



AI & ML DRIVEN  
MARKETING  
AUTOMATION



# Integration document

# Android Development

*This document will help you integrate the appICE SDK in your Android Studio project.*

## Table of Contents

<b>Setting up your app on //appice.io</b>	<b>2</b>
Sign-up	2
Setup your App	2
<b>Including appICE SDK in your project</b>	<b>3</b>
Dependencies	3
appICE SDK	4
<b>Changes to the AndroidManifest.xml</b>	<b>6</b>
Minimum Permissions required for AppICE	6
Adding appICE attributes	7
<b>Initializing appICE in your project</b>	<b>8</b>
Extend your application class	8
appICE sdk Initialization	8
For set Multiple User	11
Setting up Session Timeout	11
Install Referrer	11
FCM Handling	11
Intercepting Campaign Push Messages	12
Handling Custom Firebase FCM Listener	13
<b>Coding Custom Events</b>	<b>14</b>
Define a Custom Event	14
Adding Attributes to the Custom Event	14
Examples	14
// Add a custom event for product viewed by a user along with attributes	14
// Add a custom event on completion of a transaction	15
// Send purchase event with multiple products to sdk	15
<b>Personalizing Notifications</b>	<b>15</b>
Creating ICE-Tags	16
Using ICE-Tags in Notifications	16
DeepLink Action URL's	17
<b>Install App on your Android phone</b>	<b>19</b>
<b>Verify integration on appICE panel</b>	<b>20</b>

# Setting up your app on //appice.io

## Sign-up

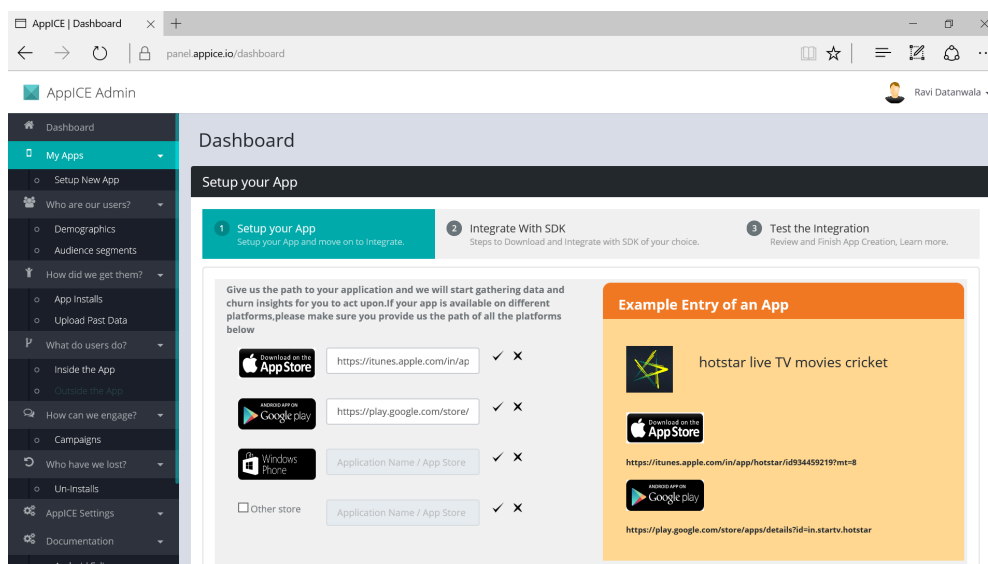
You need to sign-up and create an account with appICE.io.

1. Visit <https://panel.appice.io/Signup> to register
2. This brings you to the "Sign up" page
3. Provide your Name, Email address and your chosen password and create your account
4. Login using your username and password credentials once your account is approved

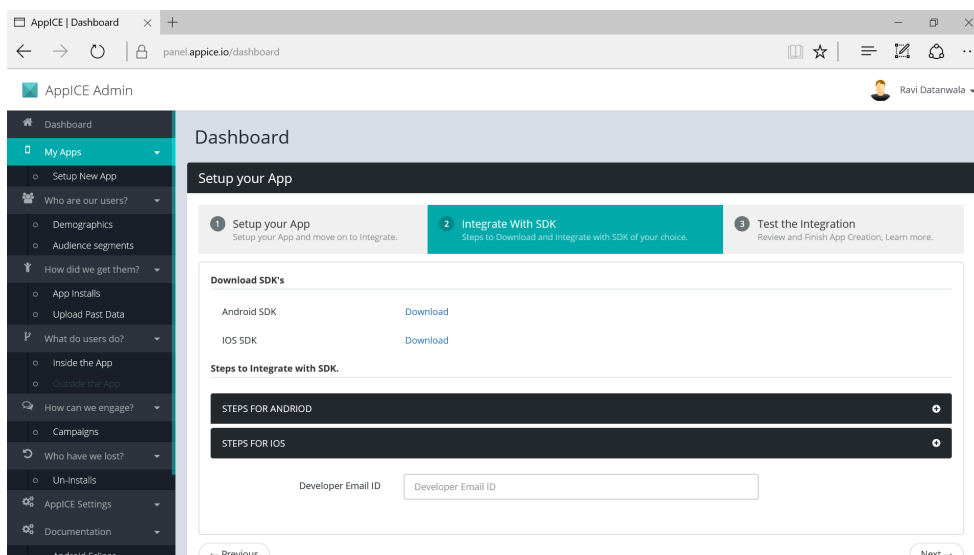
## Setup your App

To start the process of integrating appICE in your app, you will first need to setup your app on the appICE dashboard.

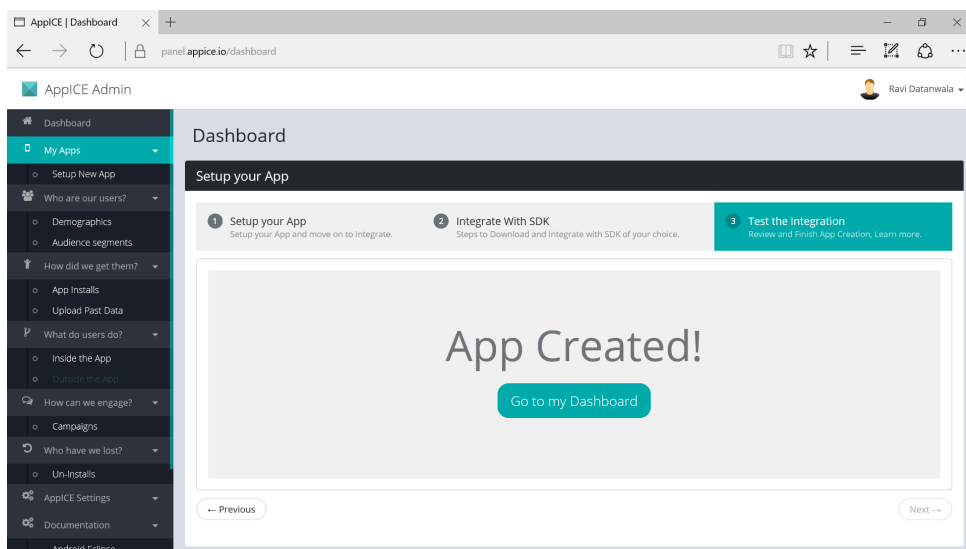
1. Provide the link to your mobile app on the Google Play Store as well as Apple store if you have both versions of your app. It will automatically pick If your app is not yet published on the play stores, you can just provide <AppName> and click on the ✓ to confirm the same



2. Now you have access to the SDK's for Android/iOS including the instructions for integration. To make it easier, we have also provided the option to provide your developer's email address so that those instructions can be emailed to them directly.



