Plumb5 Analytics SDK for Android Devices

The document details steps required to integrate Plumb5 Analytics SDK into your application for Android devices. For any questions or queries please email us at support@plumb5.com

Download Plumb5 SDK.

The Plumb5 sdk is available for download at www.plumb5.com/Plumb5AndroidSDK.zip

SDK Integration

To get up and running with Plumb5 on Android, there a couple of steps to get you there.

Android Studio:

- Step 1 : Select File → New →New Module
- Step 2 : Select " Import .JAR or .AAR Package under More Modules.
- Step 3 : Select the plugin-release.aar by clicking on the File chooser.
- Step 4 : Select Finish.
- Step 5 : Right click on app folder → Open Module Settings.
- Step 6 : Select Dependency at the top right corner.
- Step 7 : Select '+' symbol and choose 'Module Dependency' and It will list down the existing jar files.
- Step 8 : Select plugin-release.aar

Eclipse

- Step 1 : Add plugin-release.aar to your project's libs folder.
- Step 2 : Right-click the JAR file and select Build Path → Add to Build Path.

SDK Configuration and Permissions

Register with www.plumb5.com to get APP KEY on the Plumb5 dashboard and add it to the manifest file as shown below.

XML

```xml
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.WAKE_LOCK" />
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<uses-permission android:name="android.permission.SEND_SMS" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED" />
<uses-permission
android:name="com.google.android.providers.gsf.permission.READ_GSERVICES"/>
<uses-permission android:name="android.permission.BLUETOOTH" />
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN"/>
<uses-permission android:name="android.permission.CALL_PHONE" />
<uses-permission android:name="com.google.android.c2dm.permission.RECEIVE" />
```

Runtime permission required for ACCESS_FINE_lOCATION and ACCESS_COARSE_LOCATION ,if you are using Android marshmallow and above.

Modify AndroidManifest.xml

//change application name for tracking.

```xml
android:name="com.plugin.plumb5.P5Monitor"
//Add plumb5 Receiver
<receiver android:name="com.plugin.plumb5.P5Receiver"
   android:exported="false"
   android:permission="android.intent.action.BOOT_COMPLETED">
   <intent-filter>
       <action android:name="android.intent.action.BOOT_COMPLETED" />
   </intent-filter>
</receiver>
```

Add Plumb5 Dependencies

Plumb5 Android SDK is hosted on jcenter maven repository.

Update your project's build.gradle script to include jcenter.

Groovy

```
repositories {
        ...

 jcenter()

 ...
}
```

Include Plumb5 dependencies in your app/build.gradle.

Groovy

```
dependencies {

        compile fileTree(dir: 'libs', include: ['*.jar'])
        compile 'com.android.support:appcompat-v7:25.3.1'
        compile project(':plugin-release')
        compile 'com.google.android.gms:play-services-location:9.2.1'
        compile 'com.google.android.gms:play-services-places:9.2.1'
        compile 'com.google.code.gson:gson:2.3'
        compile 'com.google.android.gms:play-services-wearable:9.2.1'

     }
```

SDK Initialization

First Import plumb5 plugin:- import com.plugin.plumb5.P5Engine;

Initialise the SDK from the onCreate method of your Application class and start Session.

```
P5Engine.p5Init(this, "p5m1a2i3sdk1464", CommunicationMode.GCM);
```

Example:

Java

```
//Pass the APP KEY and CommunicationMode.GCM or .FCM(Currently using) to P5Engine.p5Init method to start
tracking the user actions .
public class MyApplication extends Application {
  @Override
  public void onCreate() {

    super.onCreate();
    P5Engine.p5Init(getApplicationContext(), AppKey, CommunicationMode.GCM);
  }
}

Or

public class MyApplication extends AppCompatActivity
{
  @Override
  public void onCreate(Bundle savedInstanceState)
{
    super.onCreate(savedInstanceState);
    P5Engine.p5Init(this, AppKey, CommunicationMode.GCM);

  }
}
```

```
*this,getApplicationContext() identify corrent context of class or Activity.
```

The client can also add additional properties like user information.

Java

```
P5Engine.p5SetUserInfo(this, "test", "test@gmail.com", "9916193666", null);
```

Track Event

Event tracking lets you track user activities in your application and tie them to the corresponding engagement campaigns or to trigger an in-app notification.

