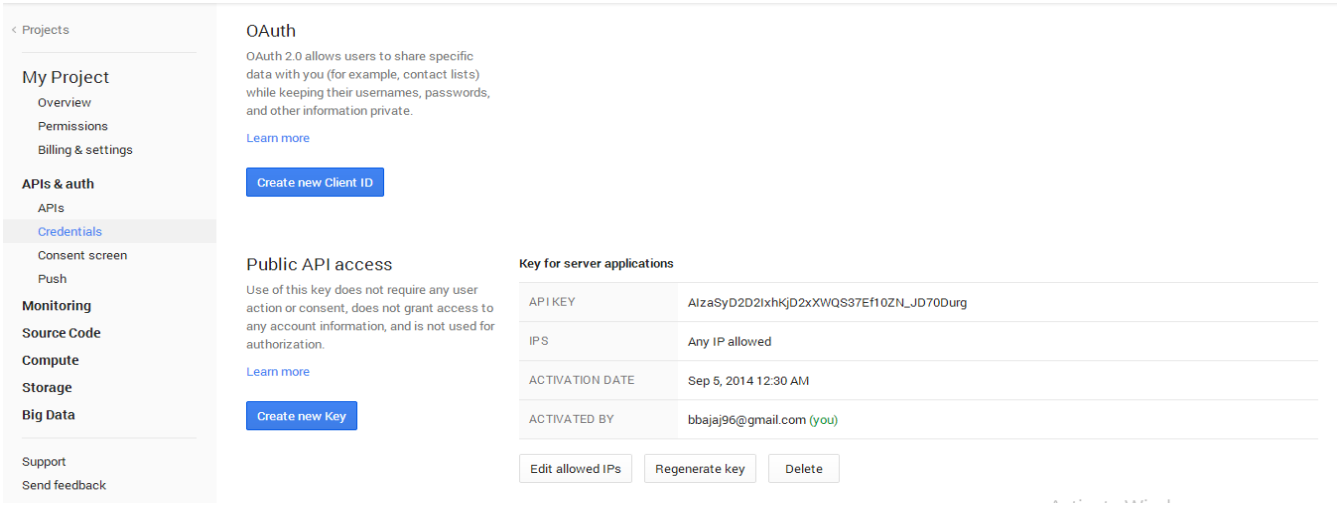
- Click **Credentials**, followed by **Create New Key**



- In **Create A New Key**, click **Server Key**. In the next window, click on **Create**.



- Note down the API key for future use.



Step 2:- Updating your AndroidManifest.xml

To send and receive push notifications, you are required to add certain permissions to your **AndroidManifest.xml** file.

```
<permission
android:name="YOUR_PACKAGE_NAME.permission.C2D_MESSAGE
" android:protectionLevel="signature"/>

<uses-permission android:name="
YOUR_PACKAGE_NAME.permission.C2D_MESSAGE"/>

<uses-permission
android:name="com.google.android.c2dm.permission.RECEIVE" />

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

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

permissions.

Example,

You will need to set up rocQ library class as a receiver for GCM intents:

<permission ndroid:name="com.example.mypack.permission.C2D_MESSAGE" android:protectionLevel="signature"/>

<receiver android:name="com.dq.rocq.push.RocqGcmReceiver" android:permission="com.google.android.c2dm.permission.SEND"><intent-filter>

<action android:name="com.google.android.c2dm.intent.RECEIVE"/>

<action android:name="com.google.android.c2dm.intent.REGISTRATION"/>

<category android:name=" com.example.mypack "/></intent-filter></receiver>

<service android:name="com.dq.rocq.push.RocqGcmIntentService"/>

Also, you are required to add:-

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

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

Step 3:- Setting Up Your App to Receive Notifications

In order to set up your app to receive notifications, insert the following code in **onCreate()** of **MainActivity** (Launcher Activity). If you have sent any extra parameters in your push notifications, you will receive them as key-value pairs.

```
RocqAnalytics.initialize(this);

RocqAnalytics.registerPush(this);
```

Step 4: - You can specify a custom notification icon to display in the notification bar by adding the following code in your manifest file.

```
<meta-data
        android:name="rocq_notofication_icon"

        android:value="CUSTOM NOTIFICATION ICON" />
```

To determine the exact number of push notifications that have been clicked on, ensure to put the given codes in **onStart()** and **onStop()** methods respectively in your Launcher activity.

- In onStart() method

```
RocqAnalytics.startScreen(this);
```

- In onStop() method

```
RocqAnalytics.stopScreen(this);
```

rocQ Campaign Receiver

rocQ Campaign Receiver allows you to keep a track of traffic sources of your app, thus improving the value of your marketing channels.

To track the source of installation of your app, insert the following code to your **AndroidManifest.xml** file.