- Now you are all ready to start receiving data from your mobile app once the developer completes the integration and publishes the app again on the app store.





## Dependencies

The appICE SDK has dependencies on the Google Play services. You would include to include them in your project. If you have already included the Google Play services, you can ignore this step. If you're using Android Studio, add the following lines to the dependency section of your application's build.gradle file:

```
dependencies {
    implementation 'com.google.android.gms:play-services-ads:20.0.0'
    implementation 'com.google.android.gms:play-services-location:18.0.0'
    implementation 'com.google.firebase:firebase-core:19.0.2'
    implementation 'com.google.firebase:firebase-messaging:23.0.6'
}
```

**Note:** If you have included Google Play Services versions higher than 7.8 it is absolutely fine. Also update firebase entries with the latest version.

## appICE SDK

Add the following lines to the dependency section of your application's build.gradle file:

```
dependencies {
    implementation 'appice.io.android:sdk:2.5.75'
}
```

This you have to add in either setting.gradle or root build.gradle depends on your Android Studio configuration.

```
maven {
    url "https://gitlab.com/api/v4/projects/10636887/packages/maven"
    credentials(HttpHeaderCredentials) {
        name = "Deploy-Token"
        value = "PJMsxXdArqsmqDx4x5B6"
    }
    authentication {
        header(HttpHeaderAuthentication)
    }
}
```

This would ensure that the latest SDK version is always included in your project.

**NOTE:** Be sure to perform a Gradle Sync to build your project and incorporate the dependency additions noted above.

## Handling proguard entries

---

If your project is having proguard entries in your app module build.gradle file.

Then under your build.gradle file you need to add up appice proguard file as well.

Get a copy of appice proguard rules from below link and add it to your build.gradle entry.

[https://webcdn.appice.io/assets/proguard\\_rule.txt](https://webcdn.appice.io/assets/proguard_rule.txt)

File name is as below

“proguard-rules-appice.pro”

```
android
{
    buildTypes {
        release{
            minifyEnabled true
            proguardFiles getDefaultProguardFiles('proguard-android.txt'), proguard-rules.pro,
proguard-rules-appice.pro
        }
        debug {
            minifyEnabled true
            proguardFiles getDefaultProguardFiles('proguard-android.txt'), proguard-rules.pro,
proguard-rules-appice.pro
        }
    }
}
```



# Changes to the AndroidManifest.xml

## Minimum Permissions required for AppICE

You also need to add the following permissions (ignore if already included):

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
<uses-permission android:name="com.google.android.gms.permission.AD_ID" />
<uses-permission android:name="android.permission.POST_NOTIFICATIONS" />
<uses-permission android:name="android.permission.USE_EXACT_ALARM" />
```

## Adding appICE attributes

To ensure that the data is added to the right application you created on the appICE dashboard, you need to add the following highlighted block in the <application> ... </application> section

```
<manifest package="com.example.gcm" ...>
  <application ...>
    <meta-data android:name="com.semusi.analytics.appid" android:value="Your_AppID"/>
    <meta-data android:name="com.semusi.analytics.appkey" android:value="Your_AppKey"/>
    <meta-data android:name="com.semusi.analytics.apikey" android:value="Your_ApiKey"/>

    // for region value
    <meta-data android:name="io.appice.analytics.region" android:value="US" />

    <meta-data
      android:name="com.google.android.gms.ads.APPLICATION_ID"
      android:value="ca-app-pub-3940256099942544~3347511713"/>

  </application>
</manifest>
```

**Note :** *ca-app-pub-3940256099942544~3347511713* This is a sample metadata provided by google (<https://developers.google.com/admob/android/quick-start>) , Please create your own APPLICATION\_ID

You need to replace the text above in BLUE with the information which can be found on your appICE Dashboard under Settings.

### Why play-services-ads sdk ?

To get GAID from the device, we rely on play-services-ads. GAID helps us uniquely identify devices, allowing us to send notifications and in-app to particular devices.

However, to use GAID and other features provided by the Google SDK, we need to follow some specific steps.

1. In our Android app's manifest file, we must include a permission declaration:

```
<uses-permission android:name="com.google.android.gms.permission.AD_ID" />
```

2. we need to specify metadata in the manifest file:

```
<meta-data
```

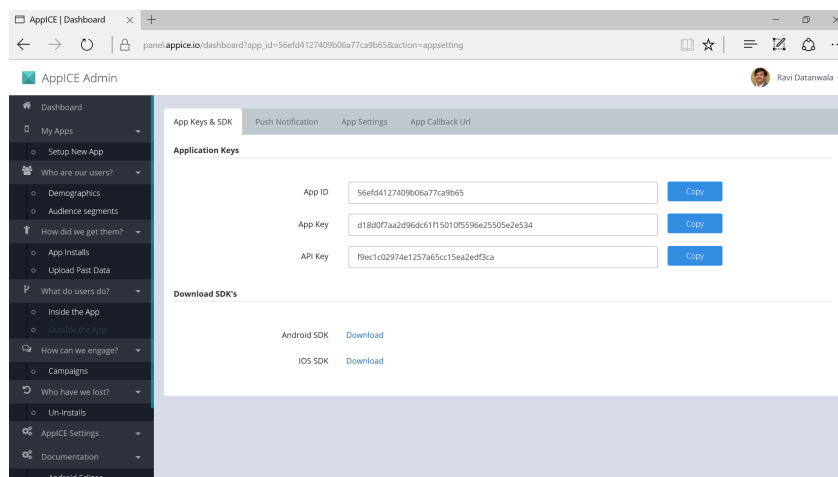
```
    android:name="com.google.android.gms.ads.APPLICATION_ID"
```