For Event tracking only one method in this API
P5Engine.P5Eventdata

Java

```
P5Engine.P5Eventdata(this, "BUTTON_CLICK", "btn-name", “value”);
```

Example:
P5Engine.P5Eventdata(this,”BUTTON_CLICK”,”Cart”,”2”)

Transaction

Transaction API allow you to pass business events to Plumb5 for analytics., user segmentation or to trigger engagements or other actions. All passed events are captured in the Event analytics screen on the dashboard.

Every event has 2 attributes, action name and key, value pairs which represent additional information about the action. Add all the additional information which you think would be useful for segmentation while creating campaigns. For eg. the following code tracks a purchase event of a product. We are including attributes like ProductId, ProductName, Quantity, TotalAmount, ProductImage, OrderId, TransStatus which describe the event we are tracking.

For Transaction only one method in the API
P5Engine.P5Transdata

Java

```
P5Engine.P5Transdata(this, [TransData]);
```

this- context
[TransData]-JSONArray

Java

```
JSONArray jsonTransaction = new JSONArray();

try {

   final JSONObject jsonItem = new JSONObject();

   jsonItem.put("ProductId", 1);

   jsonItem.put("ProductName", "samsung");

   jsonItem.put("Quantity", 1);
```

```
    jsonItem.put("TotalAmount", 200);

    jsonItem.put("ProductImage", "");

    jsonItem.put("OrderId", "234");

    jsonItem.put("TransStatus", 0);

    jsonTransaction .put(jsonItem);

} catch (JSONException ignored) {}

P5Engine.P5Transdata(this, jsonTransaction );
```

Static Form

If client have there own captcha form then they can add there from through this Method API.

Java
P5Engine.P5PostStaticFormData(this, FormId,[DataList]);

this- context
FormId- captcha form Id.
[DataList]- JSONArray

```
Java

JSONArray DataList =new JSONArray();
DataList.put("sunny");
DataList.put("sunny@gmail.com");
DataList.put("991619345");
DataList.put("Male");
DataList.put("Bangalore");
DataList.put("Ready to buy"); }
P5Engine.P5PostStaticFormData(this, FormId, DataList);
```

Engagement / Push

WebEngage SDK supports both FCM or GCM for push messaging. To use FCM follow our Firebase Cloud Messaging Integration guide. Otherwise, follow our Google Cloud Messaging Integration guide.

Firebase Cloud Messaging (FCM) integration

1. Add Firebase to your project
Follow the official Firebase instruction on how to add Firebase to your existing project.

2. Add FCM dependency to your build.gradle file

Update Plumb5 dependency in your project's top level project/build.gradle file

Groovy

```
classpath 'com.google.gms:google-services:3.1.0'
```

Include Plumb5 dependencies in your app/build.gradle.

Groovy

```
compile 'com.google.firebase:firebase-auth:9.2.1'
compile 'com.google.firebase:firebase-core:9.2.1'
compile 'com.google.firebase:firebase-messaging:9.2.1'
apply plugin: 'com.google.gms.google-services'
```

dependency `com.google.gms:google-services` is doc file which have information about firebase project_number,project_id, package_name of your application. Which you have created at the time of project creation from firebase. And that doc file you have to add in your app build file.

3. Add P5FireBaseManager and P5FireBasePushNotificationService inside application tag of your AndroidManifest.xml as shown below