```
<receiver android:name="com.dq.rocq.RocqCampaignReceiver"
android:exported="true"><intent-filter>

<action
android:name="com.android.vending.INSTALL_REFERRER"/></intent-
filter></receiver>
```

In case, you have your own Campaign Receiver then you have to pass the **intent** in our Campaign Receiver that you have received from **onReceive()** method in **MyReceiver** class.

MyReceiver class is as follows:

```
public class MyReceiver extends BroadcastReceiver {
@Override
 public void onReceive(Context context, Intent intent) {
RocqAnalytics.initialize(this);
RocqCampaignReceiver.trackCampaign(Intent intent, Context ctx); }}
```

Getting Push Registration ID

```
RocqAnalytics.initialize(this);
RocqAnalytics.getPushRegistrationId(Context context);
```

Enabling Deep Linking for your App Content

Deep Linking allows your app to be launched directly via link or search results or push notification. It provides a unique link to every activity of your app.

Example URI: rq://rocq.io/history?id=modern#today

Rq	*Scheme Name (Required)*
rocq.io	*Authority (Optional)*
History	*Path (Optional)*
id=modern	*Query (Optional)*
Today	*Fragment (Optional)*

In order to provide a link to your activity, edit your **AndroidManifest.xml**, where a particular activity is declared. This is to be declared in the **<intent-filter>** tag which is child of **<activity>** tag. Following lines are to be inserted.

```
<activity android:name=".Activity_Name">
  <intent-filter>
    <action android:name="android.intent.action.VIEW"/>
    <category android:name="android.intent.category.DEFAULT"/>
    <category android:name="android.intent.category.BROWSABLE"/>
    <data android:scheme="rq" android:host="app_name"/>
  </intent-filter>
</activity>
```

Read Data from Incoming Events

If you want to read the data of incoming events, add the following code in OnCreate() method of your respective activity.

```
Intent intent = getIntent();
String action = intent.getAction();
Uri data = intent.getData();
```

Test Deep Link

Go to the **adb** folder in your eclipse bundle and open **Command Prompt**. Type the following command-

```
$ adb shell am start
-W -a android.intent.action.VIEW
-d <URI> <PACKAGE>
```

Sending Android in-app notifications

1. Declare **RocqInappMessageActivity** in your AndroidManifest.xml file

```
<activity

android:name="com.dq.rocq.inapp.RocqInappMessageActivity"
        android:screenOrientation="portrait"

android:theme="@android:style/Theme.Translucent.NoTitleBar.Fullscreen" >
      </activity>
```

2. Insert this code in the **AndroidManifest.xml** file into your application tag.

```
<meta-data

        android:name="rocq_homescreen"

        android:value="Your Home Activity" />
```

where **YOUR HOME ACTIVITY** is the path to your HomeActivity.

For Example, "com.rocqanalyticsdemo.MainActivity"

Note: If you have a SplashScreen, HomeActivity refers to the screen succeeding it.

Congratulations! You've successfully set up Android in-app notifications.

Advanced rocq.xml parameters

Here, is the complete **rocq.xml** file:

```
<?xml version="1.0" encoding="UTF-8"?>
<resources>
<!--Your Rocq Project secret-->
<string name="rq_appsecret">xxxxxxxxxx</string>
<!--Your Rocq GCM PUSH Sender Id -->
<string name="rq_gcmSenderId">xxxxxxxxxxxx</string>
<!--The maximum amount of analytics events to queue before dropping to conserve memory-->
<integer name="rq_maxQueueSize">10000</integer>
<!--The maximum amount of analytics events to queue before dropping to conserve memory-->
<string name="rq_debugMode">true</string>
<!--Flush to the server after receiving this many messages-->
<integer name="rq_sendAt">20</integer>
<!--Flush on a timer after this many milliseconds-->
<integer name="rq_sendAfter">30000</integer>
<!--Refresh the sparq integration settings after this many milliseconds -->
<integer name="rq_sessionTimeout">30000</integer></resources>
```

Parameter	Type	Purpose
secret_key	String	The secret key is the key to the application, string type is mandatory.
session_timeout	Integer	The duration for which your application can stay in the background before the session is ended. Defaults to 30 seconds or 30000 milliseconds.
debug_key	Boolean	Flag to enable or writing debug information in the log. Contains information about the key. The value could be true or false.
flush_at_key	Integer	The number of events after which the events are flushed. Defaults to 20.
flush_after_key	Integer	The time interval after which the events are flushed. Defaults to 30 seconds or 30000 milliseconds.
max_queue_size_key	Integer	The maximum size of the queue is signified. Defaults to 10000.