```
    android:value="ca-app-pub-3940256099942544~3347511713" />
```

Please note that "ca-app-pub-3940256099942544~3347511713" is a sample ad ID. you need to replace it with your own unique ID.

3. In the project's build.gradle file, we have to include a specific dependency:

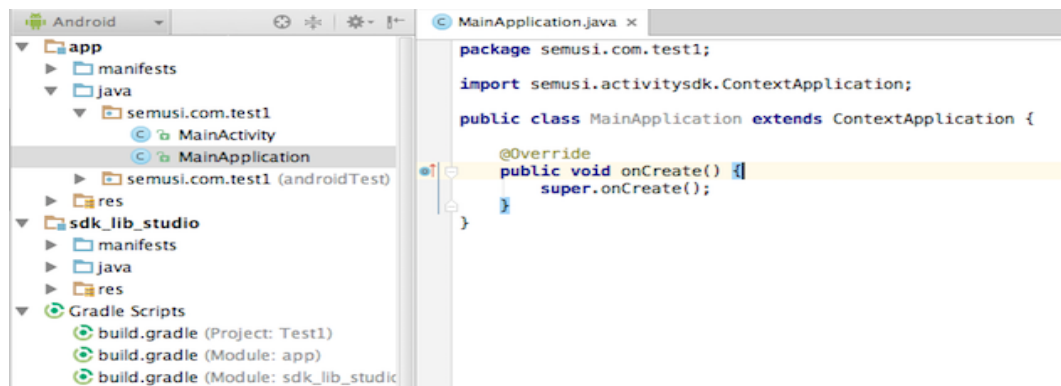
```
implementation 'com.google.android.gms:play-services-ads:20.0.0'
```



# Initializing appICE in your project

## Extend your application class

Extend your application class from 'ContextApplication' and update its entry in your AndroidManifest.xml file



Sample of AndroidManifest.xml file entry. Considering that your Application class is under package 'com.semusi.test1' with class name 'MainApplication'

```
<application
    android:icon="@drawable/logo"
    android:name="com.semusi.test1.MainApplication"
    android:label="AppiceTest">
    ...
</application>
```

If your application already extends any Application class then call below function from your application class 'onCreate()' function

*ContextApplication.initSdk(<instance of context>, <instance of application class>);*

Example:

```
public class MainApplication extends Application {
    @Override
    public void onCreate() {
        super.onCreate();

        ContextApplication.initSdk(getApplicationContext(), this);

    }
}
```

## appICE sdk Initialization

Now to initialize appICE sdk, you can set it up under any Activity, Service or Application class as shown below.

```
SdkConfig config = new SdkConfig();
```

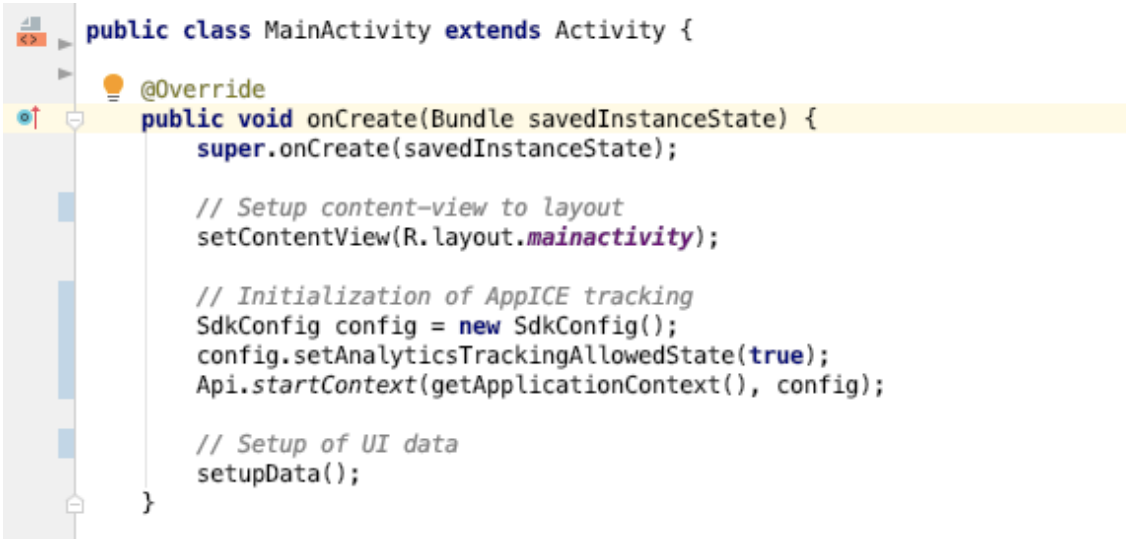
```
config.setAnalyticsTrackingAllowedState(true);
```

```
// Init sdk with your config
```

```
Api.startContext(ctx, config);
```

```
// for region initialization (This can be handled via Meta data from Manifest file or Application Class )
```

```
Api.initSdk("xxxxxx","xxxxx","xxxxxx","US",ctx);
```



```
public class MainActivity extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        // Setup content-view to layout
        setContentView(R.layout.mainactivity);

        // Initialization of AppICE tracking
        SdkConfig config = new SdkConfig();
        config.setAnalyticsTrackingAllowedState(true);
        Api.startContext(getApplicationContext(), config);

        // Setup of UI data
        setupData();
    }
}
```

Example:

```
public class MainActivity extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.mainactivity);

        SdkConfig config = new SdkConfig();
        // To enable analytics tracking - default enabled
        config.setAnalyticsTrackingAllowedState(true);

        Api.startContext(getApplicationContext(), config);
    }
}
```

```
}
}
```

## Native Display

getCampaigns(type: string): Fetches all campaigns of a specific type (e.g., "NATIVE","IN-APP")

```
List<Campaign> campaigns = new ArrayList<>();
campaigns = ContextSdk.getCampaigns("NATIVE", context);
for (Campaign campaign : campaigns) {
    Log.d("TAG", "getCampaigns : campId: " + campaign.getCampId());
    Log.d("TAG", "getCampaignById : actionType: " + campaign.getActionType());
    Log.d("TAG", "getCampaigns : customDataValue: " +
campaign.getcustomData());
    if (campaign.getActionType() != null) {
        if (campaign.getActionType().equals("lp")) {
            Log.d("TAG",
                "getCampaigns : Landing Page Url: " + campaign.getActionUrl());
        } else if (campaign.getActionType().equals("dl")) {
            Log.d("TAG", "getCampaigns : Deeplink Url: " +
campaign.getActionUrl());
        } else {
            Log.d("TAG", "getCampaigns : Action Url: " + campaign.getActionUrl());
        }
    }
}
```

getCampaignById(cmpId: string): Retrieves a campaign by its ID. Uses a generic getProperty method to access campaign properties like ID, action URL etc

```
Campaign campaign = ContextSdk.getCampaignById("1212", context);

if (campaign != null) {
    Log.d("TAG", "getCampaignById : campId: " + campaign.getCampId());
    Log.d("TAG", "getCampaignById : campId: " + campaign.getcustomData());

    Log.d("TAG", "getCampaignById : actionType: " + campaign.getActionType());
    if (campaign.getActionType() != null) {
        if (campaign.getActionType().equals("lp")) {
            Log.d("TAG",
                "getCampaigns : Landing Page Url: " + campaign.getActionUrl());
        } else if (campaign.getActionType().equals("dl")) {
            Log.d("TAG", "getCampaigns : Deeplink Url: " +
campaign.getActionUrl());
        } else {
            Log.d("TAG", "getCampaigns : Action Url: " + campaign.getActionUrl());
        }
    }
}
```