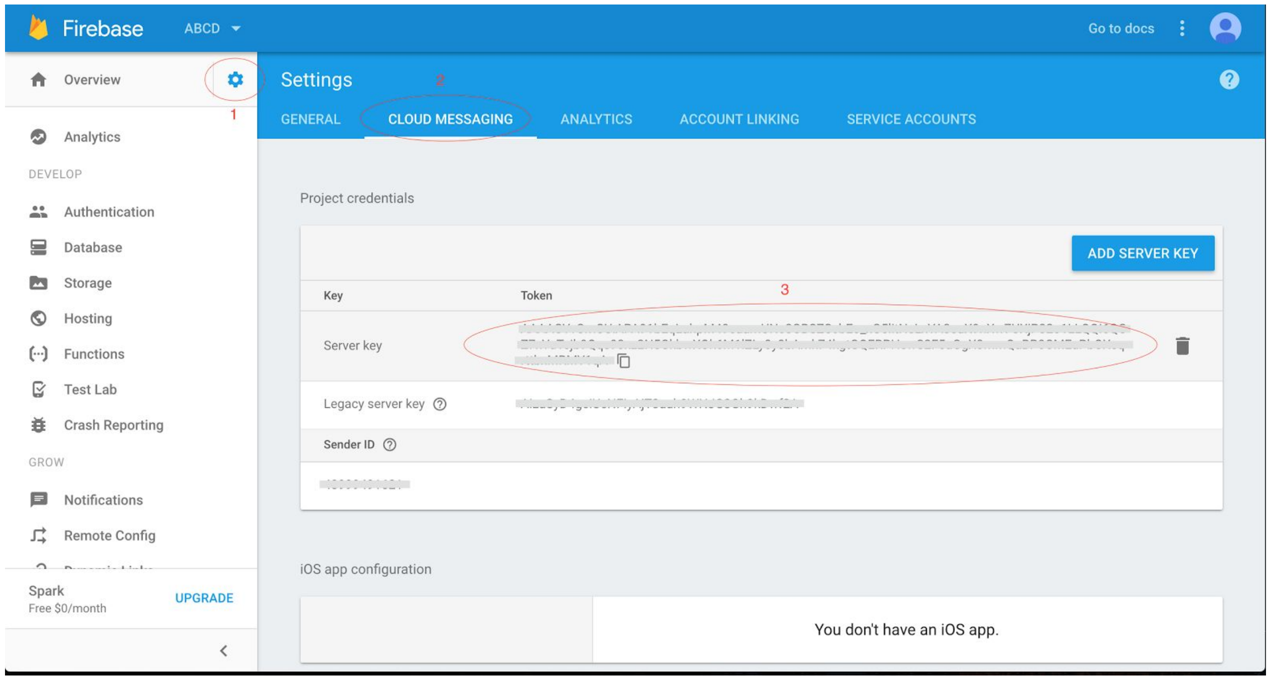
XML

```
<service
   android:name="com.plugin.plumb5.P5FireBaseManager"
   android:exported="true">
```

```
    <intent-filter>
        <action android:name="com.google.firebase.INSTANCE_ID_EVENT" />
    </intent-filter>
</service>

<service
    android:name="com.plugin.plumb5.P5FireBasePushNotificationService"
    android:exported="true">
    <intent-filter>
        <action android:name="com.google.firebase.MESSAGING_EVENT" />
    </intent-filter>
</service>
```

4. Login to your Firebase console and go to Project Settings -> Cloud Messaging and copy your server key and Sender ID as shown below.



And please provide us this server key and Sender ID.

Google Cloud Messaging (GCM) Integration

Enable GCM for your project.

- Visit Google developers page

- Create a new app or select an existing app and package name and click Choose and configure services.
- Click the Cloud Messaging button.
- Note down the Server API Key and Sender ID and povide us.
- Add GCM dependency to your app/build.gradle file

Groovy

```
compile 'com.google.android.gms:play-services-gcm:9.2.1'
```

- Add permissions in AndroidManifest.xml file

XML

```
<uses-permission android:name="com.google.android.c2dm.permission.RECEIVE" />
<permission
 android:name="YOUR.PACKAGE.NAME.permission.C2D_MESSAGE"
 android:protectionLevel="signature" />
<uses-permission android:name="YOUR.PACKAGE.NAME.permission.C2D_MESSAGE" />
```

- Add P5GcmPushNotificationService inside application tag of your AndroidManifest.xml as shown below

XML

```
<receiver
   android:name="com.google.android.gms.gcm.GcmReceiver"
   android:exported="true"
   android:permission="com.google.android.c2dm.permission.SEND">
   <intent-filter>
       <action android:name="com.google.android.c2dm.intent.RECEIVE" />
       <category android:name="YOUR.PACKAGE.NAME" />
   </intent-filter>
</receiver>

<service
   android:name="com.plugin.plumb5.P5GcmPushNotificationService"
   android:exported="false">
   <intent-filter>
       <action android:name="com.google.android.c2dm.intent.RECEIVE" />
   </intent-filter>
</service>
```

Beacon

Beacons are permanently sending out their unique identifier to notify nearby smartphones of their existence, hence the name beacons. A compatible app running on the smartphone then knows that it is in a close proximity. Depending on the application, this can trigger an action on the smartphone, e.g. displaying some information about nearby items.

Android Smartphone Requirements:

- Android > 4.3
- Bluetooth Smart radio unit

The ID consists of three parts:

- UUID (organization or company)
- major (arbitrarily, e.g. specific chain store)
- minor (e.g. location in store)

Open the “build.gradle (Module: app)” Gradle Script and declare Estimote SDK dependency:

Groovy

```groovy
compile project(':estimote-sdk')
```

Add Runtime Permission in your java class. (and also things like, is Bluetooth on, is Location on, etc.)