```

    }
}
Log.d("TAG",
    "getCampaignById : customDataValue: " + campaign.getcustomData());
} else {
    Log.d("TAG", "Campaign not found");
}

```

## For set Multiple User

=> `ContextSdk.setUser(String[] userID, Context ctx)`

### User Profile - getUser :

getUser which has the userObject like userDetails, Demographic

1. **getUser** - without UserId : It will return User class object

- **ContextSdk.getUser(Context ctx)**

2. **getUserForIds**

- with UserId : this will return ArrayList of user class

**ContextSdk.getUser(ArrayList<String> userList, Context context)**

-before calling getUserIds we have to synchronizeData to fetch the latest activities

**ContextSdk.synchronizeData(IAppICEDataCallback callback, int timeoutInSec, Context context)**

**ContextSdk.getUser(ArrayList<String> userList, Context context)**

**UserDetails** : such as Name, Phone, Gender, Dob, EducationType, Email, EmploymentType, Age, Married, IsEmployed.

**Demographic**: such as FIRST\_SEEN, LAST\_SEEN, TOP\_N\_PRODUCTS\_VIEWED, N\_COMPLAINTS\_RAISED, PREF\_LOGIN\_DEVICE, REFERRAL\_CAMPAIGN, TOTAL\_ORDER\_VALUE, ADD\_TO\_CART\_N\_DAYS, CREDIT\_SCORE, DEBT\_TO\_INCOME\_RATIO, SAVINGS\_BALANCE, CHECKING\_BALANCE

## Setting up Session Timeout

By default session timeout is of 30 seconds, but you can setup your time of session management to the AppICE sdk.

To setup or gather current session info by the user.

```
ContextSdk.setSessionTimeout(30, getApplicationContext());
```

```
ContextSdk.getSessionTimeout(getApplicationContext());
```

## Install Referrer

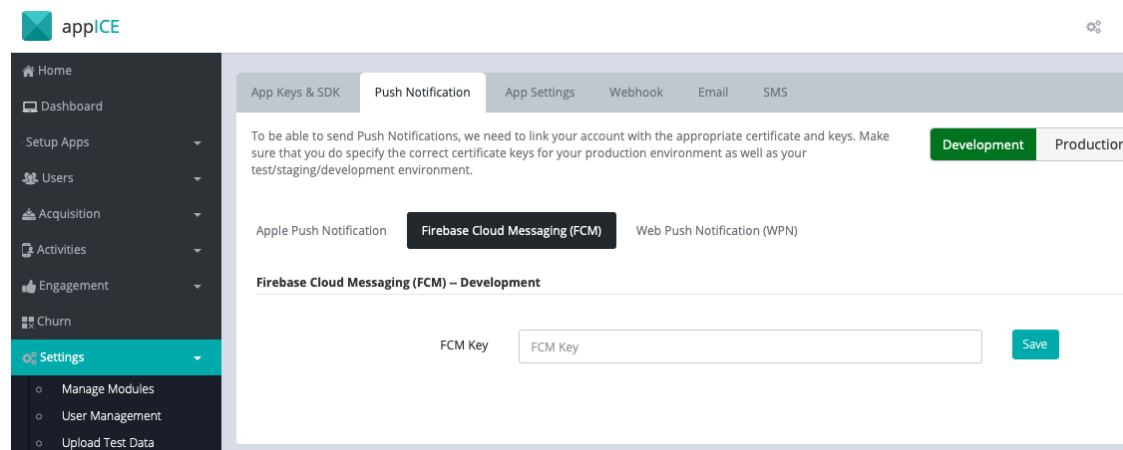
Please add the following block in the application section of your AndroidManifest.xml file

```
<receiver
    android:name="semusi.analytics.handler.InstallReferralHandler"
    android:exported="true">
    <intent-filter>
        <action android:name="com.android.vending.INSTALL_REFERRER" />
    </intent-filter>
</receiver>
```

## FCM Handling

If your app already has FCM enabled for push notifications handling, you need to update the same

Update your FCM Server Key on the AppICE Settings tab in the Dashboard. You need to update FCM key in both Development and Production tabs. Toggle and press save button next to input field respectively.



## Intercepting Campaign Push Messages

To intercept push campaign events by user, setup a broadcast receiver to intercept them in your AndroidManifest.xml file

```
<!-- Events receivers from appice sdk -->
<receiver android:name="com.myapp.sample.CampaignEventReceiver" >
    <intent-filter>
        <action android:name="com.appice.CampaignEvent" />
    </intent-filter>
</receiver>
```

Now, create a custom class to handle broadcast of campaign events in your app package and then handle event as per your requirements

```
public class CampaignEventReceiver extends BroadcastReceiver
{
    @Override public void onReceive(Context context, Intent intent)
    {
        // value of extUrl will be data - which you have sent from campaign
        extUrl = intent.getStringExtra("Exturl");

        // value of clicked is True/False based upon user click or not

        String clicked = intent.getStringExtra("Clicked");

        // System.out.println("CampaignEvent Recevied with ExtUrl : " + extUrl
        //                      + " , " + clicked);
    }
}
```

To set the **Campaign Clicked** event from the app side, user can call this method in their broadcast receiver.

**ContextSdk.pushNotificationClicked( messagePayload, context)**



## Handling Custom Firebase FCM Listener

If your application is using Google play services 7.3+ and above, then GCM InstanceID is returned after device registration on GCM.

### Step 1 :

Check in your AndroidManifest file for entry somewhat similar to below

```
<service android:exported="false" android:name="com.test.sample.MyFcmListenerService">
    <intent-filter>
        <action android:name="com.google.firebase.MESSAGING_EVENT"/>
    </intent-filter>
</service>
```

\* If you have custom FCM MESSAGING\_EVENT service intent-filter then add the following in your custom class MyGcmListenerService.

Under method **onMessageReceived**

```
@Override
public void onMessageReceived(RemoteMessage message) {
    // Pass message bundle to appice layer
    new SdkFcmListenerService().onMessageReceived(message, getApplicationContext());
}
```

To set the **Campaign Received** event from the app side call this method in fcm listener:

*ContextSdk.pushNotificationReceived( messagePayload, context)*

```
// rest of your app code
}

@Override
public void onNewToken() {
    // Call appice layer to update token
    new SdkFcmListenerService().onTokenRefresh(getApplicationContext());

    // rest of your app code
}
```

# Coding Custom Events

---

You can add Custom events in your Android apps to capture user actions along with associated data attributes.

## Define a Custom Event

```
HashMap<String, String> eventKey = new HashMap<String, String>();
ContextSdk.tagEvent("Event_SignIn", event_key);
```

The above code snippet creates a custom event called 'Event\_SignIn' in your Dashboard.

## Adding Attributes to the Custom Event

You can pass data attributes along with the Event for granularity:

```
HashMap<String, String> eventKey = new HashMap<String, String>();
eventKey.put("userid", "john.doe@appice.io");
eventKey.put("time", "20-feb-2016 10:15:14 AM");
ContextSdk.tagEvent("Event_SignIn", event_key);
```

The above code snippet highlighted adds 2 data attributes to the Custom Event Event\_SignIn. It added the userid and time of event.

## Examples

### // Add a custom event for product viewed by a user along with attributes

```
HashMap<String, String> eventKey = new HashMap<String, String>();
eventKey.put("userid", "rick.john@appice.io");
eventKey.put("productId", "a2f32827ad");
eventKey.put("productCost", 520);
ContextSdk.tagEvent("Event_ViewProduct", event_key);
```

### // Add a custom event on completion of a transaction

```
HashMap<String, String> eventKey = new HashMap<String, String>();
eventKey.put("userid", "rick.john@appice.io");
eventKey.put("productIds", ["28a739a383", "28a739a383"]);
eventKey.put("orderId", "28a739a383");
eventKey.put("totalCost", 690);
eventKey.put("transactionStatus", "success");
ContextSdk.tagEvent("Event_TxnCompleted", event_key);
```

---

## // Send purchase event with multiple products to sdk

```

HashMap<String, Object> map = new HashMap<>();
try {
    // Product array list
    JSONArray plist = new JSONArray();

    // Product 1st object
    JSONObject p1 = new JSONObject();
    p1.put("Product_ID", 123);
    p1.put("Category", "product_1");
    p1.put("Price", 100);

    // Product 2nd object
    JSONObject p2 = new JSONObject();
    p2.put("Product_ID", 456);
    p2.put("Category", "product_2");
    p2.put("Price", 150);

    // Add all product to product list
    plist.put(p1);
    plist.put(p2);

    // Add all product to cart
    map.put("Cart", plist);

    // Add cart value to map
    map.put("Cart_Value", 450);
}
catch (Exception e) {
}

// Generate event in sdk of 'Purchase Event'

HashMap<String, String> newMap = new HashMap<String, String>((HashMap) map);
ContextSdk.tagEvent("Purchase Event", newMap);

```

## JS Code sample

---

### Android AppICE Hybrid Bridge:

#### Native app side changes :

```
webView.getSettings().setJavaScriptEnabled(true);
```

```
String url = "webUrl?isWebView=1"
```

```
// this line is compulsory
```

```
webView.addJavascriptInterface(new AppICEWebBridge(), "AppICEWebBridge");
```

```
webView.loadUrl(url);
```

#### **Note:**

<https://support.google.com/faqs/answer/7668153?hl=en>

### WebApp should have AppICE WebSDK integrated internally.

So when the user clicks on a button inside the website and if the web team is using recordEvent (from WebSDK ) on that button then the event will be captured in the same session.

#### Web app side Example:

##### Function call :

##### Use this index.js

<https://webcdn.appice.io/sdk/index.js>

You have to use this index.js for calling native sdk.

##### Record Events

Ex:

---

```
function recordAppICEEvent() {
  var key = document.getElementById("recordKey").value;
  var segmentKey = document.getElementById("SegmentK").value;
  var segmentValue = document.getElementById("SegmentV").value;
  var props = {
    segmentKey: segmentValue
  };

  // this is index.js function which is responsible for calling native functions
  recordEvent(key,props)
}
```

### setUserId

```
function setAppICEUserId() {
  var userId = document.getElementById("userId").value;
  const userIdArray = [
    userId
  ];

  // this is index.js function which is responsible for calling native functions
  setUserId(userIdArray);
}
```

### setUser

```
function setAppICEUserDetail() {

  var name = document.getElementById("name").value;
  var phone = document.getElementById("phone").value;
  var email = document.getElementById("email").value;
  var age = document.getElementById("age").value;
  var dob = parseInt(document.getElementById("dob").value);
  var education = document.getElementById("education").value;
  var gender = document.getElementById("gender").value;
  var employed = document.getElementById("employed").value;
  var employmentType = document.getElementById("employmentType").value;
  var married = document.getElementById("married").value;

  var userDetails = {
    "n": name,
    "p": phone,
    "e": email,
    "a": age,
    "d": dob,
    "edt": education,
    "g": gender,
    "em": employed,
    "emt": employmentType,
    "m": married
  };
}
```

---

```
// this is index.js function which is responsible for calling native functions  
setUser(userDetails)  
}
```

## setCustomVariable

```
function setAppICECustomVariable() {  
  var customKey = document.getElementById("customKey").value;  
  var customValue = document.getElementById("customValue").value;  
  var parsedValue;  
  if (!isNaN(customValue)) {  
    if (customValue.includes('.')) {  
      parsedValue = parseFloat(customValue); // float or double  
    } else {  
      parsedValue = parseInt(customValue); // int or long  
    }  
  } else if (customValue.toLowerCase() === 'true' || customValue.toLowerCase() === 'false') {  
    parsedValue = customValue.toLowerCase() === 'true';  
  } else {  
    parsedValue = customValue;  
  }  
  
// this is index.js function which is responsible for calling native functions  
  setCustomVariable(customKey, parsedValue);  
}
```

# Personalizing Notifications

appICE allows you to personalize your notifications (push/in-app) to ensure that your users can feel more engaged in the app. In order to enable personalization in your campaigns, you would be making use of “ICE-Tags”

## Creating ICE-Tags

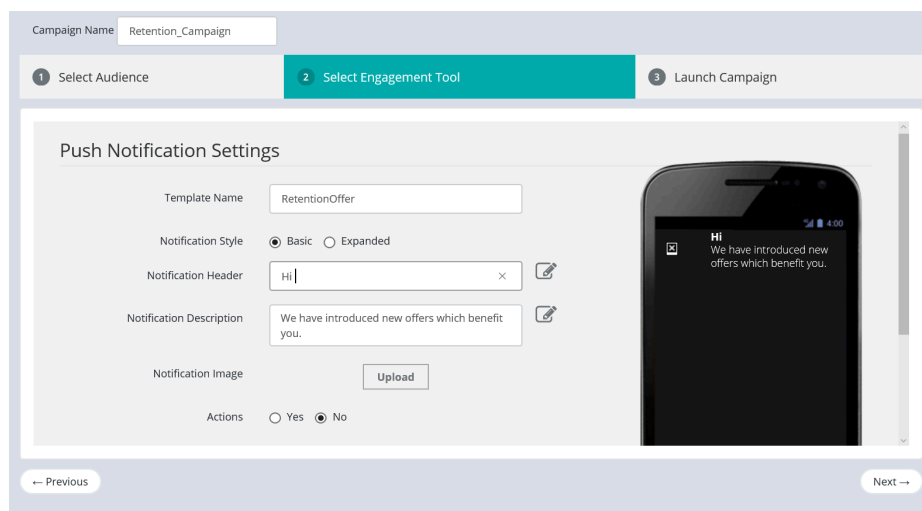
You can use your own application/CRM data and pass that to the appICE services to be used in push/in-app notifications. Ideally this would be done as soon as the app starts and you can set the values in these ICE-Tags.