Java

```java
if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) { String[] permission = {
"android.permission.ACCESS_FINE_LOCATION"};
ActivityCompat.requestPermissions(this, permission, 1); }
```

Geofence

Geofencing combines awareness of the user's current location with awareness of the user's proximity to locations that may be of interest. To mark a location of interest, you specify its latitude and longitude. To adjust the proximity for the location, you add a radius. The latitude, longitude, and radius define a geofence, creating a circular area, or fence, around the location of interest.

- Add permissions in AndroidManifest.xml file

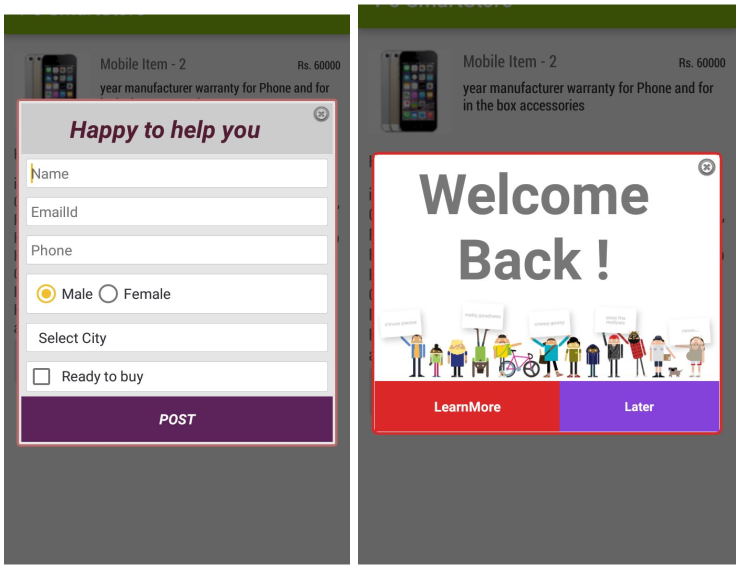
XML

```xml
<service android:name="com.plugin.plumb5.P5GeoMonitorService" />
```

Engagement / In-App / Notifications

- ## Implementing InApp Messaging

There is no extra implementation step required for using in-apps.



- ## Implementing InApp banner Messaging

Java

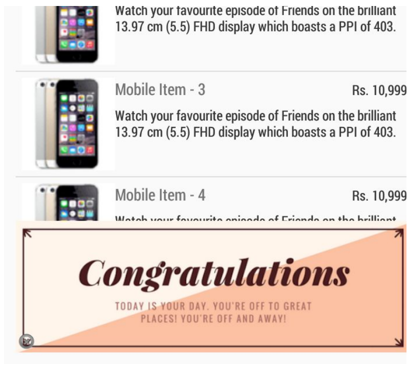
```java
ImageView bmp5Image=(ImageView) findViewById(R.id.p5banner);
ImageView bmp5CloseImage=(ImageView) findViewById(R.id.p5close);
P5Engine.p5LoadBanner(this, bmp5Image, bmp5CloseImage);
```

Need to modify your activity's layout with this below layout.

(Example /app/src/main/res/layout/activity_main.xml in the editor.)

XML

```xml
<FrameLayout
android:layout_width = "wrap_content"
android:layout_height = "wrap_content"
android:layout_centerHorizontal = "true"
android:layout_alignParentBottom = "true" />
<ImageView
android:id= "@+id/p5banner"
android:layout_width = "wrap_content"
android:layout_height = "wrap_content"
android:scaleType = "fitEnd"
android:layout_gravity = "center|bottom" />
<ImageView
android:id= "@+id/p5close"
android:layout_width = "wrap_content"
android:layout_height = "wrap_content"
android:layout_gravity = "left|bottom" />
</FrameLayout>
```



Watch your favourite episode of Friends on the brilliant 13.97 cm (5.5) FHD display which boasts a PPI of 403.

Mobile Item - 3 Rs. 10,999

Watch your favourite episode of Friends on the brilliant 13.97 cm (5.5) FHD display which boasts a PPI of 403.

Mobile Item - 4 Rs. 10,999

Congratulations
TODAY IS YOUR DAY. YOU'RE OFF TO GREAT PLACES! YOU'RE OFF AND AWAY!