```
ContextSdk.setCustomVariable("CustomerType", "<Value>", ctx);
ContextSdk.setCustomVariable("CustomerStatus", "<Value>", ctx);
```

The above code snippet creates 2 ICE-Tags which can be populated with the right from your internal systems.

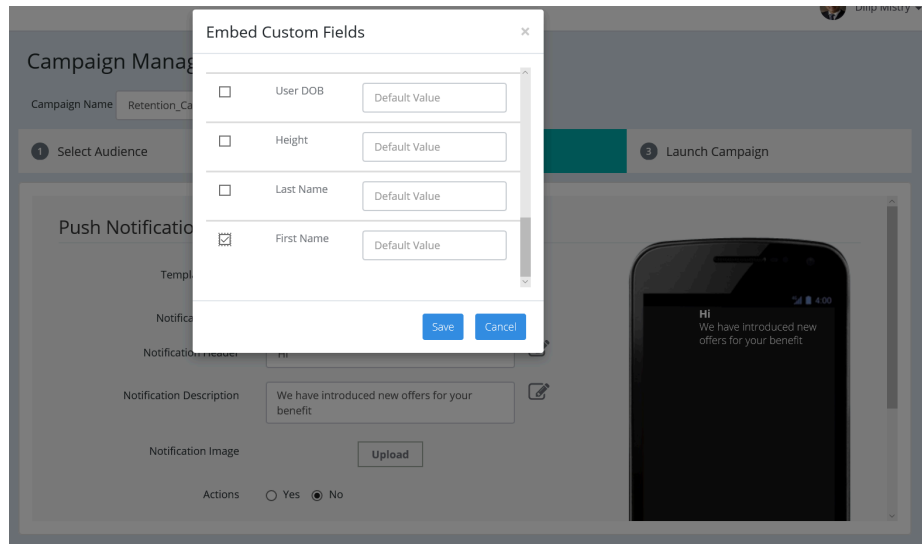
## Using ICE-Tags in Notifications

Once you have custom data setup. You can use them for setting up tokens for campaigns. To personalize campaign contents.

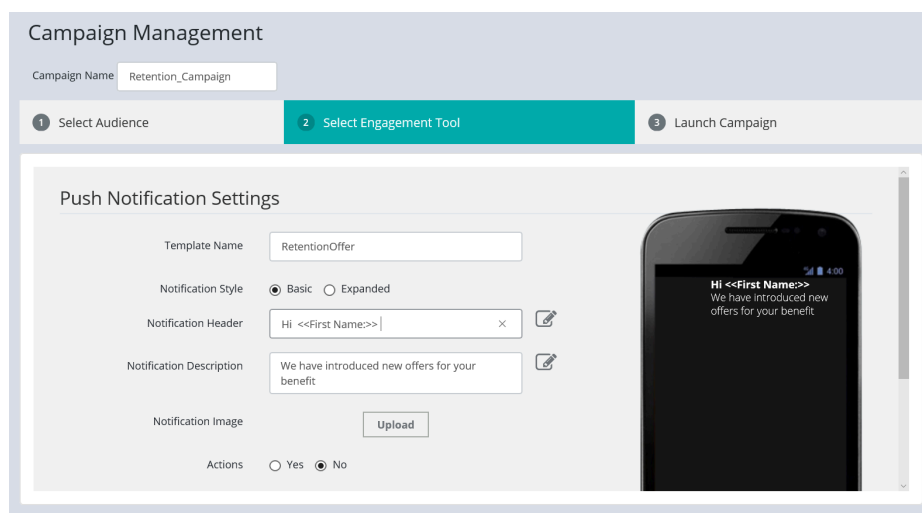


The screenshot shows the 'Push Notification Settings' screen in the appICE interface. At the top, there's a 'Campaign Name' field with 'Retention\_Campaign' entered. Below this are three steps: '1 Select Audience', '2 Select Engagement Tool' (highlighted in teal), and '3 Launch Campaign'. The main section is titled 'Push Notification Settings' and contains several fields: 'Template Name' (RetentionOffer), 'Notification Style' (Basic selected, Expanded unselected), 'Notification Header' (Hi), 'Notification Description' (We have introduced new offers which benefit you.), and 'Notification Image' (Upload button). There are also 'Actions' (Yes/No) and 'Previous/Next' navigation buttons at the bottom. A preview of a smartphone notification is shown on the right.

Click on the edit buttons next to ‘Notification Header’ OR ‘Notification Description’ to add ICE-Tags in the content.



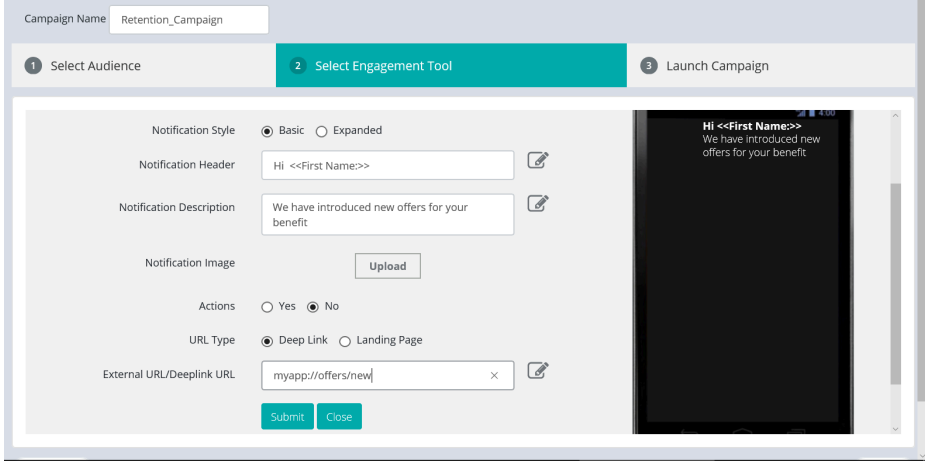
This would at run-time personalize your notification and replace the <<First Name>> ICE-Tag with the actual name.



## DeepLink Action URL's

You can also deeplink the action to go to a specific section of the app based on your campaign. In order to deeplink, make sure that you have handling incoming deeplinks in your app and in the Actions URL, you can specific the deeplink.





To support deeplink handling in your application below steps are required to follow:

### Step1: Enable activity to handle deeplink scheme://host

```
<activity
    android:name="com.sample.app.CartActivity">

    <intent-filter>
        <data
            android:host="product_cart"
            android:scheme="myapp" />
        <action android:name="android.intent.action.VIEW" />

        <category android:name="android.intent.category.DEFAULT" />
        <category android:name="android.intent.category.BROWSABLE" />
    </intent-filter>
</activity>
```

In above activity select of intent-filter is required with category of DEFAULT and BROWSABLE

- Section of 'data' is required to specify your scheme ex. 'myapp' and host as screen to handle deeplink data.

- Action is also required as VIEW

### Step 2: Handle incoming data in activity

```
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.splashactivity);

    // gather deeplink data
    HashMap<String, Object> deeplinkData = ContextSdk.gatherDeepLinkData(getIntent());
    if (deeplinkData != null) {
        // process received data
    }
}
```

```
@Override
public void onNewIntent(Intent intent) {
    // set new intent to current context
    this.setIntent(intent);
}
```

In above we need to handle onCreate function to fetch incoming deeplinking data as HashMap object

And override onNewIntent function so, that any open activity can also received incoming data.

Step3: Setting up Deeplink at appice campaign dashboard

As your have used in above example that

Scheme: myapp

host : product\_cart

Then your deeplink url will be :

myapp://product\_cart

You can also pass any other data in terms of query params.

myapp://product\_cart?data1=value1&data2=value2

To get the **getInternalId** of the device user can use following method:

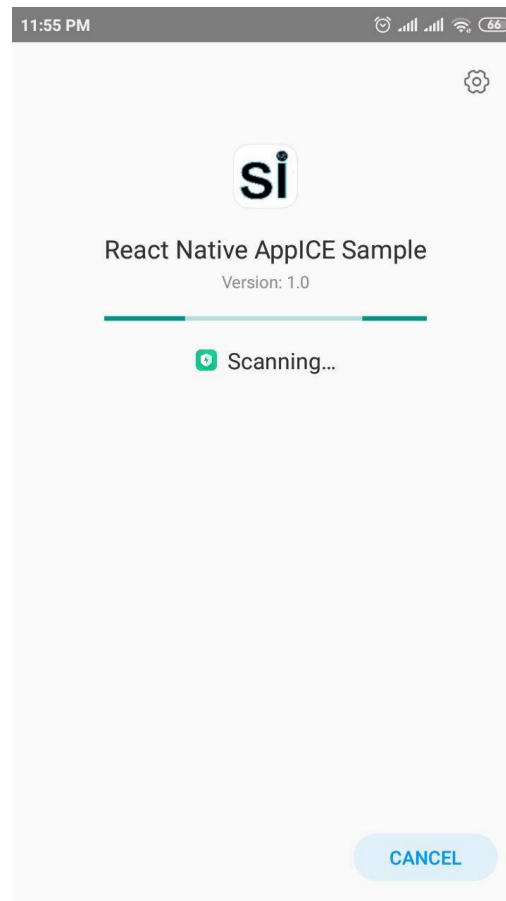
```
String internalId = ContextSdk.getInternalId(context)
```

--

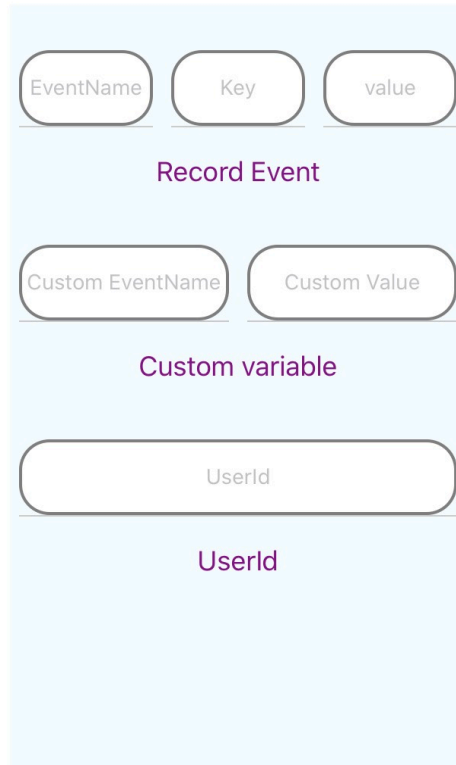
## Install App on your Android phone

---

1. Download & install app on your Android phone.



2. Launch/Open/Initialize app. It shows three functions - "Record Event", "Custom Variable" & "UserId".
-



The screenshot shows a light blue background with three distinct input sections, each separated by a horizontal line. The first section is labeled 'Record Event' in purple text and contains three rounded rectangular input fields labeled 'EventName', 'Key', and 'value'. The second section is labeled 'Custom variable' in purple text and contains two rounded rectangular input fields labeled 'Custom EventName' and 'Custom Value'. The third section is labeled 'UserId' in purple text and contains a single wide rounded rectangular input field labeled 'UserId'.

3. For Record Event, put 3 values:

EventName

Key

Value

and click on the 'Record Event' button.

Test

KeyName

Demo

Record Event

Custom EventName

Custom Value

Custom variable

UserId

UserId

4. For Custom Variable, put 2 values:

Custom EventName

Custom Value

and click on the 'Custom Variable' button.

EventName

Key

value

Record Event

CustomTest

CustomVal

Custom variable

UserId

UserId

5. For UserId, put 1 value

UserId

and click on the 'UserId' button.

EventName

Key

value

Record Event

Custom EventName

Custom Value

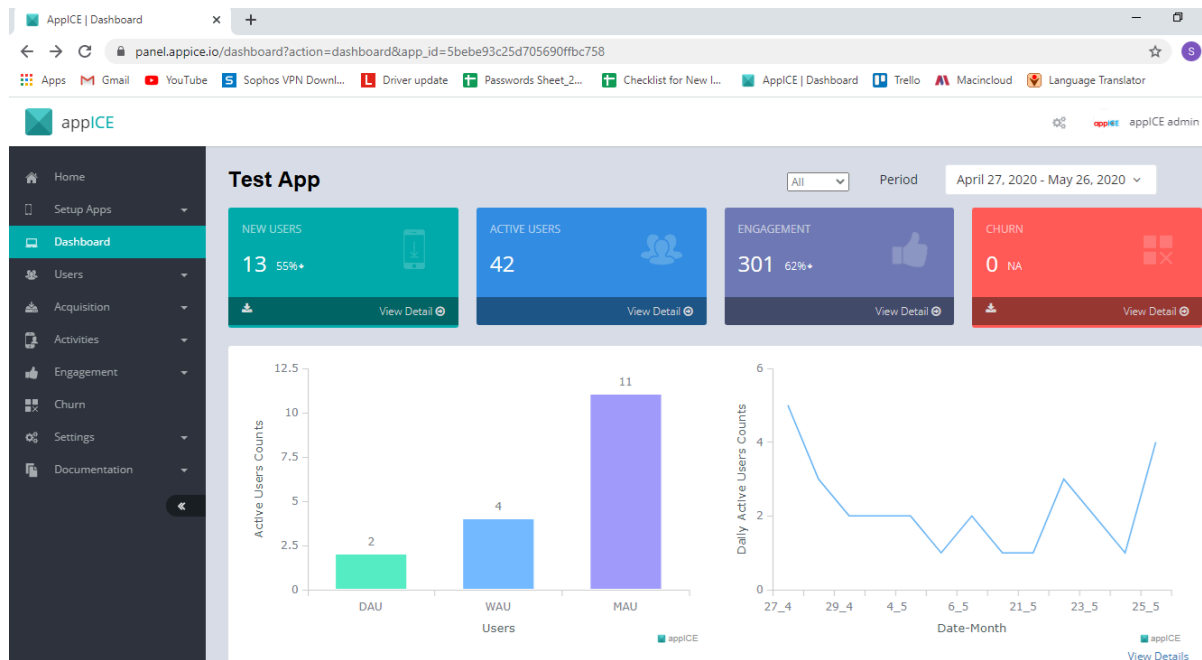
Custom variable

Dummy123

UserId

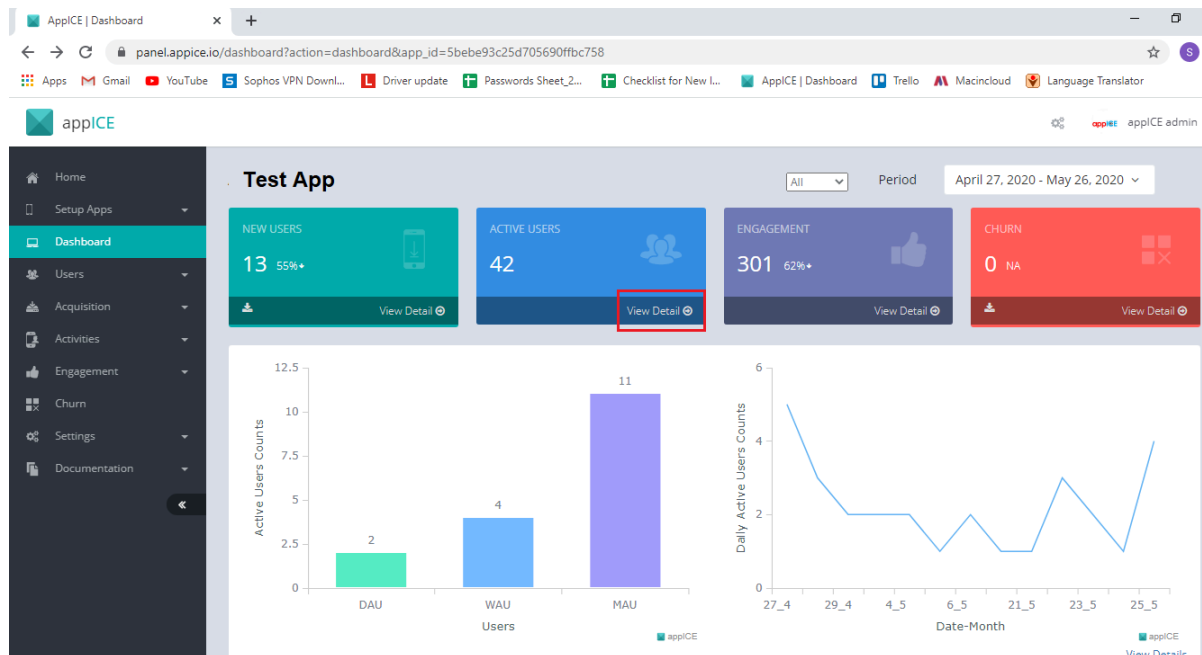
# Verify integration on appICE panel

1. Login to panel.appice.io to see these values. Click on your app and go to its dashboard.

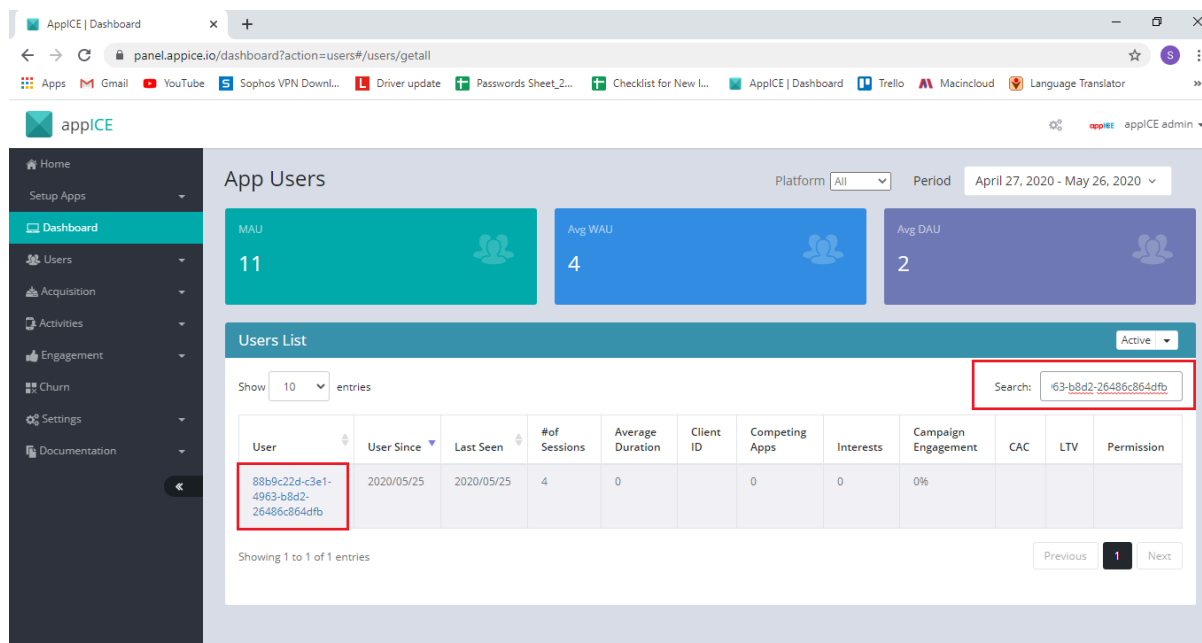


2. Click on 'Active Users' ☐ 'View Detail' to see details of your phone app.





- Clicking on 'View Detail' takes to App Users page. Search your phone Advertising Id in Search box to search for your phone Ads Id.



- Click on the searched Ad Id to see details:

panel.appice.io/dashboard?action=users#/users/details/69becb856b2169020b46edd644ab65e1252bcc31

Apps Gmail YouTube Sophos VPN Downl... Driver update Passwords Sheet\_2... Checklist for New L... AppICE | Dashboard Trello Macincloud Language Translator

appICE appICE admin

Home

Setup Apps

**Dashboard**

Users

Acquisition

Activities

Engagement

Churn

Settings

Documentation

88b9c22d-c3e1-4963-b8d2-26486c864dfb

Demographics

Gender: N/A

User Attributes

Variable	Value
Model	Xiaomi Redmi 9T
Carrier	Vodafone Be Safe
App Version	1.0

Competing Apps

Interests

Recent Activity

05/25/20 23:56:52	App_Background
05/25/20 23:56:32	App_Foreground
05/25/20 23:56:23	App_Background
05/25/20 23:56:17	Test recordEventIdlay : Test
05/25/20 23:55:49	App_Foreground
05/25/20 23:55:49	Session_Start
05/25/20 23:21:52	App_Background
05/25/20 23:21:47	App_Foreground
05/25/20 23:21:38	App_Background

Recency

Seen 28 mins 0 secs ago

Monetary

0

Campaign Engagement

0

CAC / LTV

0